Introduction to Database Management System

Introduction

In today's digital world, organizations generate vast amounts of data every second. This data needs to be saved, processed and organized in a proper manner so that it can be used later to make decisions. Understanding the difference between data and information thus becomes important for effective data management. A database serves as a structured repository allowing for efficient retrieval and manipulation of data. In this chapter, we will learn about the difference between data and information. We will look at the Database Management System (DBMS) and its components. We will also create a template of a sample database and learn about data types that play an important role in databases.

Data vs Information

Sometimes, the users use the terms Data and Information synonymously. Therefore, it is necessary to know the exact meaning of both these terms. Data and information can be defined in a number of ways, so let us start by figuring out what those might be.

Data refers to unorganized facts, figures and details related to people, places, things or events. It is often considered to be raw. Data can be in any format like numbers, text, images or video . It could be written or spoken. Data in its raw state may not be very helpful. Given the significance of data in the decision-making process, several companies view it as a key asset. The basic property of data is that it is irrelevant at times and often unstructured.

Information is processed data. The raw data when passed through a transformation process of some kind gets converted into information. The basic property of information is that it is structured, relevant and meaningful.

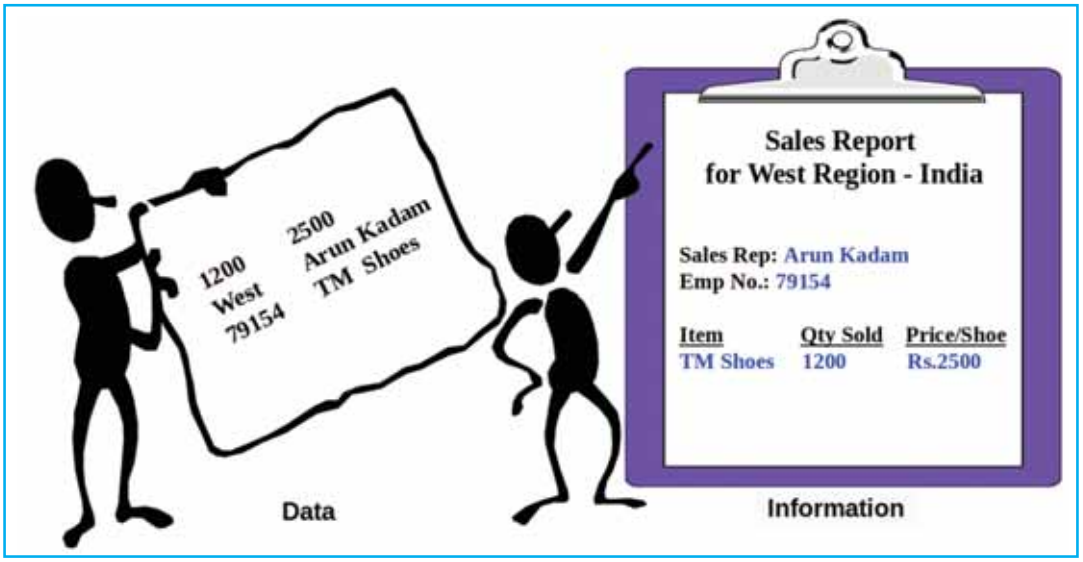


Figure 1.1 : Difference between Data and Information

Let us try to understand the difference between data and information by taking an example. A sequence of numbers and text 1200, 2500, 79154, "West", "Arun Kadam", and "TM Shoes" is just data. Individually each of them does not seem to be relevant. However, if you put it into context: "Mr. Arun Kadam, Emp No.: 79154 has sold 1200 TM Shoes at Rs. 2500 each in the West Region of India", it becomes information because it provides meaning and context. Figure 1.1 shows the difference between data and information.

What is Database?

Today all of us use one or more types of databases in our day-to-day life. A most understandable example of a database would be the contact book on our mobile phone that contains contact numbers and emails of our acquaintances. The details pertaining to an individual acquaintance is known as a record.

Database is a collection of multiple related data items stored in a properly organized manner. According to this definition, there are two key points to be considered when we mention a database. First; it must represent data items that are related to each other and second, it must be an organized collection.

Logical arrangement of things makes searching always easier as and when required. The databases are often designed according to a predefined set of rules and data models. The data model describes a way of storing and retrieving the data. There are different data models in DBMS like hierarchical, network, relational, object-oriented, flat, semi-structured, associative and context. Two of these models, flat and relational, play a significant role. The flat model was the initial model of storing data while the relational model is the most widely used model today.

Flat Data Model

Data is represented in the flat data model as a single, two-dimensional table with no defined connections. The quantity and kind of fields are consistent across all records. Due to its simplicity and ease of comprehension, this format is ideal for small or one-time datasets. This format is frequently used in comma-separated values (CSV), LibreOffice Calc and MS Excel files. A sample dataset of a CSV file and LibreOffice Calc is shown in figure 1.2 and 1.3.

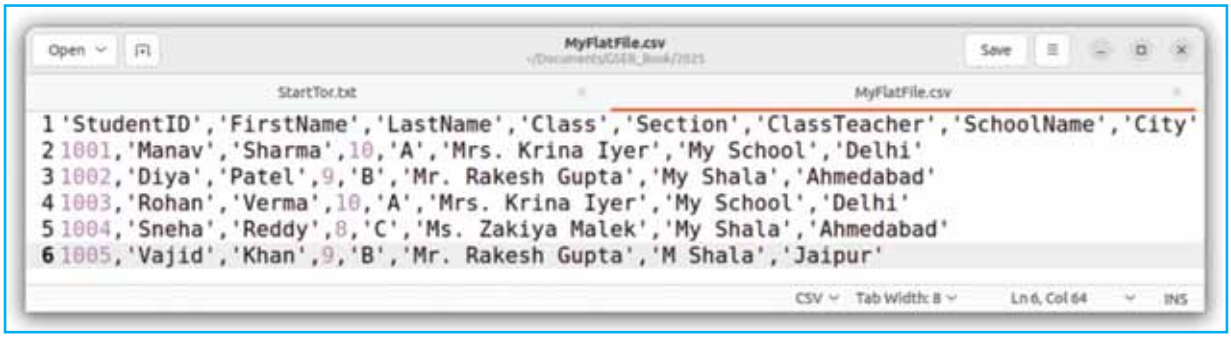
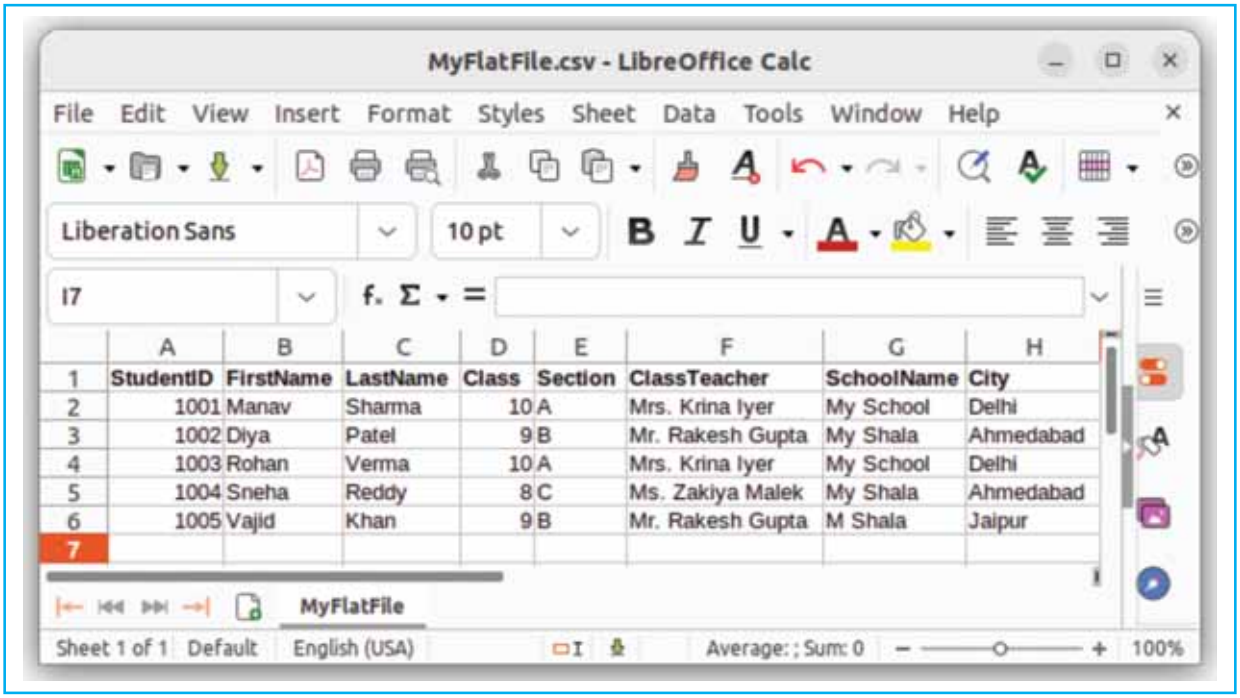


Figure 1.2 : Example of Flat Data Model (CSV file)



	A	B	C	D	E	F	G	H
1	StudentID	FirstName	LastName	Class	Section	ClassTeacher	SchoolName	City
2	1001	Manav	Sharma	10	A	Mrs. Krina Iyer	My School	Delhi
3	1002	Diya	Patel	9	B	Mr. Rakesh Gupta	My Shala	Ahmedabad
4	1003	Rohan	Verma	10	A	Mrs. Krina Iyer	My School	Delhi
5	1004	Sneha	Reddy	8	C	Ms. Zakiya Malek	My Shala	Ahmedabad
6	1005	Vajid	Khan	9	B	Mr. Rakesh Gupta	M Shala	Jaipur
7								

Figure 1.3 : Example of Flat Data Model

The flat database model is very simple and easy to understand. It though cannot be used for complex or large-scale data sets due to several drawbacks. Few of the drawbacks of flat database models are as mentioned:

Data Redundancy: There are very high chances that the same data is repeated multiple times in flat databases. Observe that in figure 1.3 class teacher's name appears multiple times.

Update Anomalies: As the data is repeated multiple times, if the value, for example, My School is to be updated then we need to change data in each and every row that has this data. If the number of rows are too many then it increases the risk of errors.

Limited Modularity: As can be seen in figure 1.3, all data is stored in a single table. This makes it very hard to isolate changes or reuse the data components in different contexts.

Poor Scalability: For small amounts of data say 100 records, flat data model works fine. As the data grows, managing, updating, and querying become inefficient. The lack of structure further leads to performance issues.

No Data Relationships: Representing real-world structures is difficult in a flat data model as it can not define relationships between different entities. For example, in figure 1.3 identifying the relation between students-teachers or classes-subjects is quite difficult.

Data Integrity Issues: Maintaining consistency during operation might become difficult thus creating data integrity issues. For example, while updating a school name a typographical error may go undetected.

Querying Complex Data is Difficult: The database is required to make some decisions in an organization. In flat data models, performing analytics or extracting meaningful relationships becomes very hard due to unavailability of structured links between data.

Relational Data Model

The relational data model solves all the issues that the flat data model faces. The data in the relational data model is arranged in the form of tables (relationships). The tables are made up of rows and columns. A table is used to represent a particular entity, and the relationships between different tables are controlled using keys (fields of tables). The database shown in figure 1.3 thus can be divided into tables that may represent entities like student, teacher, school and others.

The relational data model makes use of Structured Query Language (SQL) (an english like statements) for querying the complex databases. Because of its simplicity, adaptability and solid theoretical basis, the model is used extensively. We will learn how to create tables in the upcoming sections of this chapter.

Need of Database

Businesses today thrive on data, without data it would be impossible for them to come up with new solutions and products. So let us now look at what makes a database so important and what is its need. The following requirements of an organization highlight the need of a database.

Data Organization and Structure

As mentioned earlier databases provide a systematic way or approach to organize the large amounts of data generated by the organization in structured formats. It enables logical grouping of related information thus leading to efficient data storage.

Creation of multiple tables like student, teacher and school thus helps us organize the data that would be required for any school management activities.

Efficient Data Retrieval

Businesses need to make quick decisions based on the data that they have. As data is stored in logical groups it enables fast searching and retrieval of the required information.

Suppose we wanted to find details about a student, it could be easily retrieved by looking into the data stored in the student table. Additional information if required for example, marks of students could be fetched by using data from another related table that stored marks.

Data Integrity and Consistency

No business would like to have incorrect data. Creating a database that can enforce validation rules on data allows the businesses to maintain accurate and consistent data across all their operations.

As data is organized in a logical and structured manner any updates required would be done only at a single place. The effect of this change would then be reflected across the entire dataset. For example, if we changed the name of the school, we would only change it in the school table. The updated value would then be reflected across all tables.

Concurrent Access

Many times a single piece of data needs to be accessed by multiple stakeholders. The database allows multiple users to access and modify data simultaneously without conflicts. In the case of figure 1.3, the class teacher's name may be accessed by the student as well as the school principal.

Access Control

Data needs to be often shared but in a secured and restricted manner. Database provides role-based permissions and user-level security to protect sensitive information. In figure 1.3, if we add a field of total marks in every row then we would need different types of access control.

For example, a teacher would like to know marks of all students of a particular class hence needs to be given access to all the marks of that particular class. A student on the other hand should be able to see only his or her marks.

Data Backup and Recovery

A fallback mechanism for any business is a must. Having a database allows for establishing automated backup solutions to prevent data loss. The backups thus enable point-in-time recovery capabilities.

Assume that we had stored a physical copy of the student results for the past ten years, but the records got destroyed for some reason. Getting this data back would be almost impossible, but if the same records were available in the database, it would be very easy to regenerate the mark sheets.

Reporting and Analytics

Having a database enables a business to generate reports and dashboards for decision-making. It further facilitates complex data analysis and business intelligence operations that can be used to enhance the business capabilities.

Creating and printing mark sheets of hundreds of students would be very easy if the data is part of a database. Also performing analysis like student failure in a subject, best performing teacher and others would be very easy task.

It is possible to think of many more reasons why a database would be needed but for now the above reasons should suffice.

What is a Database Management System?

Database in simple sense is a repository. The repository can be created manually or using computers. A Database Management System (DBMS) is a software application that serves as a bridge between users and databases. It enables efficient creation, management, and manipulation of data using computers.

A DBMS functions as a central system for arranging data in a structured manner. It uses tables, relationships, and schema to maintain data integrity and consistency. The DBMS provides essential functionalities such as data storage, retrieval, modification, and deletion via standard query languages such SQL. It also simultaneously manages multiple user access and maintains security aspects. It manages complicated tasks such as transaction management, backups, and recovery. Additionally, it facilitates scalability by enabling databases to expand in size and handle more users.

Oracle, IBM DB2, Microsoft Access, and Microsoft SQL Server are well-known examples of proprietary DBMS. There are also several outstanding open source database management systems available, such as CouchDB, MariaDB, PostgreSQL, LibreOffice Base and MySQL. Each of these has a unique set of attributes and features that make them ideal for a wide range of uses.

In general, a DBMS is a vital component of most software application since it offers a strong, safe, and effective framework for handling the enormous amounts of data that businesses depend on to support their everyday operations and decision-making procedures.

Components of Database Management System

Most DBMSs provide four basic objects or components known as Tables, Queries, Forms and Reports to work with a required database. These objects become a building block for creating, using and managing any database. Let us have a brief look at each of these objects.

Table

The basic unit for storage in a DBMS is known as a table. A table is a two dimensional structure, it often consists of multiple rows and columns. The column is referred to as a 'Field' and the row referred to as a 'Record'. A table generally belongs to a specific entity like person, place, account, exam or others.

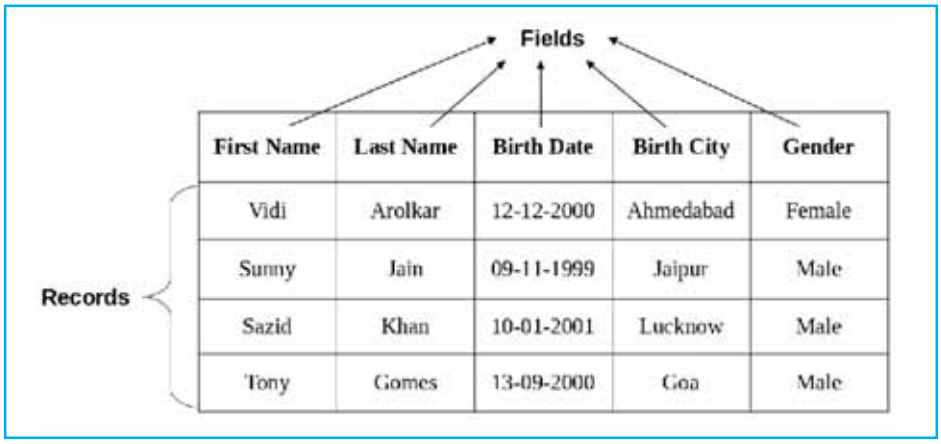


Figure 1.4 : Sample representation of a Table storing data of persons

Let us look at the terms entity, field and record from the perspective of figure 1.4.

Entity: Observe that we have stored the data pertaining to the person in the table. Here a person thus becomes an entity.

Field: It represents the discrete piece of data known as attribute, independently it may not have much of a significance. Here First Name, Last Name, Birth Date, Birth City and Gender are known as fields of the table.

Record: It is a collection of fields that shows details about a single instance of an entity. Observe that combined First Name, Last Name, Birth Date, Birth City and Gender represent a detail of one person.

Looking at the table given in figure 1.4, we can say that we have stored records of four unique persons and each person has five attributes.

Forms

We can insert data in the table as and when needed. Once inserted we may need to edit existing records, delete a record or view the records available within the table. Form provides us a graphical user interface that can be customized to one's personal requirement or liking. The forms may have a specific data entry format, design or layouts that ease the work of the user in case he wants to add, edit or delete a record.

Queries

The data in the table is used to answer the questions asked by the user. A question formed and asked in a database is known as Query. The queries can be created to answer questions like, How many

males are there in the persons table or What is the birth date of Sunny Jain. The query can be used to display a selected set of data contained in different tables within a database.

Reports

The output of a table or a query in a database is often displayed in the form of rows and columns. A user, though, expects more explainable and formatted output for the questions raised so that things can be inferred properly. The reports thus refer to proper presentation of the required information in a properly formatted, organized and readable format.

Macros

One additonal component of Base that is very helpful to users is a Macro. A macro is sequences of commands that automate repetitive tasks within the database. It can perform a series of actions with a single click, such as running a query, opening a report or updating records. We can create user-interface macros to control forms and reports or data macros to automate actions triggered by table events. Additionally macros can be used to implement checks and enforce rules, making data entry more reliable.

Sample Database Creation

Designing a database for any application requires a structured approach. We initially need to understand the requirements of an application, its processes, and the workflow that connects different activities. Based on these we need to identify key entities and their attributes that may be needed.

Let us try and create a structure for a relational data model that can be used to store data pertaining to school management activities. Although the school management includes a wide range of activities, such as managing students and teachers, examination management, payroll, and many more. We will look into only a small part of it where we will look into details of the student, teacher, subject, class and grade.

We have now decided about whom data will be stored? All these terms pertaining to whom data will be stored are known as entities. We will design a separate table for each entity. Thus in this case we will design five tables namely Student, Teacher, Subject, Class and Grade.

The next step is to decide what attributes each of these entities will have and what type of values are stored in them. These attributes will be represented as fields within the table. The sample schema of all these entities and their attributes is as shown in table 1.1.

Entity	Key Attributes
Student	StudentID, FirstName, MiddleName, LastName, BirthDate, Gender, Address, ClassID
Teacher	TeacherID, FirstName, MiddleName, LastName, BirthDate, Gender, Address, SubjectID
Subject	SubjectID, SubjectName, ClassID
Class	ClassID, ClassName, TeacherID
Grade	GradeID, StudentID, SubjectID, GradeDate, Marks

Table 1.1 : Sample Schema for School Database

This schema can be expanded to include any additional features as per the need of the school. For example we may add the details such as payroll, training, or recruitment.

Let us try and understand the details of attributes that will be stored in each of the tables as shown in table 1.2, 1.3, 1.4, 1.5 and 1.6.

STUDENT		
Attributes	**Description**	**Type of value to be stored**
StudentID	Used to store the ID of the student.	Can be a text or numeric value depending on organization policy.
FirstName	Used to store the first name of the student.	A text value.
MiddleName	Used to store the middle name of the student.	A text value
LastName	Used to store the last name of the student.	A text value.
BirthDate	Used to store the birth date of the student.	A date value.
Gender	Used to store the gender of the student.	A text value.
Address	Used to store the address of the student.	An alpha numeric value (May contain text, number as well as special characters).
ClassID	Used to store the ID of the class in which the student studies.	Will depend on the type of data stored in the Class table.

Table 1.2 : Details of Student Table

TEACHER		
Attributes	**Description**	**Type of value to be stored**
TeacherID	Used to store the ID of the teacher.	Can be a text or numeric value depending on organization policy.
FirstName	Used to store the first name of the teacher.	A text value.
MiddleName	Used to store the middle name of the teacher.	A text value
LastName	Used to store the last name of the teacher.	A text value.
BirthDate	Used to store the birth date of the teacher.	A date value.
Gender	Used to store the gender of the teacher.	A text value.
Address	Used to store the address of the teacher.	An alpha numeric value (May contain text, number as well as special characters).
SubjectID	Used to store the ID of the subject that the teacher teaches.	Will depend on the type of data stored in the Subject table.

Table 1.3 : Details of Teacher Table

SUBJECT		
Attributes	**Description**	**Type of value to be stored**
SubjectID	Used to store the ID of the subject.	Can be a text or numeric value depending on organization policy.
SubjectName	Used to store the name of the subject.	A text value.
ClassID	Used to store the ID of the class in which the subject is taught.	Will depend on the type of data stored in the Class table.

Table 1.4 : Details of Subject Table

CLASS		
Attributes	**Description**	**Type of value to be stored**
ClassID	Used to store the ID of the class.	Can be a text or numeric value depending on organization policy.
ClassName	Used to store the name of the class.	A text value.
TeacherID	Used to store the ID of the teacher who manages the class.	Will depend on the type of data stored in the teacher table.

Table 1.5 : Details of Class Table

GRADE		
Attributes	**Description**	**Type of value to be stored**
GradeID	Used to store the ID of records within the Grade table.	An integer number.
StudentID	Used to store the ID of the student who has been given the grade.	Will depend on the type of data stored in the student table.
SubjectID	Used to store the ID of the subject for which the student is graded.	Will depend on the type of data stored in the subject table.
GradeDate	Used to store the date when the student is given the grade.	A date value.
Marks	Used to store the marks.	A numeric value.

Table 1.6 : Details of Grade Table

We will revisit these tables in the next chapter when we create a database using LibreOffice Base. Let us talk about the concept of keys in a relational data model. As mentioned earlier, data in relational data model is arranged into tables (relationships) made up of rows and columns. The keys help us maintain these relationships properly. Four keys namely; primary, foreign, candidate and alternate play an important role.

Primary Key

A field that uniquely identifies a row in a table is called a primary key. The primary key can be made up of one or more columns with distinct values. The primary key cannot have the same value for multiple rows in the table, also the data value for that field cannot be left empty. For example, every student in the Student table has a distinct StudentID that can be considered as a primary key. If, in a table we use more than one field to identify a record, then this combined set of fields is known as a composite key.

Foreign Key

The foreign key is the field in one table that can be used to uniquely identify records in another table, either by itself or in combination with other fields. This foreign key facilitates the establishment of a relationship between two tables. For example, the ClassID field in the Student table acts as a foreign key, it tells us in which class the student studies. It also identifies the unique records in the class table. Thus we can say that the ClassID key relates the Student and Class tables.

Candidate Key

The candidate keys for a table are all the field values that are eligible to be the primary key. There must be no duplicate values in such fields, and they cannot be left empty. Therefore, in the Class table, both the ClassID and ClassName are potential candidate keys.

Alternate Key

One or two of the candidate keys are chosen as a primary key for a table. The candidate keys that are left out are known as alternate keys. Therefore, in the Class table the ClassName is the alternate key if the ClassID is used as the primary key.

Data and Data types

We have already seen in table 1.2 that the nature of data stored in different fields can vary. For example, in our data set the fields FirstName, LastName and Gender have text data while the field BirthDate is a date. We may also use alphanumeric data in case of a field named Address or a numeric data in a field named Age or a decimal number in a field named Percentage.

A data type thus refers to the type of data that will be stored in a specific field. The data type also decides the amount of memory that would be allocated to a particular field. It is possible that the memory size may vary within an individual data type also. Most DBMS support some common data types like text, numeric, currency, date/time, boolean and binary. In this section, we will discuss the data types that are supported by LibreOffice Base.

Text Data Type

The text data type allows us to store a combination of alphabets, numbers or special characters as a data value in a field. We cannot perform any arithmetic calculations on such data. Few examples of the fields that can use text data type are Aadhaar Card Number, First Name, Delivery Address and Email. There are four different text data types, and they differ mostly in how they consume space. Table 1.7 shows the list of different text data types that can be used in LibreOffice Base.

Type	Field Data Type in Base	Storage in memory
Text	VARCHAR	Variable
Text	VARCHAR_IGNORECASE	Variable
Text (fix)	CHAR	Fixed
Memo	LONGVARCHAR	Variable

Table 1.7 : Text Data Types

VARCHAR: The term here represents variable characters. It is possible to define the maximum number of characters that can be entered in a field when using VARCHAR. For example, if a field has data type VARCHAR(30) then we can enter a maximum of 30 characters as data in it. In case the user enters data which is less than 30 characters, only the needed space will be automatically allocated by the DBMS.

VARCHAR_IGNORECASE: It is similar to the VARCHAR, but is a case insensitive version of VARCHAR. The ignore case here plays a critical role at the time of searching. Assume that the user has stored some data "AHMEDABAD" in the field named city, now at the time of search if the user looks for "Ahmedabad" he will still get an output.

CHAR: This data type refers to a fixed size text field. The size of the field is set at the time of definition similar to the VARCHAR data type. If the text entered in the field does not occupy all the characters, the remaining characters are padded with spaces. This data type is best used when we expect a predefined number of characters like in a PAN or a credit card number.

LONGVARCHAR: This data type is designed to store very large blocks of text. It stores up to the maximum length of characters indicated by the user. It generally is used to store data having more than 255 characters. A good example of data for such a field is an article whose text may be up to 64,000 characters.

Numeric Data Type

This data type is used to store data of numerical form. The numeric data can be in the form of integer numbers or decimal numbers (floating point numbers). Both integer and decimal numbers can be signed or unsigned. It is possible to perform arithmetic operations on such data. Few examples of the fields that can use numeric data type are Marks, Salary, Interest and TotalPayment. LibreOffice Base has four integer data type which generally vary in size and four floating-point data types which vary in level of accuracy. Table 1.8 shows the list of different numeric data types along with the number of bits/bytes it uses and its range.

Type	Field Data Type in Base	Range of values	Bytes required in memory
Tiny Integer	TINYINT	0 to 255 if unsigned –128 to +127 if signed	1 Byte
Small Integer	SMALLINT	0 to 65535 if unsigned –32768 to +32767 if signed	2 Bytes
Integer	INTEGER	0 to 4294967295 –2147483648 to +2147483647	4 Bytes
BigInt	BIGINT	0 to 18446744073709551615	8 Bytes
Number	NUMERIC	Unlimited	Variable
Decimal	DECIMAL	Unlimited	Variable
Float	FLOAT	$5 \times 10^{(-234)}$ to $1.79 \times 10^{(308)}$	4 Bytes
Real	REAL	Not exact	4 Bytes
Double	DOUBLE	Adjustable with 15 decimal places maximum	8 Bytes

Table 1.8 : Numeric Data Types

TINYINT: It is the smallest of the integer data types. Its size is one byte hence it occupies 8 bits in memory.

SMALLINT: It is double the size of TINYINT. Its size is 2 bytes hence it occupies 16 bits in memory.

INTEGER or INT: The most commonly used numeric data type is INTEGER. Its size is 4 bytes hence it occupies 32 bits in memory.

BIGINT: Though available most of the simple programs do not require it, hence its rarely used. Its size is 8 bytes hence it occupies 64 bits in memory.

Floating-point numbers are numbers with decimals, or real numbers. They are made of a whole part and a partial part separated by a decimal point.

DECIMAL and NUMERIC: These data types have an unlimited range. While defining them we need to specify the total numbers of digits allowed along with the number of digits that will fall after the decimal point. For example, a definition DECIMAL(10, 2) for a field would mean that the field has a maximum of 10 places and two digits after the decimal point. The accuracy for DECIMAL and NUMERIC is nearly perfect.

DOUBLE or REAL: These data types have a limited range and can have a maximum of 15 decimal places. The accuracy of this data type is not so good.

Currency Data Type

The currency data type allows us to store numerical values that indicate the monetary values. We can store data using currencies of various countries. For example, ₹ 1250.50, $1500 or £ 700. This data type ensures that calculations involving currency are accurate and that the display of monetary values is formatted correctly.

Date/Time Data Type

The Date/Time data types are used to store information like year, month, day, hour, minute, second and fraction of a second. Few examples of the fields that can use date/time data type are Date of Joining, Birth Date, In Time and Out Time. LibreOffice Base has three types under it, they differ in the content they store. Table 1.9 shows the list of different date/time data types.

Type	Field Type in Base	Range of Values	Storage in memory
Date	DATE	–	4 Bytes
Time	TIME	–	4 Bytes
Date/Time	TIMESTAMP	Adjustable (0.5 – 5 with milliseconds)	8 Bytes

Table 1.9 : Date/Time Data Types

DATE: It stores the system date. The format for date entry is YYYY-MM-DD, i.e. 2025-05-30.

TIME: It stores a clock time. The default format for time is 24 hour clock, as HH:MM:SS, i.e. 15:30:36 for 3:30:36 PM.

TIMESTAMP or DATETIME: It is a combination of both the date and the time. The default format of data here is YYYY-MM-DD HH:MM:SS, i.e. 2025-05-30 15:30:36. This data type is used in the audit fields like Time of Transaction.

Boolean Data Type

The boolean data type allows us to store only two values, True or False. These two values can also be represented in multiple formats like Yes/No and On/Off.

Binary Data Type

The Binary data types allow us to store information like digitized images, videos, sounds or may be files. All these entities are stored as a long string of zeros and ones. Few examples of the fields that can use binary data type are Profile Picture and Voice Sample. Table 1.10 shows the list of different binary data types.

Type	Field Type in Base	Range of Values	Storage in memory
Binary field (fix)	BINARY	Similar to Integer	Fixed
Binary field	VARBINARY	Similar to Integer	Variable
Image	LONGVARBINARY	Similar to Integer	Variable used for very large images

Table 1.10 : Binary Data Types

Summary

In this chapter, we learned the difference between data and information. Data consists of raw facts, while information is the organized form of data. We further looked at data model in a database, it shows how data is stored and retrieved. Database management systems use various data models, including flat, hierarchical, and relational models. The relational model connects tables within a database. We also learned the concepts of database and DBMS, a database is an organized collection of data, and DBMSs help add, update, and access this data. We came to know that real world objects are called entities, and their specific details are known as attributes. The tables are representation of entities and contain records that are arranged in form of rows and columns, with the smallest unit being a field that represents an attribute. Data values stored in the fields can be numbers, letters or special characters. A record thus is a complete set of data values for a specific entity. A primary key uniquely identifies a row, and foreign keys connect different tables. Candidate keys are potential main keys for a table. This chapter has laid down a strong foundation for learning further concepts of database in the coming chapters.

EXERCISE

1. Explain the terms Data and Information with proper examples.

2. Differentiate between Database and Database Management systems.

3. Define the terms table, record and field.

4. Explain the need of a database.

5. Write a note on general components of a database.

6. What is Data type? List the data types available in Base.

7. What is the difference between CHAR and VARCHAR?

8. What is the use of currency data type?

9. When should one use boolean data type?

10. Identify any five attributes of an entity STUDENT.

11. State whether true or false.

(1) Database is a software that allows us to manipulate data.

(2) If we store data about a bus then it can be considered as an entity.

(3) Forms allow us to delete data from a table.

(4) It is possible to store a numeric value in Text data type.

(5) An audio clip can be stored in a Time data type.

12. Fill-in the blanks.

(1) Database is a ……... of related data items.

(2) The attributes of an entity are referred to as ……... in a table.

(3) A table consists of a set of ……... that give details on an entity.

(4) The BIGINT uses ……... bytes of memory storage.

(5) Digitized images can be stored in ……... data type.

13. Multi-choice questions. Choose the most correct answer.

(1) Which of the following is the full form of DBMS?

(a) Data Block Management System

(b) Database Management System

(c) Data Byte Management System

(d) Data Bit Management System

(2) Which of the following terms refers to processed data?

(a) Information (b) Raw (c) Facts (d) Field

(3) Which of the following is not a data model?

(a) Flat (b) Relational

(c) Rotational (d) Both (a) and (b)

(4) Which of the following best describes a StudentName in Database?

(a) Relationship (b) Attribute (c) Entity (d) Data

(5) Which of the following is not a database?

(a) MySQL (b) LibreOffice Base

(c) LibreOffice Impress (d) SQL Server

(6) Which of the following is not a general component of the database?

(a) Chart (b) Table (c) Queries (d) Form

(7) Which of the following data types is used to store an image in a Base database?

(a) Text (b) Binary (c) Boolean (d) Byte

(8) In which of the following forms, can a data not be represented?

(a) Text (b) Numeric

(c) Alphanumeric (d) Picture

(9) Which of the following is a feature that allows us to enter data in a table in an easy and user friendly manner.

(a) Report (b) Query (c) Form (d) Field

(10) Which of the following cannot be used to store numbers?

(a) TINYINT (b) DOUBLE (c) DATE (d) CHAR

Laboratory Exercise

1. Given the following entities, identify their attributes and decide what data types will be suitable to store the data.

   a. Book

   b. Customer

   c. Department

   d. Employee

   e. Flight

   f. Magazine

   g. Movie

   h. Product

   i. Salesman

   j. Train

   k. Vehicle

2 Introduction to LibreOffice Base

Introduction

LibreOffice Base is a free and open source desktop database tool. It comes as a part of LibreOffice Suite. It supports major database engines like MySQL, MS Access and PostgreSQL. Base includes the HSQL relational database engine as default. Users can create tables, queries, forms and reports with the help of wizards and pre-defined templates. Additionally, Base integrates well with other LibreOffice applications, providing data for mail merges in Writer and creating linked data in Calc for analysis. In this chapter, we will learn how to use LibreOffice Base and create a database and tables.

Opening Base

LibreOffice Base is not installed by default in Ubuntu Linux due to its dependencies on Java. So before we want to use it we need to install it. One of the simplest ways to install it on Ubuntu Linux is to run the command "*sudo apt install libreoffice-base*" in the terminal window. Alternatively you can download it from the official LibreOffice website. Once installed we can start using it.

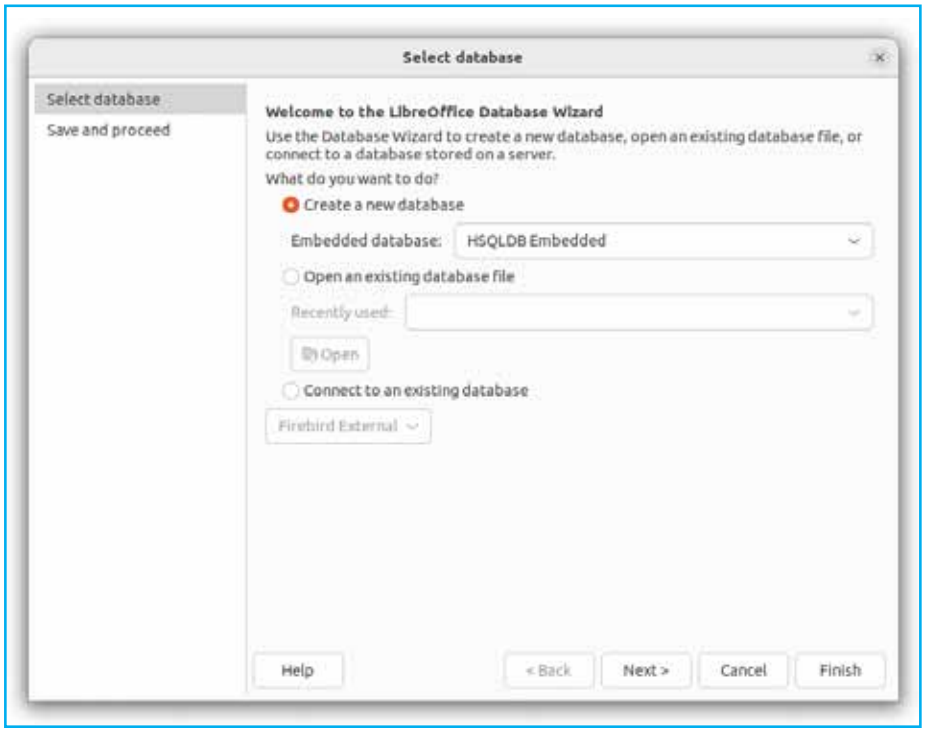


Figure 2.1 : Initial Screen of Database Wizard

You can open LibreOffice Base in multiple ways. The simplest is to pin the LibreOffice Base application to the Launcher and click on it. A *Database Wizard* dialog box as shown in figure 2.1 will pop in front of you.

A wizard is a guided, step-by-step graphical interface tool that simplifies complex tasks by prompting users for information and automates the process of performing the task.

Observe that the screen in figure 2.1 provides us with three options that we can choose from; *Create a new database*, *Open an existing database file* and *Connect to an existing database*. We will need to choose one of these options. As can be easily understood, we can create a new database using option one, while other options allow us to work with a database that has been previously created and saved in our computer.

Note: The contents visible on the screen in figure 2.1 may differ based on the type of installation.

Let us try and create a database for the tables that we had prepared in chapter 1 for the school management system. To create this new database select the *Create a new database* option and click on the *Next* button, a screen shown in figure 2.2 will appear.

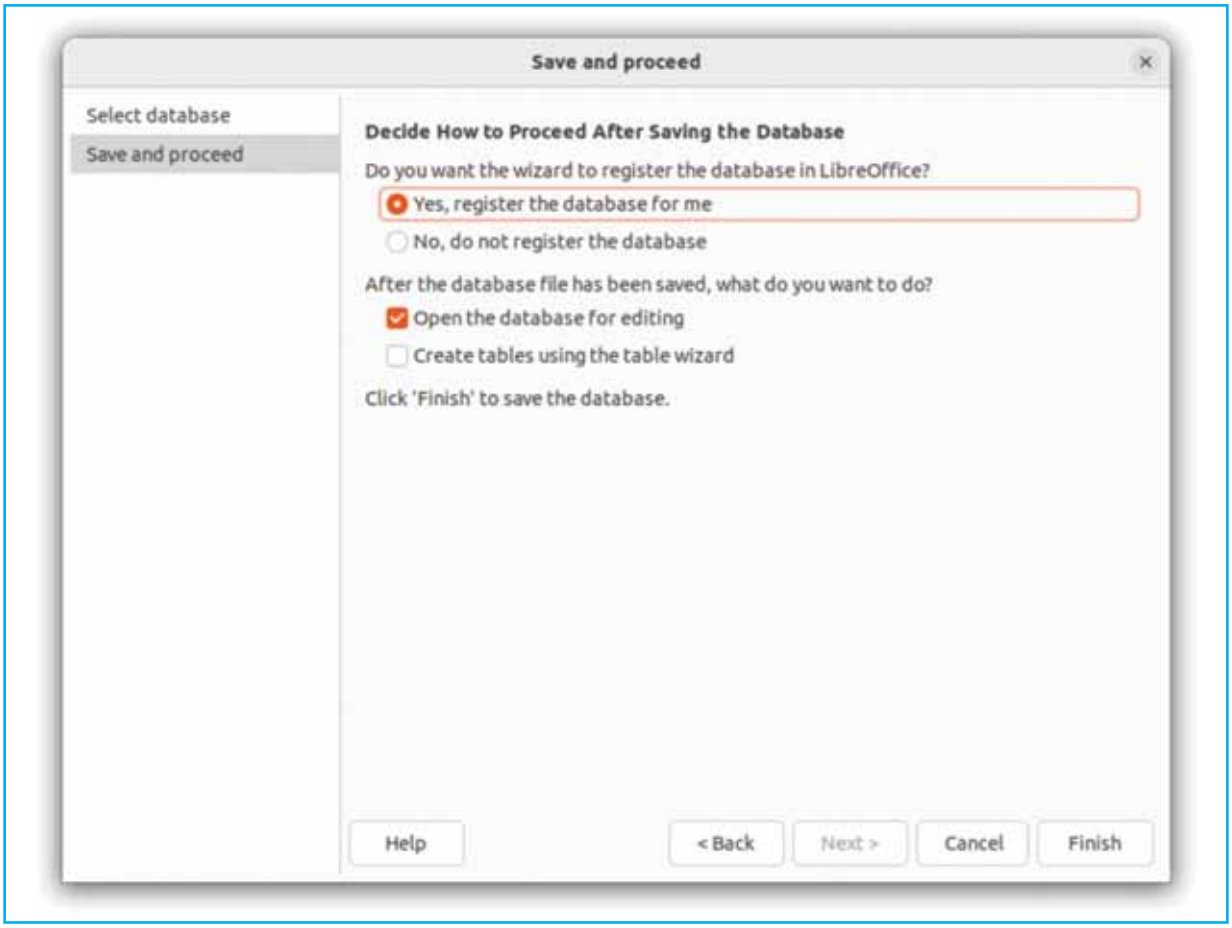


Figure 2.2 : Screen 2 of the Database Wizard

Observe that we need to perform two activities here. First, decide whether we want to register our database in LibreOffice or not? By default, the option *Yes, register the database for me* is selected. If registered the databases will be accessible from other LibreOffice suite applications like Calc and Writer. Second we need to

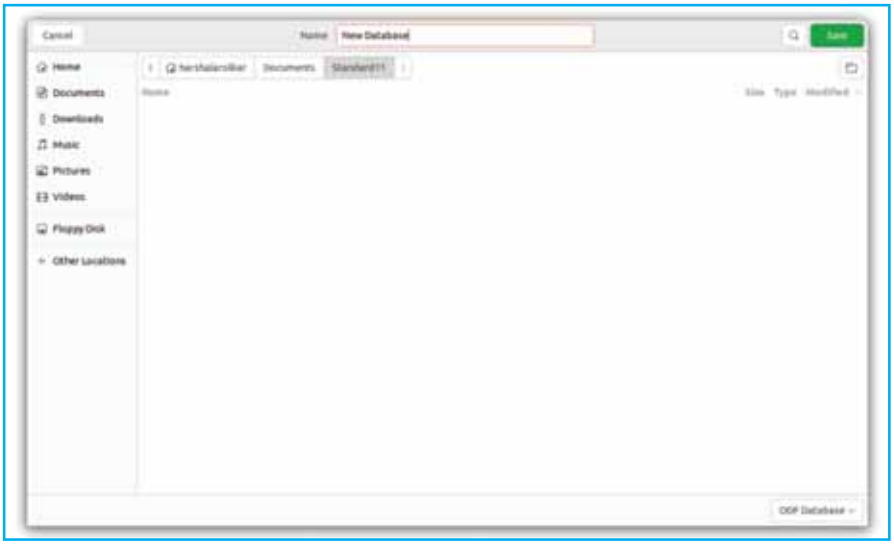


Figure 2.3 : Screen to Save the Database

decide how we want to start working with our database once it is saved. As can be seen, by default the *Open the database for editing* option is checked. We can also select the option *Create tables using the table wizard* if we want to perform both these actions. For now we will first only create the database, click on the *Finish* button. This will open a screen as shown in figure 2.3.

First select an appropriate folder to store the database file. Observe that we are trying to save our database in a folder named Standard11 within the Documents folder of the machine. In the textbox with label *Name* type ScMS instead of *New Database*. Note that by default the *ODF* (Open Document Format) *Database* is selected. Thus LibreOffice Base automatically assigns *.odb* extension to the database file that we want to create. Lastly click on the *Save* button. The database will now be saved and ready for further use and you will be presented with the screen as shown in figure 2.4.

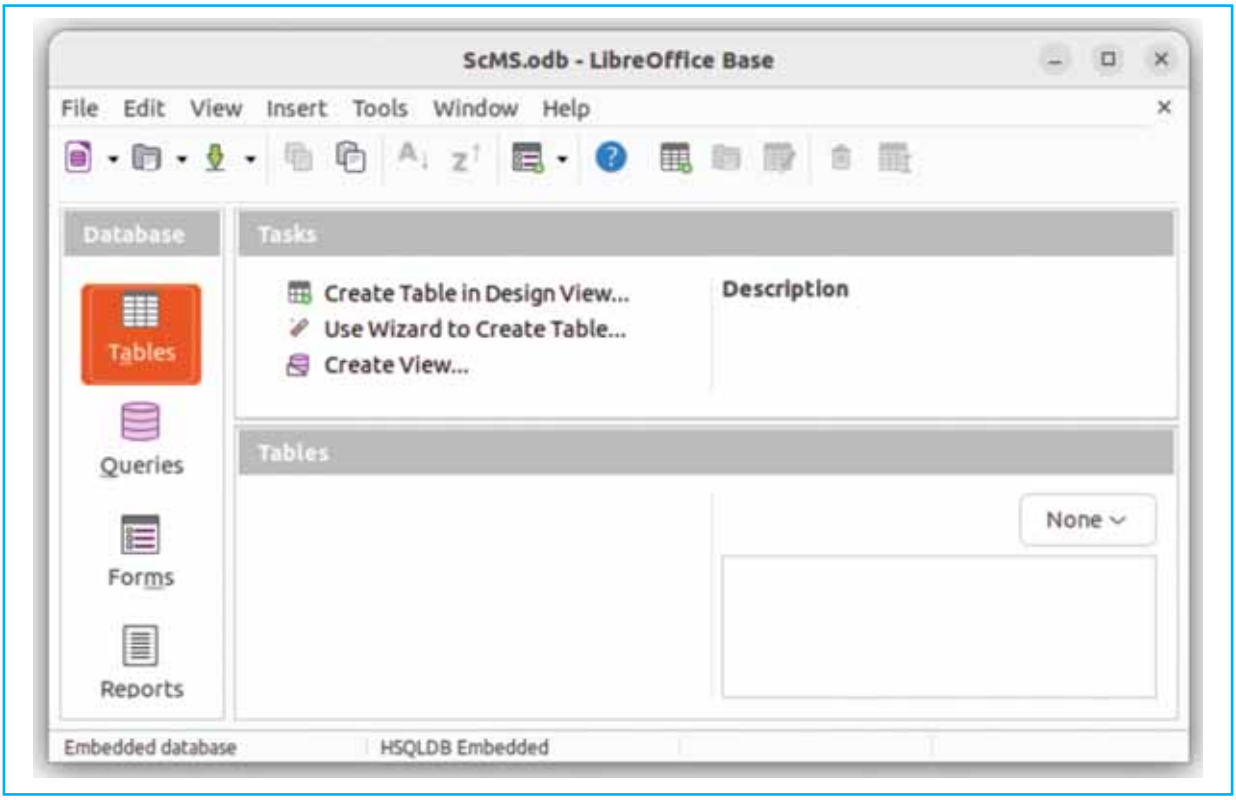


Figure 2.4 : ScMS Database Window

In the left pane, we can see different database objects like *Tables*, *Queries*, *Forms* and *Reports*. By Default, the object Tables would be visible as selected. We are now ready to work with different objects of the ScMS database.

Creating Tables using Wizard

A database without any tables does not make sense, so the next step is to create tables. We will use wizards to create tables. Click on the second option *Use Wizard to Create Table...* under the Tasks pane shown in figure 2.4. A Table Wizard dialog box as shown in figure 2.5 will appear.

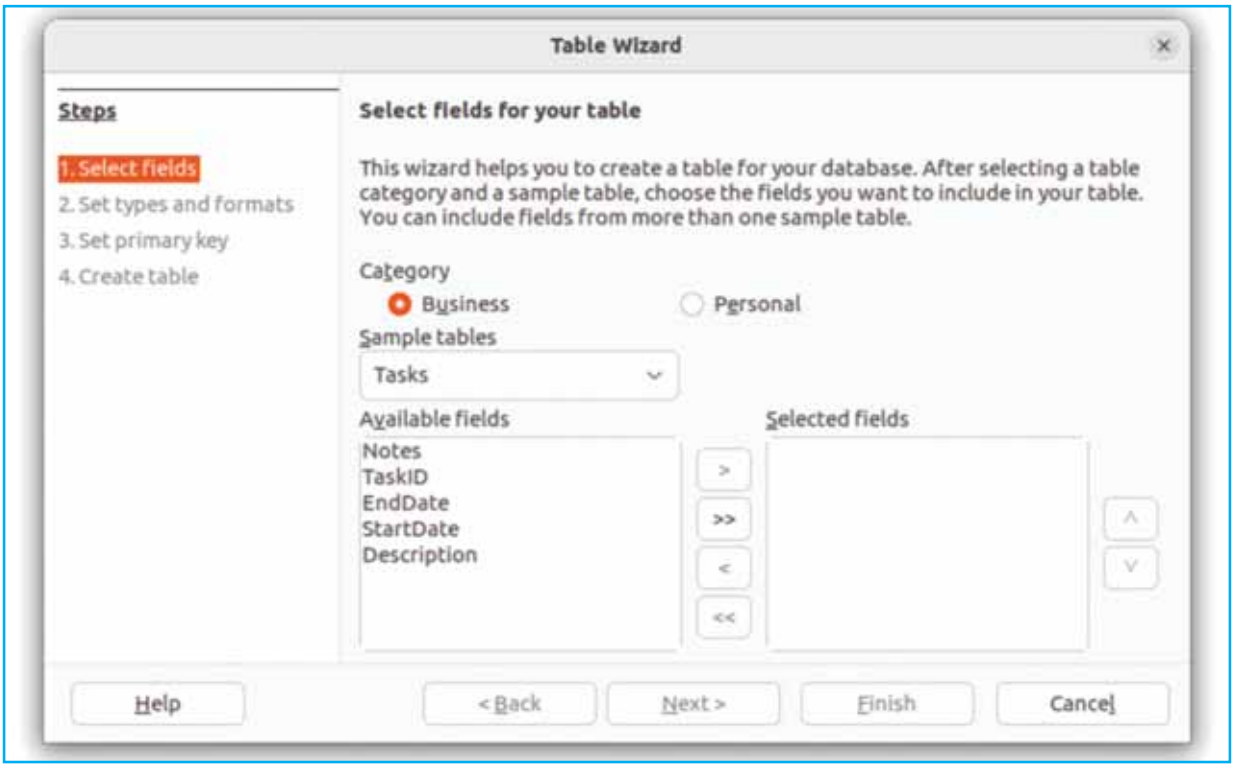


Figure 2.5 : Table Wizard Dialog Box

The *Table Wizard* provides templates of two categories of tables namely, Business and Personal. The business category tables have templates for entities like Tasks, Assets, Events, Orders and others while the personal category tables have templates for entities like Plants, Authors, Libraries, Recipes and others.

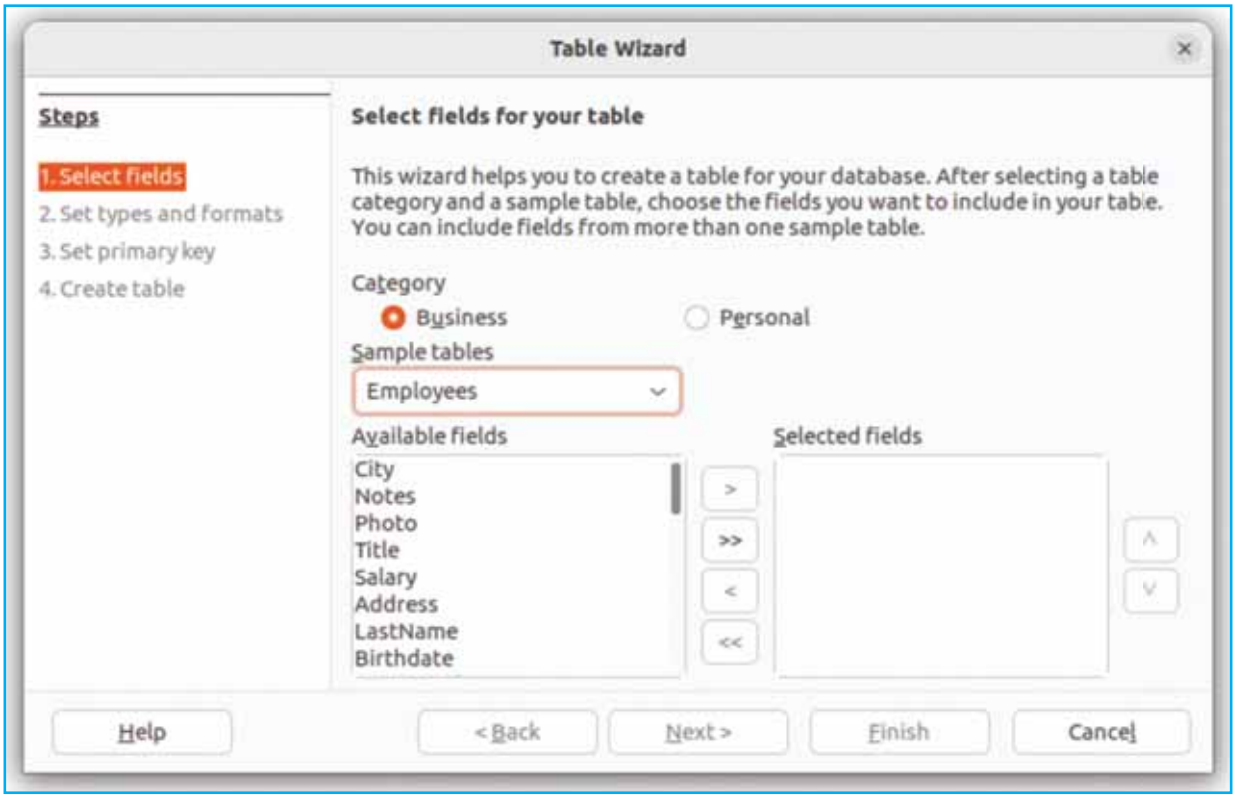


Figure 2.6 : Table Wizard Dialog Box for Creating Employees Table

We will select the Business option under Category label of figure 2.5. Click on the drop down list visible under the label *Sample tables*. For understanding purposes let us create the Employee table. Observe that the drop down list contains a table named *Employees*, select it and we will be presented a list of fields as shown in figure 2.6.

To make all the fields part of our table click on the >> button icon. Alternatively we can choose a single field or set of fields by holding the Ctrl key and selecting desired fields one by one. Let us select fields EmployeeID, FirstName, MiddleName, LastName, Birthdate, Address and DepartmentID. Once we have selected the desired field click on the > button icon to make it part of our table. All the selected fields will now be visible under the *Selected fields* list box as shown in figure 2.7.

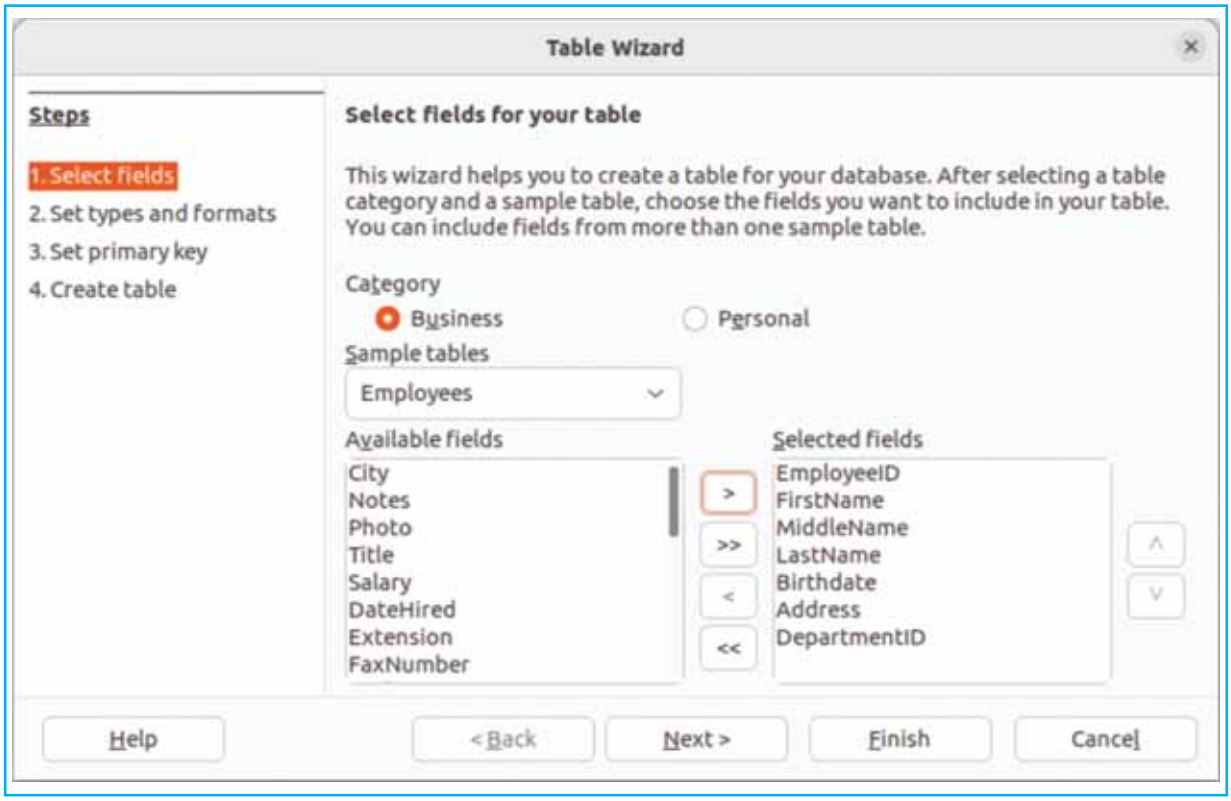


Figure 2.7 : Table Wizard Dialog Box for Selecting Fields of the Employees Table

We can rearrange the order of the fields as per our requirement using the ⋀ and ⋁ button icons. Once all fields have been arranged as per our needs click on the *Finish* button. The Employees table will pop up in a new window known as *Table Data View* as shown in figure 2.8. The *Table Data View* allows users to enter records within the table.

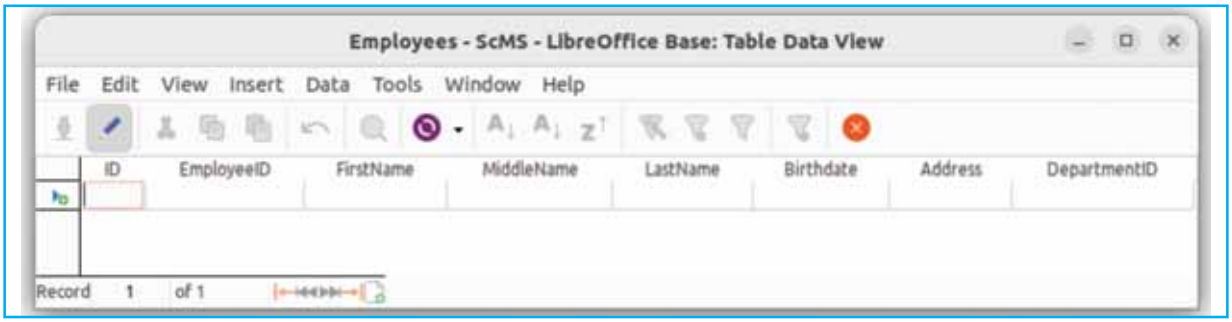


Figure 2.8 : Table Data View of Employees Table

We can start entering data in the table and once data entry is complete we can close the data view. We will be led back to the ScMS database window when we close it.

Creating Tables using Design View

The tables created using wizards many times need to be modified as per the needs of the user. To create a customized table, LibreOffice Base provides an option of Design View. When a table is created or opened in a Design View a user can create, edit, update or delete the fields of the table as per their needs. The table design view allows us to create a customized field name, select the type of data that can be stored in the field, if needed add a description of the field to assist users and also allows us to control and validate the data that can be stored in the field.

Let us now create the Student table having attributes StudentID, FirstName, MiddleName, LastName, BirthDate, Gender, Address and ClassID that we had designed in chapter 1. To create this table click on the *Create Table in Design View*... option of figure 2.4. This will open a blank table design view as shown in figure 2.9.

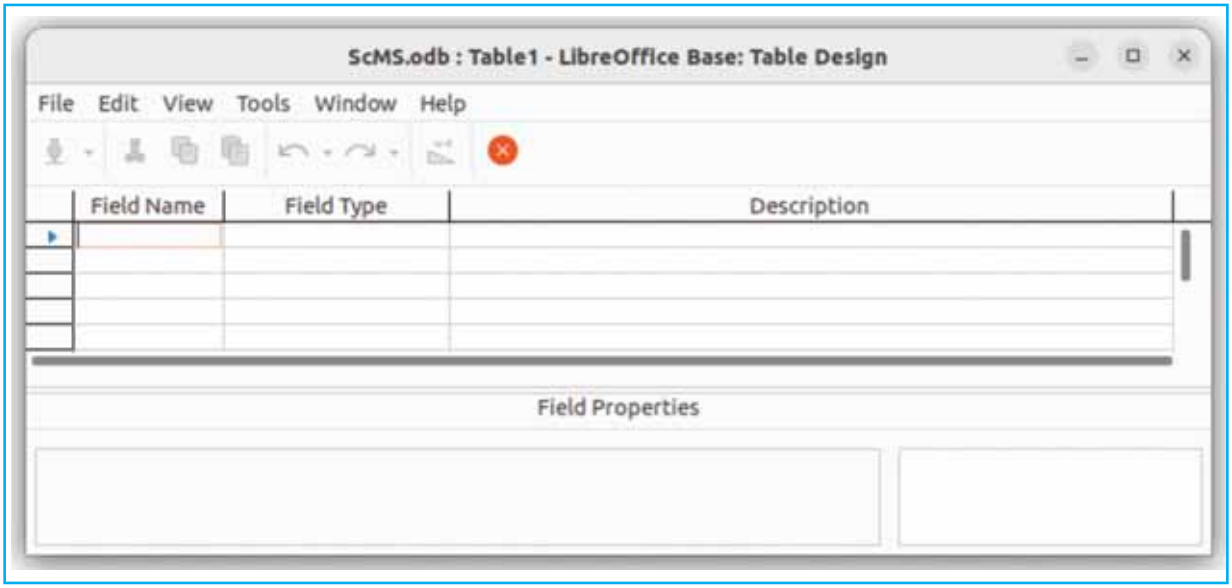


Figure 2.9 : Blank Table Design View

As can be seen, we are provided with a data grid with three columns, *Field Name*, *Field Type* and *Description*. We also have a *Field Properties* pane visible below the table grid. We can now start creating the table structure by typing field names under the *Field Name* column and selecting the required data type under the *Field Type* column. At this juncture, we will not add a description of the fields. Enter all the field names for the Student table as mentioned earlier and select their respective data types. After completion of the process the screen will look something similar to the one shown in figure 2.10.

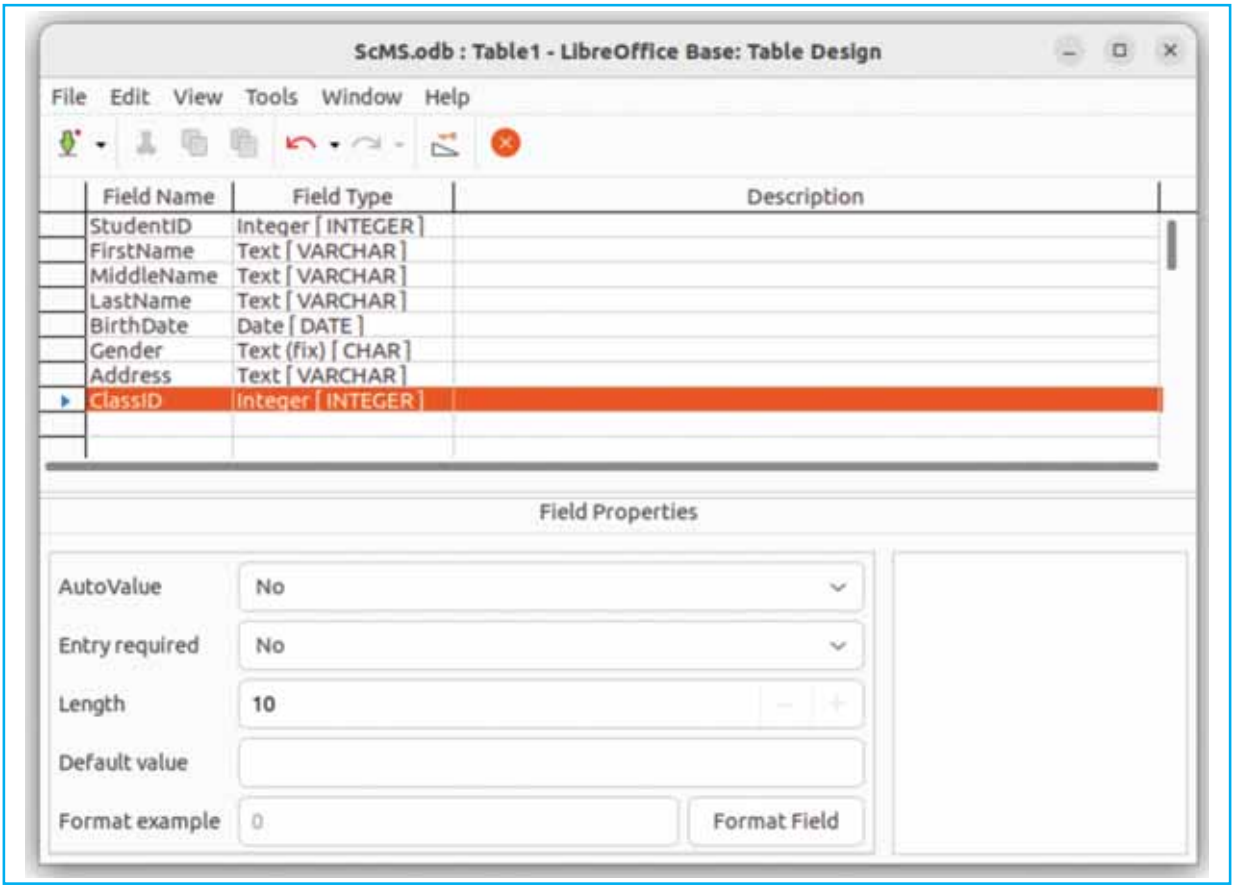


Figure 2.10 : Table Design View with Field Names

Save the table by clicking on the green down arrow with red dot (*Save* button), this will open a *Save as* dialog box as shown in figure 2.11.

Type the name of the table (Student) and click on the *OK* button, a pop up as shown in figure 2.12 will appear.

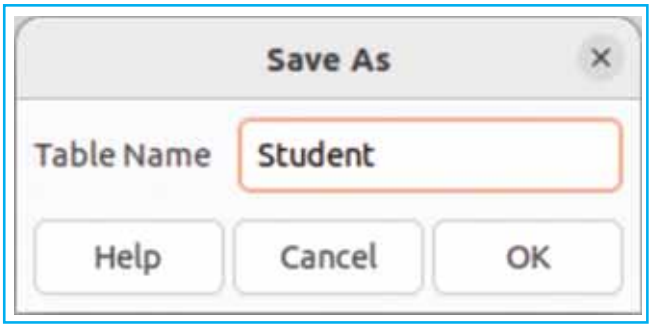


Figure 2.11 : Saving the table

Figure 2.12 : Alert for Primary Key

The screen shows an alert message indicating that the user needs to select a unique primary key. As we learned in Chapter 1, a primary key is a field that uniquely identifies a row in a table. For now click on *No* button, the table is now saved and we can perform required operations on it. Close the table, observe that the ScMS database window will now list two tables Employees and Student.

Modifying the Structure of Table

Till now we have learnt two ways to create tables in LibreOffice Base. At times we need to modify the structure of the tables that we have created. We may need to add a new field, update a name of an existing field, change a data type or may want to add or remove field properties. It is recommended that all the said operations should be performed before entering data into the table. Let us try to modify the Employees table such that it can be used as the Teacher table that we had designed in Chapter 1.

Figure 2.13 : Rename to Dialog Box

The Teacher table has attributes TeacherID, FirstName, MiddleName, LastName, BirthDate, Gender, Address and SubjectID. The Employees table also has a somewhat similar structure. To change the structure of the Employees table, go to the ScMS window, select the Employees table and right click on it. From the drop down menu select the *Rename*... option, we will get a *Rename to* dialog box as shown in figure 2.13.

Change the name from Employees to Teacher and click on the *OK* button. The table will now be renamed and we will be back in the ScMS window.

Once again select the Teacher table and right click on it, from the drop down menu select the *Edit*... option, this will open the table in the design view. Observe that we are able to see a field named ID with a blue triangle icon and a key preceding it. The blue triangle icon is known as the current row pointer that indicates which row is the current row. Select the current row by left clicking on the key. The row will now be highlighted. The ID field was not part of the table design that we had created, so to delete this field right click on the current row icon and a screen similar to the one shown in figure 2.14 will appear.

The menu shown in figure 2.14 can be used to perform different operations. To remove the field, we can use the *Cut* or *Delete* option of the menu. A new row can be inserted using the *Insert Rows* option. We can copy the field name and its data type using the *Copy* option. Lastly we can make the selected field a primary key by selecting the *Primary Key* option. Observe that the Primary Key option in the screen is preceded by a ✓ sign. This indicates that the field we are trying to work on is at present the primary key of the table.

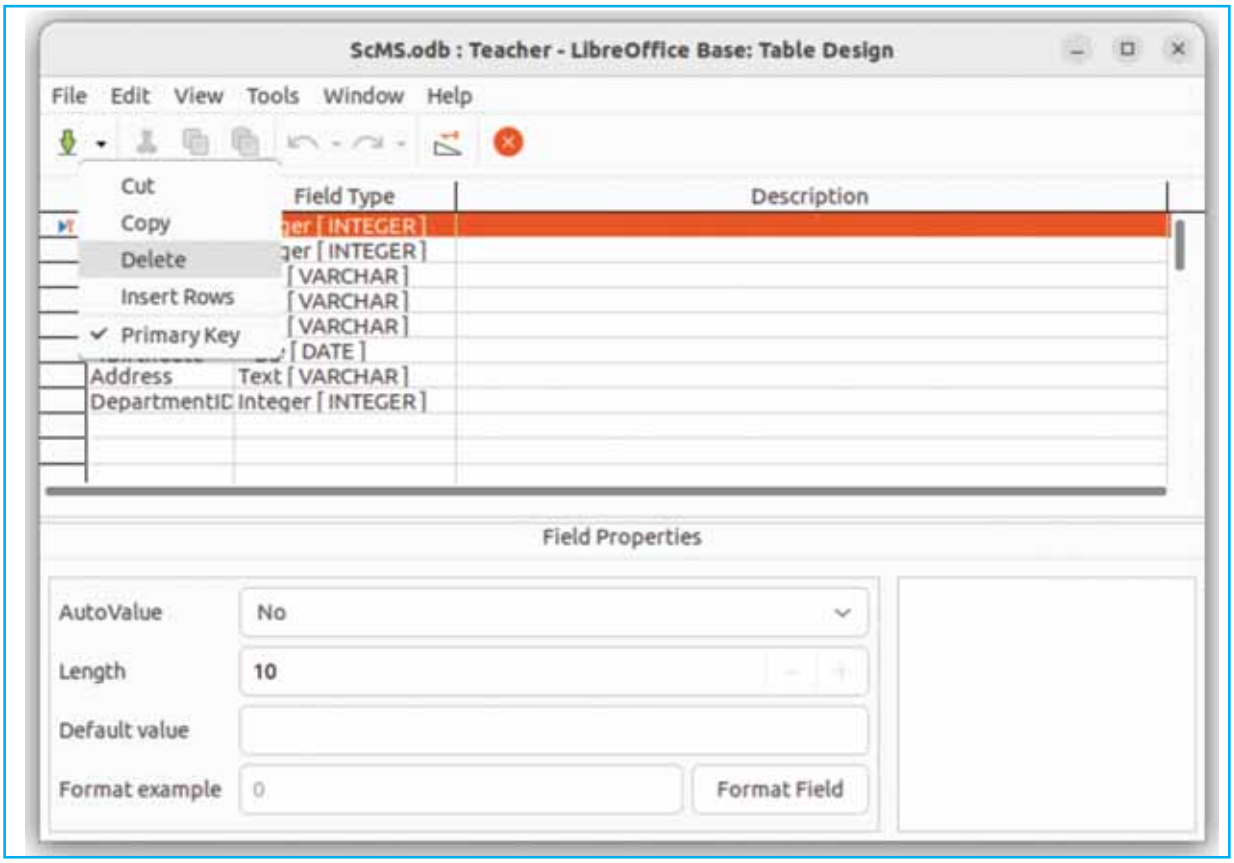


Figure 2.14 : Editing the Table Structure

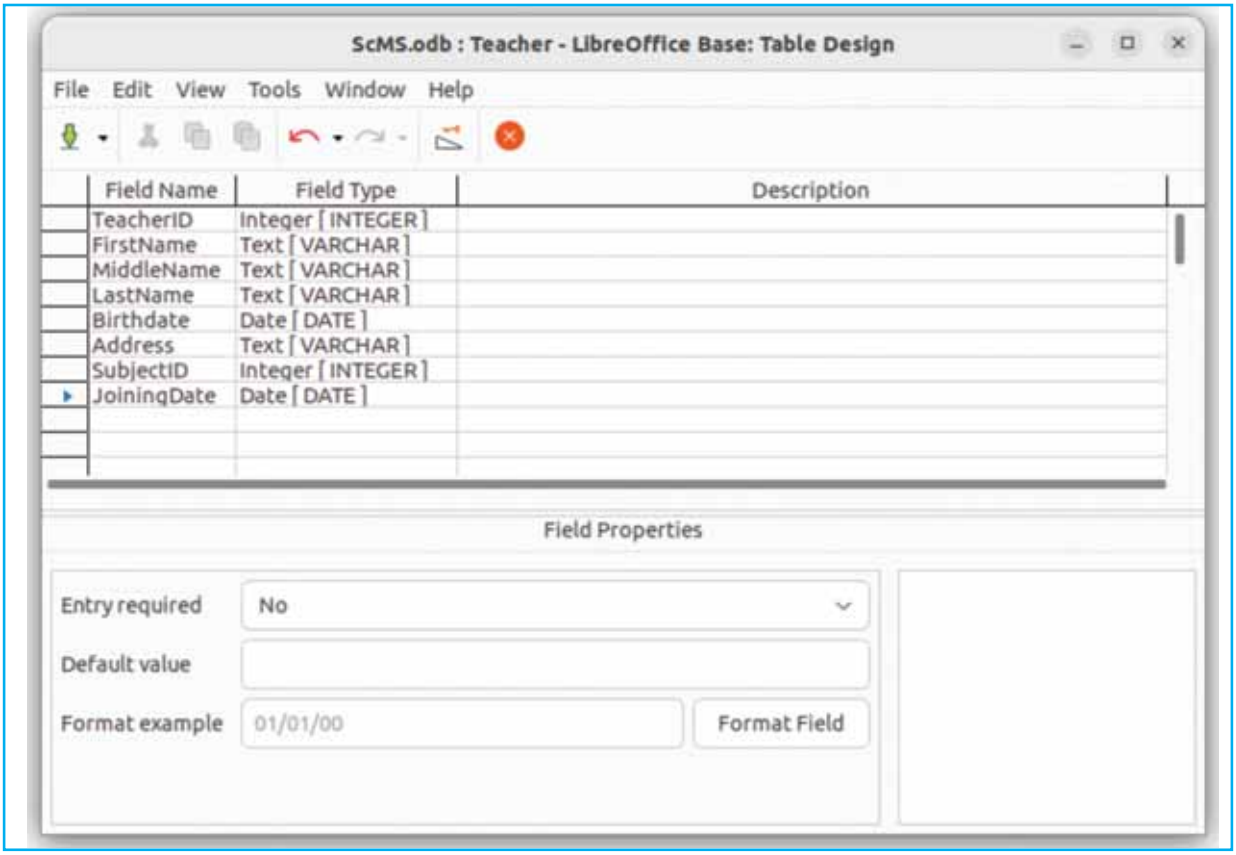


Figure 2.15 : Update Table Structure

First select the *Delete* option from the drop down menu and the field ID will be deleted and the EmployeeID field will be shown as the current record. To update the field name simply double click on the name and type the new name. Thus rename the field EmployeeID to TeacherID and DepartmentID to SubjectID. We will also add one new field JoiningDate here to show the change in the table structure. Save the changes, if we get a warning for deleting the primary key, we will ignore it for now. The screen will now look like the one shown in figure 2.15.

Create the remaining tables that we had designed in Chapter 1 using the design view so that we now have five tables in the database. We are now ready to enter data in the tables that we have created. But before we do that, let us look in the Description and the Field Properties given in the table design view.

Setting Description and Field Properties

Entering a description for each field is not mandatory but it is always a good practice to add a description as it assists in documentation purposes. It helps the user or anyone who is trying to use the database to understand what is the purpose of each field.

The *Field Properties* option when used allows us to control and validate the data that is to be entered. It also identifies how the values in the field are stored and displayed. For example, we may decide in what formats like DD-MM-YY or MM-DD-YY the user will enter the data in the BirthDate field, how it should be displayed while printing or viewing the data and what message is to be given to the user in case of invalid date entry.

A set of different field properties is displayed related to the datatype that the user selects. It is possible to change all the field properties as per our requirement. Some of the common field properties are discussed in this section:

Default value: We can specify a value that will be stored by default in a field. Once we set this property for any field, the specified default value will automatically be displayed when we add a new record in our table. A user may change the value if required at the time of data entry.

AutoValue: This property is used with numeric fields, usually the one that is assigned as the primary key. For example, the Teacher table has TeacherID set as primary key and its datatype is integer. We expect values like 1,2,3… and so on to be stored in the TeacherID field. Thus we can enable the AutoValue property for this field. Let us set the AutoValue property for the TeacherID field in the Teacher table. Open the Teacher table in the design view, select the TeacherID field, we will be able to see the *Field Properties* as shown in figure 2.16. Select "Yes" from the drop down menu visible succeeding the *AutoValue* label.

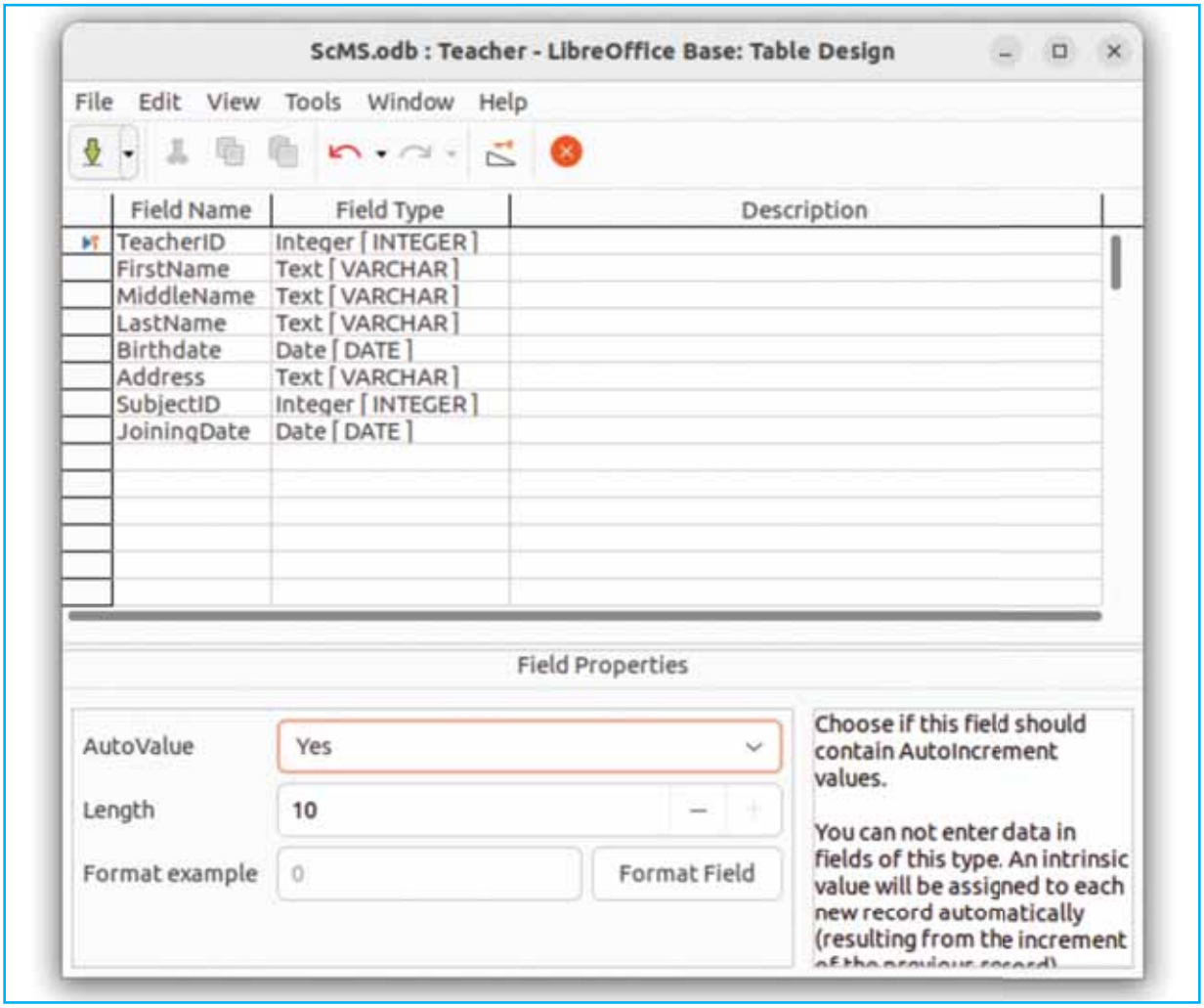


Figure 2.16 : Setting Field Properties

Length: The length property is associated with the text data type. It is automatically assigned a value for example 50 in case of Text [VARCHAR] datatype. We can assign the size of length property based on our needs to store different types of data. LibreOffice Base automatically assigns the predefined length size to different datatypes like Numeric, Date/Time, Yes/No and Memo, in such cases this property will be disabled on the screen. Observe that the value 10 is specified in figure 2.16 for length. As the field in discussion is numeric we will not be able to update it.

Entry Required: At times the data requirement in some of the fields that we use in the table is optional or mandatory. The entry required property allows us to manage such fields. To ensure that the data should not be left blank the field property needs to be set to "Yes". The default selection is "No". In the case of the Teacher table, FirstName, MiddleName and LastName fields may require this property to be set to "Yes".

Format: This property specifies a format of text, numbers or date that is used at the time of entry, display and print. The format property uses different settings for different data types. LibreOffice Base provides some predefined formats for Number, Date/Time, and Yes/No data types. Let us set the format for the joining date field of the teacher table. Open the table in the design view, select the JoiningDate field, click on the *Format Field* button similar to the one visible in figure 2.16. This will open a *Field Format* dialog box as shown in figure 2.17.

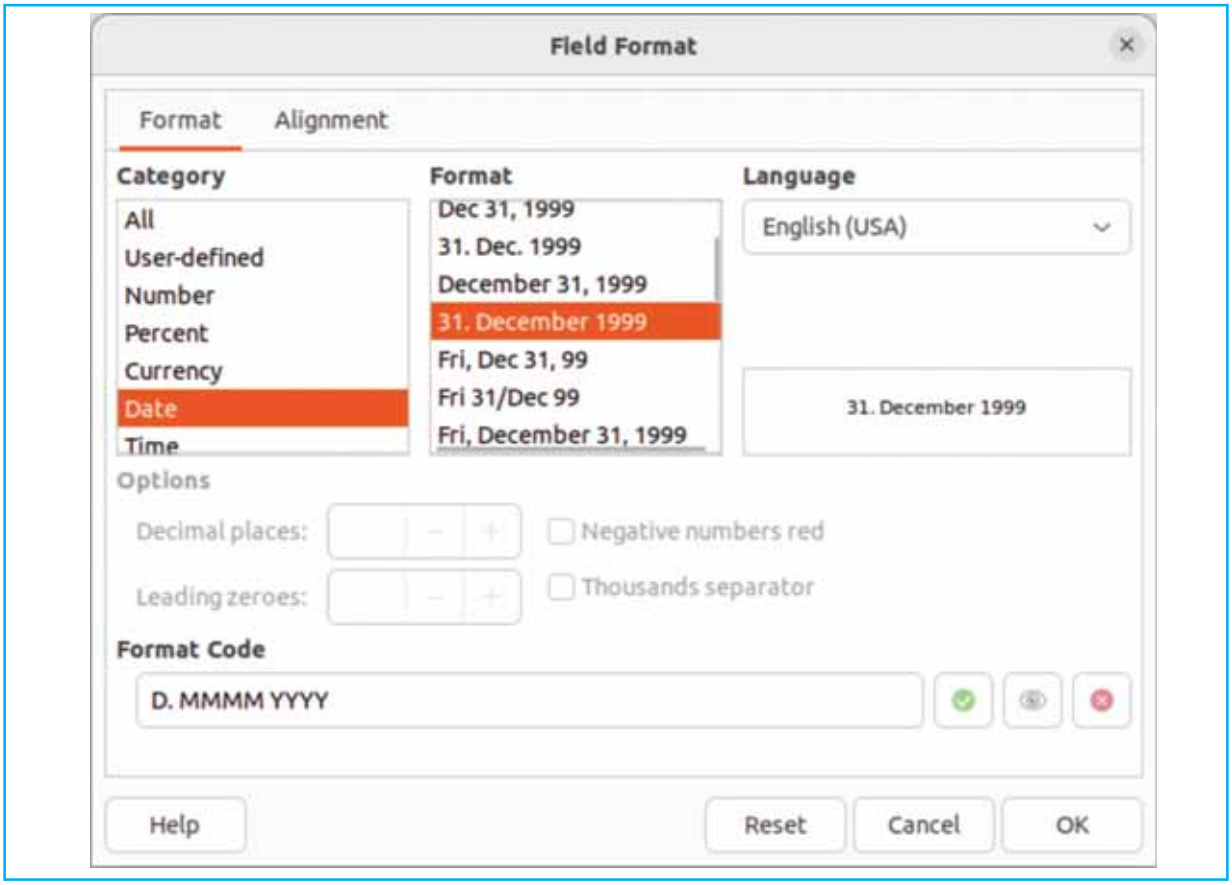


Figure 2.17 : Field Format Dialog Box

Observe that Date is selected under the *Category* label, select the option *31. December 1999* under the *Format* label. Click on the *OK* button to apply this format, LibreOffice Base will now automatically convert any date entered into D.MMMM YYYY format.

Operations on Table Data

Once the table structure is created and finalized, the next step is to insert the required data into the tables. Let us look at the different operations that can be performed on tables.

Insert Data

To insert records into the table, first we need to open the required table. To open the table go to the Tables Pane in ScMS database window and double click on the name of the table or right click on the desired table and click the *Open...* option from the menu. This will open the selected table in the Datasheet view as shown in figure 2.18. Note that if the primary key is not set we might not be able to enter the data.

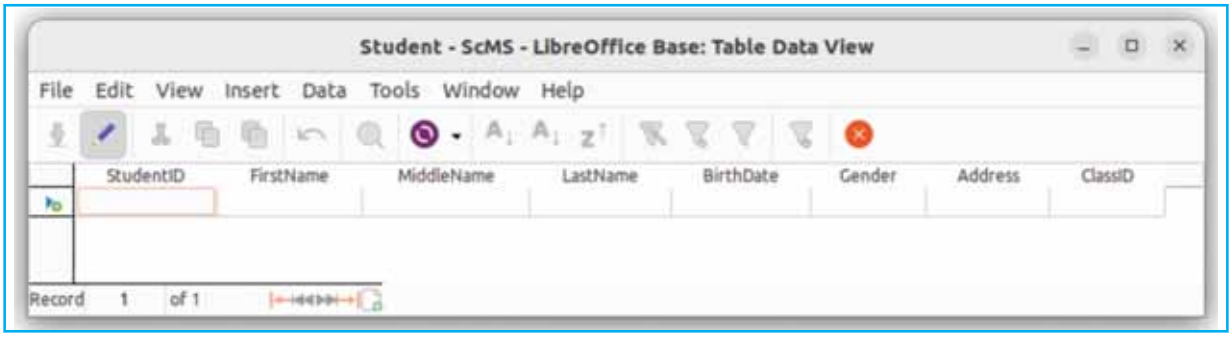


Figure 2.18 : Table Datasheet View

Observe that in figure 2.18, the field names are shown in a horizontal line known as Title line. Below the Title Line, there is a row consisting of empty boxes. The text box under the StudentID field is selected and has a blue arrow with a green + sign preceding it. The blue arrow is referred to as a *Record Selector* icon. It shows the current record that we are working on. Enter the data as shown in figure 2.19 so that we have five records in the Student table as of now.

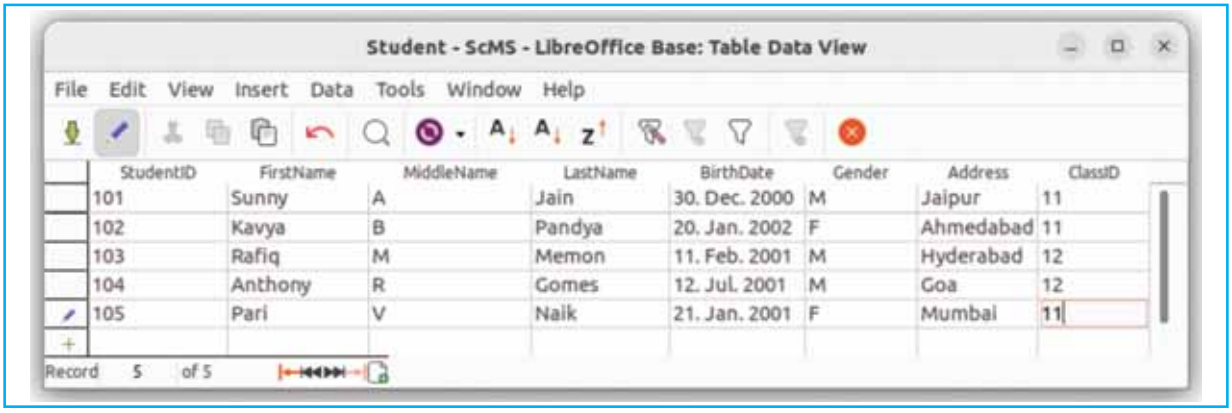


Figure 2.19 : Data of Student Table

While entering the data observe that the blue triangle changes to the pencil icon when we are entering data into a row. Also the last empty row has a green + sign visible at the beginning. To add a new record in the table we need to scroll to the last row and then click in any of its fields. Our cursor will now be positioned in the selected field and the icon will be changed to the blue arrow and green plus sign.

The bottom most part of the figure 2.18 is known as the Navigation Bar. It shows the number of records in the table as well as is capable of showing the position of any selected record. It also contains navigation buttons that allow us to scroll the records vertically. Observe that in figure 2.19 at the left bottom of the screen we are able to see 'Record 5 of 5' indicating that there are a total 5 rows in the table and currently we are at row number 5.

Edit Data

Updation or editing of data is a continuous process. The data might be updated for several reasons like wrong data entry or change in data due to genuine reasons. For example, what if Sunny shifts to Ahmedabad from Jaipur. In this case though the address of Sunny was correctly entered, we still need to edit it.

The process of correcting the data already entered in the table is usually known as Editing. To edit any data we need to open the table in the datasheet view, go to the desired field and place the cursor at the field value that we want to edit. We can thus make any changes that are needed.

Delete Data

To keep our database clean any unnecessary data or records in the table needs to be removed.

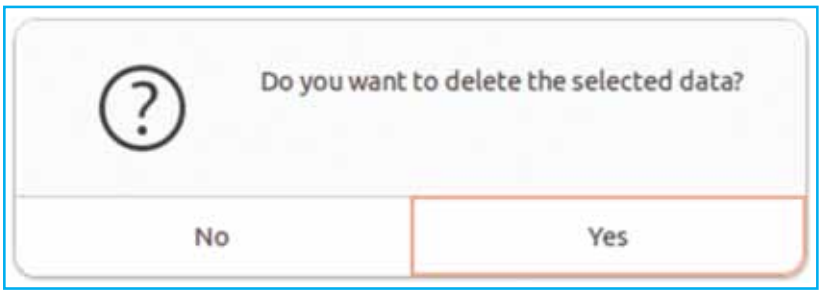


Figure 2.20 : Delete Alert Box

To delete a record from the table, open the desired table in the datasheet view, select the record or records (multiple records can be selected by selecting starting record, pressing a Shift key and selecting the last record).

The simplest way to delete the records is to press the Delete key on the keyboard. Alternatively we can select the *Delete Record* option from the *Edit* menu or right click the selected records and select the *Delete Rows* option from the sub menu. In all cases, we will be presented with a alert dialog box as shown in figure 2.20.

Click on the *Yes* button and the record will be deleted from the table. If the user selects *No* button then no action will be taken on the records.

Sort Data

The objective of storing data in a table is to be able to access it as and when needed. The data within a table should be organized in a specific way such that it is easy to obtain the desired information. As the number of records in the table grows, finding things can become a bit difficult. Sorting the data on a specific field is one way for arranging data properly. Assume that we wanted to know the students who reside in Ahmedabad by looking at the data in the Student table. We would be able to get this answer easily if the data was sorted on the Address field. Let us sort the data of the Student table.

We can sort the table in multiple ways, the simplest is to use the sort icons  shown in the datasheet view toolbar. This sort icon is similar to the *Sort...*, *Sort Ascending* or *Sort Descending* options of the *Data* menu.

The *Sort Ascending* or *Sort Descending* option is used for immediate sorting from the point of cursor. For example, if you are on the fourth record of a field, say MiddleName and we use this option, then the entire data of the table will be sorted ascendingly or descendingly based on the MiddleName field.

The *Sort...* option is an advanced sort option, it allows us to select multiple fields that we would like to sort the data on. When we use this option a Sort Order dialog box as shown in figure 2.21 is shown to the user.

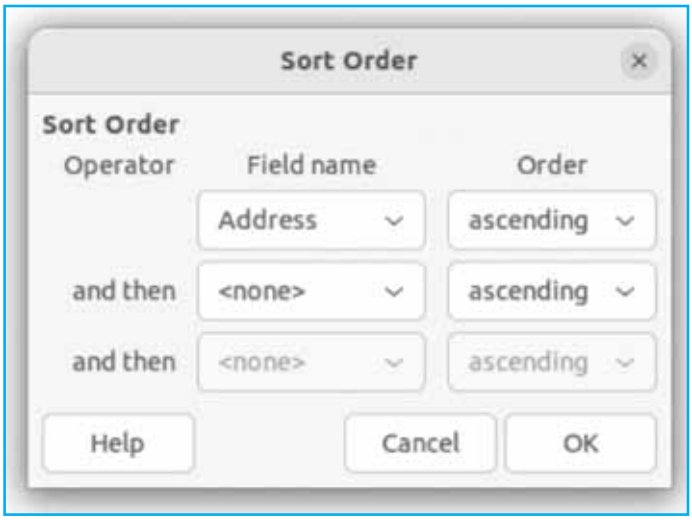


Figure 2.21 : Sort Order Dialog Box

The user now needs to choose appropriate field names from the dropdown menu under the *Field name* label. Observe in this case, we have chosen the field name Address. We can sort on a combination of a maximum of three fields. The sorted table is as shown in figure 2.22.

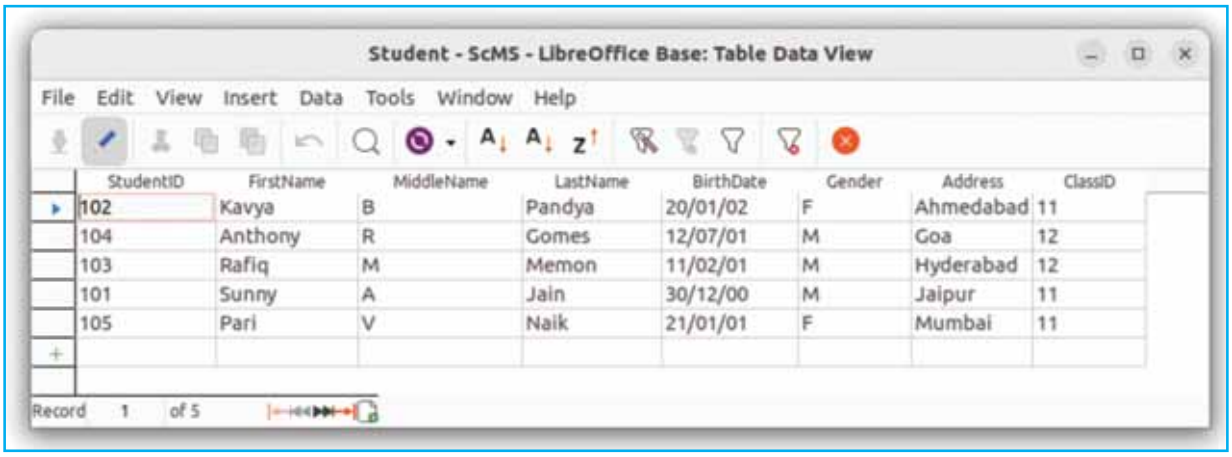


Figure 2.22 : Table Data Sorted on Address Field

Redundancy and Keys

Many times we may observe that the same piece of data is stored in more than one place, this is known as redundancy. It leads to inefficiency, inconsistency, and increased storage costs of a database. For example, the Grade table that we have created needs information about the student and subject both. If we had stored the name of the student as well as subject in it then we may have repeated the student and subject name multiple times leading to redundancy.

Redundancy is undesirable and is usually removed through a concept called database normalization. Here we split the data into related tables so each piece of information is stored only once. The primary key and foreign key discussed in Chapter 1 plays a very significant role in the normalization process, as it allows decomposition of tables into multiple tables and relating them.

We may apply other methods like using master data or automated data validation to control redundancy. In the case of the ScMS database the tables Student, Teacher, Subject and Class can be considered as master data, and are used as a single source of truth for common information about these entities. The Grade table here is a transaction table as it contains data that might be considered as temporary.

Creating Relationship

The tables that we have created so far in ScMS database are not related. We have simply used certain fields that are common. For example, the Student table has a field ClassID that is also part of the Class table, similarly the Teacher table has a field SubjectID that is also part of the Subject table. Having a common field name does not guarantee that the data entered in them will always be correct. What if a user enters a ClassID 21 in the Student table but it is not there in the Class table, in such a case the database would be termed as inconsistent. Establishing appropriate relationships among the different tables ensures that such inconsistencies do not happen.

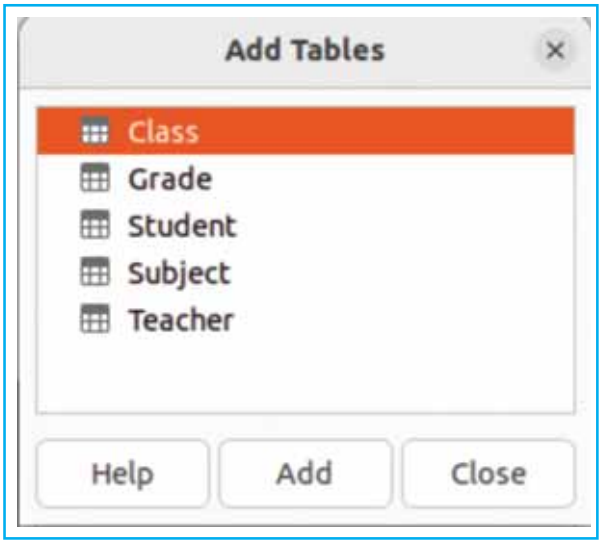


Figure 2.23 : Add Table Dialog Box

Let us now learn to establish relationships amongst different tables that we have created. Open the ScMS window, go to the *Tools* menu, select the *Relationships*... option from it. This will open a Add Tables dialog box with a list of tables created as shown in figure 2.23.

Select the tables one by one by double clicking on them or select one table at a time and click on *Add* button, we will see that the selected table along with its fields is displayed in a *Relation Design* window in the background. Once all tables are selected close the *Add Tables* dialog box. The *Relation Design* window will now look somewhat similar to figure 2.24.

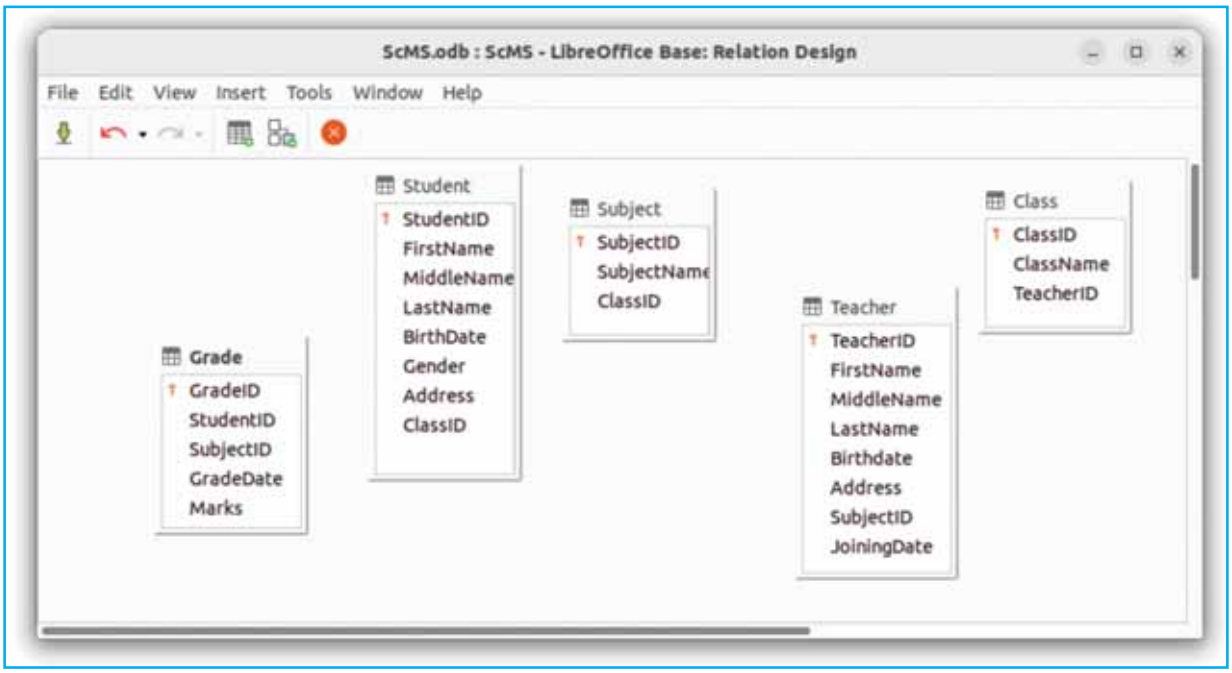


Figure 2.24 : Relation Design Window

It is possible to rearrange the positions of the table visible in figure 2.24 by using drag and drop operation. Also ensure that the primary key is set in all the tables otherwise we may get errors while creating relationships.

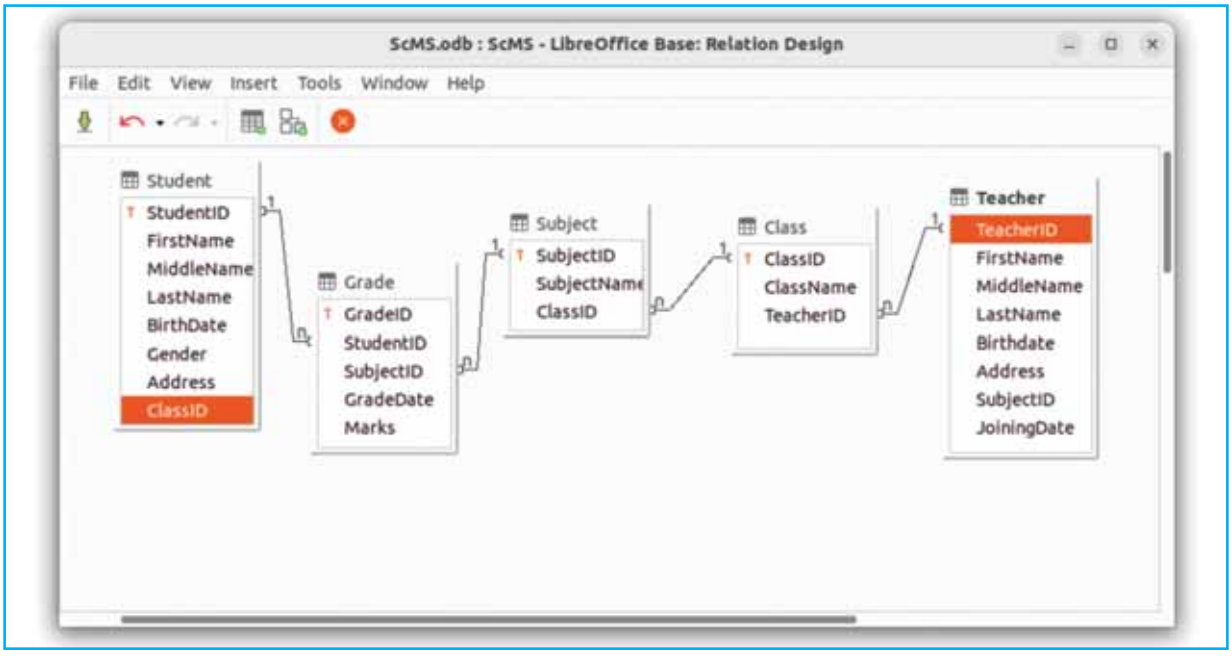


Figure 2.25 : Relation Design Window after Sample Relationship Setup

Finally to create a relationship, we will use the drag and drop operation. First click on the StudentID field of the Student table, keep the left mouse key pressed and drag the mouse on the StudentID field of the Grade table and release the left mouse key. We will see a connected line created between the two fields with labels 1 and *n*. Observe that label 1 is displayed on the StudentID (primary key) of the Student table and label *n* is displayed on StudentID (foreign key) of the Grade table. It indicates that the StudentID field in the Student table will hold unique values while the StudentID field in the Grade table may have repeated (*n*) values .

Note that when we are creating a relationship the data types of both the fields being related must be the same. Create all the relations possible and save them, the *Relation Design* window will now look somewhat similar to figure 2.25.

Once the relationship between tables is created, we can work on the data integrity constraints of the database. We are aware that the Student table and Grade table in ScMS database are related to each other. This relation states that the StudentID entered in the Grade table should be first available in the Student table. Records pertaining to students in the Student table are considered as master or parent records, while records pertaining to students in the Grade table are considered as transaction or child records.

Referential Integrity: Assume a scenario when a user deletes a parent record from the Student table. What should happen to its related child record in transaction tables? For example, if there is a record that has StudentID as 101 in the Student table and corresponding multiple records of this StudentID in the Grade table. What will happen to the records of StudentID with value 101 in the Grade table if the user deletes or updates StudentID with value 101 from the master table?

The referential integrity rule says that there must be no entry of data in the transaction table without related data in the master table. Thus there should be no unmatched foreign key values in the database.

Referential integrity needs to be enforced in a database while performing update or delete operations on records. Let us now try and enforce the referential integrity rules on the Student and Grade tables. Double click on the relationship line seen between Student and Grade table in figure 2.25. This will open a Relations dialog box as shown in figure 2.26.

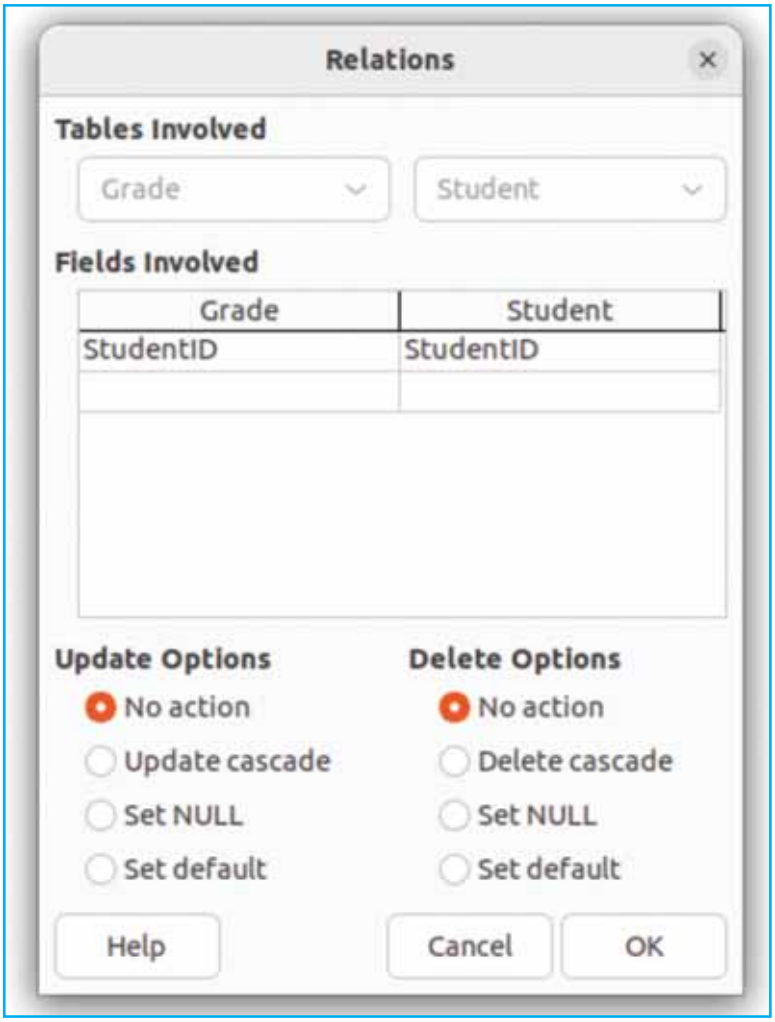


Figure 2.26 : Relations Dialog Box to set Referential Integrity

A database designer can now choose and select one of the four options seen in the figure 2.26 based on the transaction needs of the organization. We can select different option values for update and delete operations. For example, we may select *No action* option under the *Update Options* label and *Delete cascade* option under the *Delete Options* label. Let us see what each of these options are meant for.

No action: If this option is selected, the user will not be allowed to either delete or update the record if its related record exists in some other table. For example, if we try to delete a record with StudentID having value 101 in the Student table we will not be allowed to do so as corresponding records exist in the Grade table.

Update cascade: This option assigns an updated value to all the related fields if the master record is updated. For example, if we update a record with StudentID having value 101 to 125 in the Student table, then all the corresponding StudentID records in the Grade table will be assigned value 125.

Delete cascade: This option deletes all the related fields if the master record is deleted. For example, if we delete a record with StudentID having value 101 from the Student table, then all the corresponding StudentID records in the Grade table will also be deleted.

Set NULL: This option assigns a NULL value to all the related fields if the master record is deleted or updated. For example, if we delete a record with StudentID having value 101 in the Student table, then all the corresponding StudentID records in the Grade table will be assigned a NULL value.

Set default: This option assigns any fixed default value to all the related fields if the master record is deleted or updated. For example, if we delete a record with StudentID having value 101 in the Student table, then all the corresponding StudentID records in the Grade table will be assigned a default value.

We can delete or edit the relationship that has been established between the tables if required. To do so, open the *Relationship Design* window, select the desired relationship line visible between the tables, right click on it, a popup menu with *Delete* and *Edit...* option will appear. Perform the operation required and save the relationship again.

We are now ready with a database that can be used to store and analyze data. Enter proper records in all the tables designed so that we can generate information out of it using queries.

Summary

In this chapter, we learned how to use LibreOffice Base, a tool for creating a database. It provides four objects namely Tables, Queries, Forms and Reports. We saw how to create tables in two ways; using a Wizard and custom design. A wizard is a guided, step-by-step graphical interface tool that simplifies complex tasks by prompting users for information and automates the process of performing the task. We also learned about the Field Properties option, which can be used to control and validate the data that is to be entered in the records of the table. Finally we learned how to set relationships amongst tables that allows us to control the data redundancy and also establishes referential integrity. We are now ready to create tables and perform different operations on them. The next chapter will teach us how to get the required information from the tables using queries.

EXERCISE

1. Explain the term Database. What is the use of a Database?
2. What is a table? Explain the different ways in which a table can be created.
3. What is the use of Wizard?
4. Explain the significance of relations between tables.
5. Explain the significance of the labels 1 and *n* in table relationships.
6. Explain the concept of data redundancy giving an appropriate example.
7. What is the use of Normalization?
8. State why Referential Integrity is required in the database.
9. What does a field property signify?
10. Explain the importance of Format field property.

11. State whether true or false?

(1) LibreOffice Base allows us to create text documents.

(2) The Authors table is provided under the business category tables of LibreOffice Base.

(3) Customized tables are created using Table Wizard.

(4) The Field Properties option allows us to control and validate the data that is to be entered.

(5) The Record Selector icon shows the current record of the table we are editing.

12. Fill-in the blanks.

(1) LibreOffice Base assigns ..... extension by default to the database created in it.

(2) The Table Wizard provides templates of two categories of tables, ... and Personal.

(3) In the table design view we can see field name, type, ... and properties.

(4) The AutoValue property is used with ..... fields.

(5) Redundancy is removed through a concept called .....

13. Multi-choice questions. Choose the most correct answer.

(1) LibreOffice Base allows us to create databases in how many of the following ways?

(a) 1 (b) 2 (c) 3 (d) 4

(2) Which of the following objects is not visible when we open the LibreOffice Base database window?

(a) Table (b) Queries (c) Forms (d) Views

(3) Which of the following operations can be performed on the records in the database to arrange them in the proper way?

(a) Insert (b) Delete (c) Update (d) Sort

(4) Which of the following best describes data redundancy in a database?

(a) Decomposition of tables (b) Relating tables

(c) Repetition of data (d) Deletion of data

(5) Which of the following statements is not correct?

(a) Names of Primary key and Foreign key should be the same.

(b) Every table must have a primary key.

(c) A primary key can consist of multiple fields.

(d) The data type of primary key and foreign key should be the same.

(6) Which of the following labels would be visible on the relationship line if we relate the StudentID field of Student and Grade table?

(a) n…n (b) n…1 (c) 1…n (d) No labels

(7) While setting referential integrity between tables, which of the following will not allow the user to either delete or update the record if its related record exists in some other table?

(a) Set default (b) Set NULL (c) No action (d) Delete cascade

(8) Which of the following field properties can be considered as equivalent to Not NULL?

(a) Default (b) Format (c) Length (d) Required

(9) In which of the following, the data that is used as a single source of truth for common information about entities is stored?

(a) Transaction Table (b) Master Table

(c) Big Table (d) Accounts Table

(10) Which of the following is the benefit of using foreign keys?

(a) Data consistency (b) Faster data retrieval

(c) Improved data redundancy (d) Reduced storage space

Laboratory Exercise

1. For the databases listed in following table, perform the following operations:

(a) Create tables.

(b) Identify appropriate primary key and foreign key for each table.

(c) Decide what data type will be suitable for each attribute in a table.

(d) Establish relationships amongst the tables.

(e) Insert at least ten records in each table.

DB 1	Student (StudentId, StudentName, Address, City, BirthDate, ContactNo, Email) Book (BookId, BookTitle, Description, BookAuthor, Status, PublishYear) Book_Issue (BookIssueId, BookId, StudentId, IssueDate, ReturnDate, FineAmount)
DB 2	Product (ProductId, ProductName, Quantity, ProductPrice, ManufactureYear) Salesman (SalesmanId, SalesmanName, Address, BirthDate, ContactNo, Gender) SalesOrder (SalesId, SalesmanId, ProductId, QtySold)

DB 3	Customer (CustomerId, CustomerName, Gender, Address, City, Area, Email, ContactNo) Magazine (MagazineId, MagazineName, UnitRate, PublisherName, PublishedMonth) Subscription (CustomerId, MagazineId, StartDate, EndDate)
DB 4	Employee (EmployeeId, EmployeeName, Address, City, Salary, DesignationId) Designation (DesignationId, DesignationName, BasicSalary) Project (ProjectId, ProjectName, StartDate, ProjectPrice) ProjectWorkHrs(PId, ProjectId, EmployeeId, HoursWorked)
DB 5	Vehicle (VehicleId, VehicleType, Price, Description, ManufactureDate) Customer (CustomerId, CustomerName, Address, BirthDate, ContactNo) VehicleOwner (VehicleId, CustomerId, PurchaseDate, DeliveryDate)

Data Retrieval using Queries

Introduction

In previous chapters, we have studied about the database and how we can create databases using LibreOffice Base. Base is a free and open source relational database management system which provides a graphical interface to create databases. As we know, in Base the data is stored in tabular form. Base provides a very user-friendly mechanism to store and retrieve data from the tables. Interaction with databases happens through a query language. In this chapter, we will learn about how to retrieve information from the database. We will first learn about query and then we will discuss how we can retrieve data using query.

Example Database for Query Processing

Following the steps discussed in the previous chapter, let us first create a database as shown in the figure 3.1. It is a student database, which stores information related to students, grades, subjects, classes and teachers. Now if we want to know, 'how many students have passed with more than 70% of marks?' or 'How many students are studying in Class IX?', we can query the database and get answers to our questions. In the following sections, we will learn about how to write queries to retrieve information from the database.

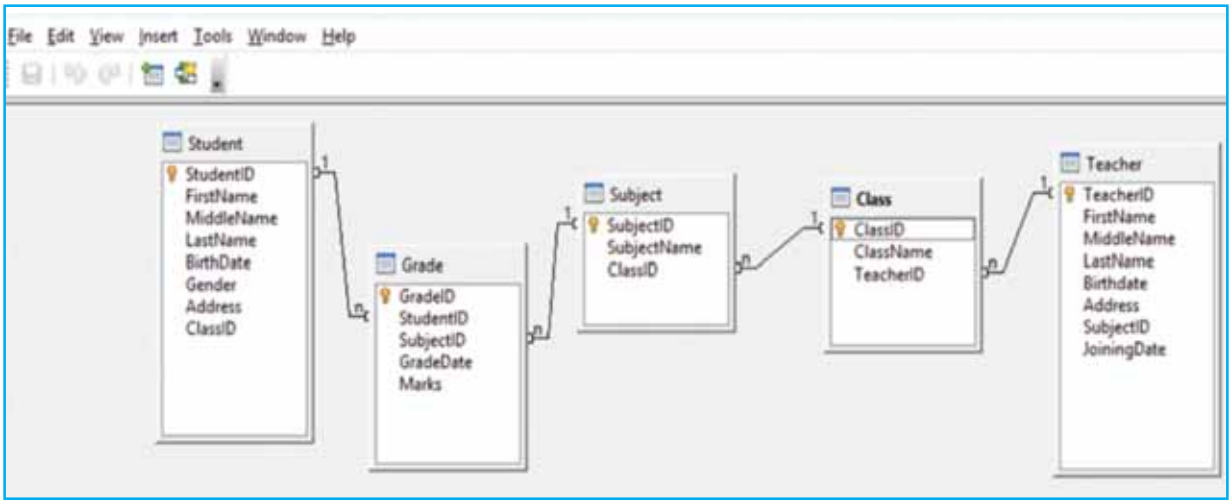


Figure 3.1 : Student Database

What is a query?

A database query is a request made to a database to retrieve, modify, or manage data. It serves as the primary means for interacting with databases, enabling users and applications to access and manipulate stored information, efficiently. At its core, a database query is a command written in a query language, (most commonly SQL -Structured Query Language) to perform operations such as:

- **Retrieving data**: Extracting information from one or more tables.
- **Inserting data**: Adding new records to a table.
- **Updating data**: Modifying existing records.
- **Deleting data**: Removing records from a table.

Table 3.1 summarizes the basic SQL queries.

Retrieval	*SELECT * FROM STUDENT;* Retrieves and displays all data from the STUDENT table
Insert	*INSERT INTO SUBJECT (SubjectID, SubjectName, ClassID)* *VALUES (1, 'Science', 9);* Inserts a new record into Subject table with SubjectID = 1, SubjectName = 'Science' and ClassID = 9
Update	*UPDATE SUBJECT SET ClassID = 8 WHERE SubjectID = 1;* In Subject table change the value of ClassID to 8 in the row with SubjectID = 1
Delete	*DELETE FROM SUBJECT;* *Deletes all rows of the Subject table.* *DELETE FROM SUBJECT WHERE SubjectName = 'Science';* Deletes those rows of the Subject table in which SubjectName is 'Science'

Table 3.1 : SQL Query Examples

Base provides a very user-friendly interface to write queries. Using this query interface, we can define a set of rules for fetching information from the database tables. In other words, one can tell Base to display exactly which fields and records a person would like to view from the database. The result of a query is returned in the form of a table. It consists of a set of records organized in rows and columns.

When we open a database in Base, the left side panel shows the *Queries* icon. Clicking the *Queries* icon located in the left side pane, shows different ways to create a query in the *Tasks* pane. Figure 3.2 shows the window when the *Queries* icon is selected.

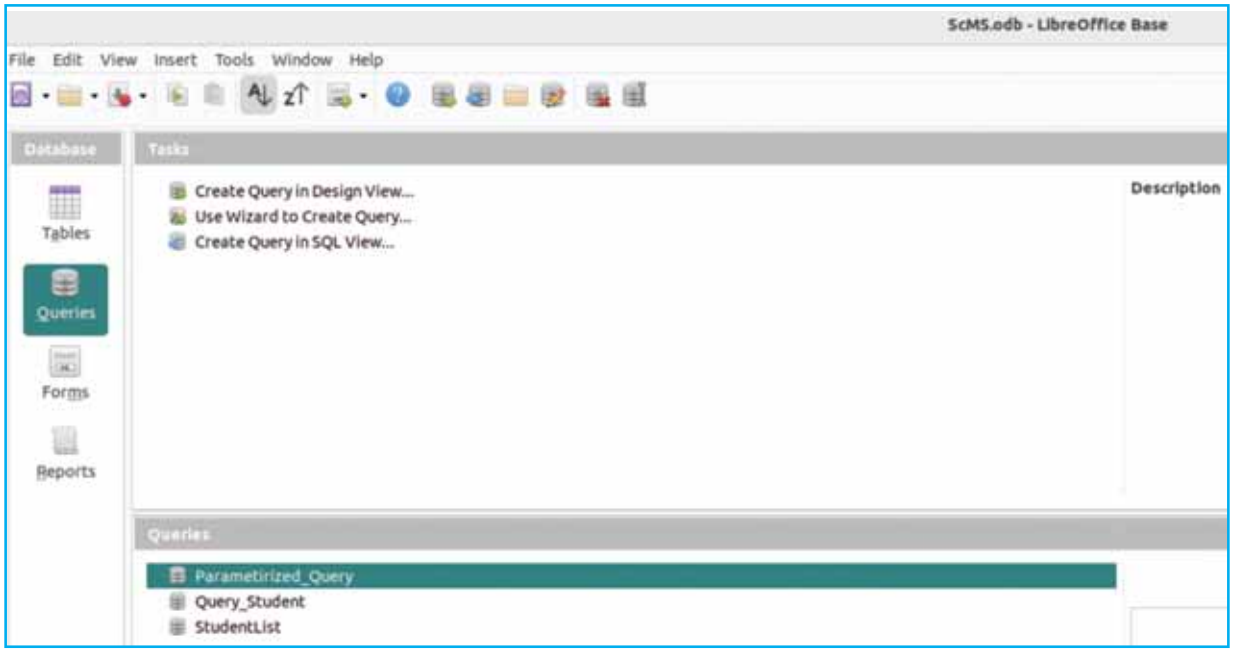


Figure 3.2 : Queries Window

Creating Query Using Wizard

The easiest way to query a database is through the query wizard. Double clicking the option '*Use Wizard to Create Query'* opens a Query Wizard dialog box as shown in the figure 3.3. As can be seen on the left pane of the dialog box, there are a total eight steps required to create the query. However, not all of them are compulsory. In fact, only the first step, *Field selection*, is compulsory. It helps us in identifying the fields that are to be displayed in the output. Other steps allow us to format the output and can be skipped if not required.

Step 1: The first step in creating the query is to select the table and the set of fields in that table from which information is to be retrieved. Once we select the table from the dropdown under the *Tables* label, *the Available fields* list box on the left side shows the fields which can be used in the query. We can move the fields using the left and right arrow from *Available fields* to *Fields in the Query* and vice versa. The fields that we finally select to be part of our query will be listed under *Field in the Query* list box. These fields can then be arranged in the order required using the up and down buttons. Once the fields are finalized, click on the *Next* button. Observe that in figure 3.3, we have selected *Table: Student* and can see all the fields related to it under the *Available fields* list box.

Step 2: The second step is to mention the sort order in which the output of the query will be displayed. It allows us to select up to four fields for deciding the sorting order of the output. For example, we might want to display the query result sorted in order of first name initially, and then by last name of the student. Once the sorting order is decided, click on the *Next* button. Figure 3.4 shows how to select the sorting order of fields.

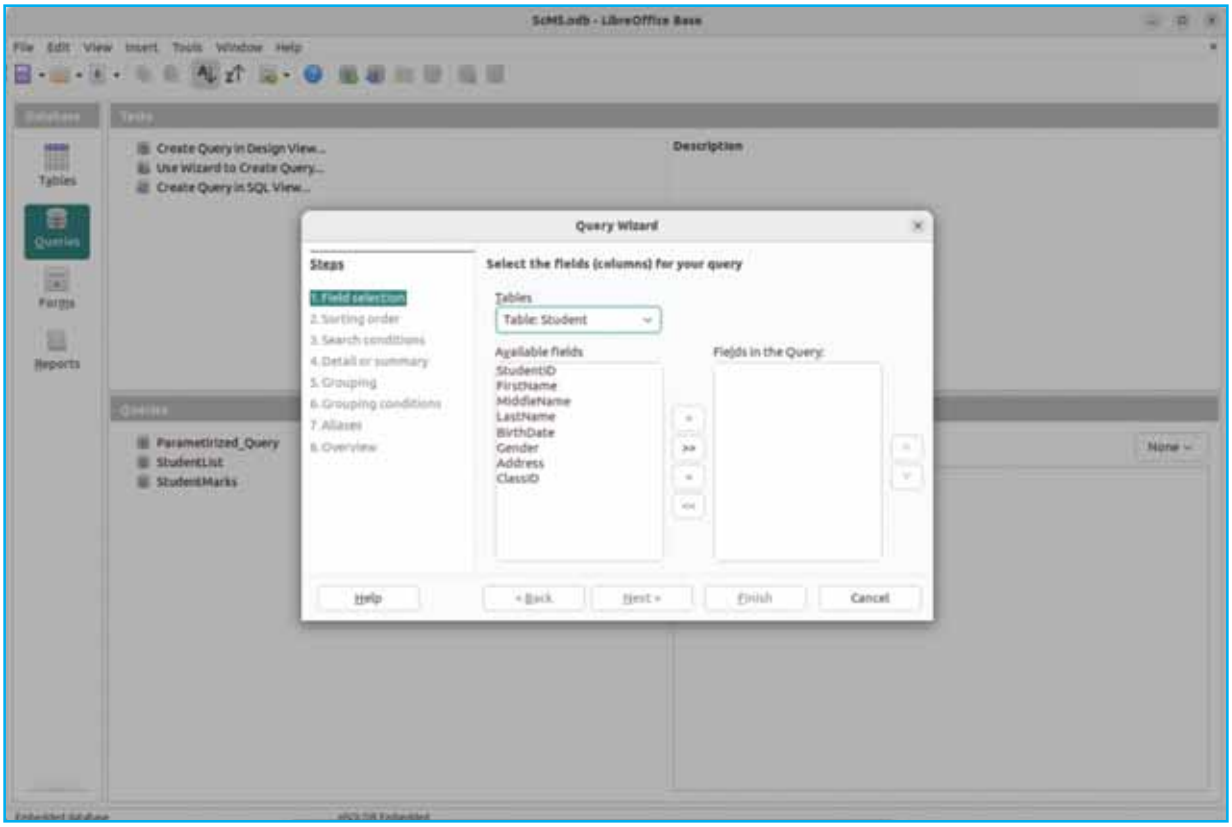


Figure 3.3 : Selection of Table and Fields for Query

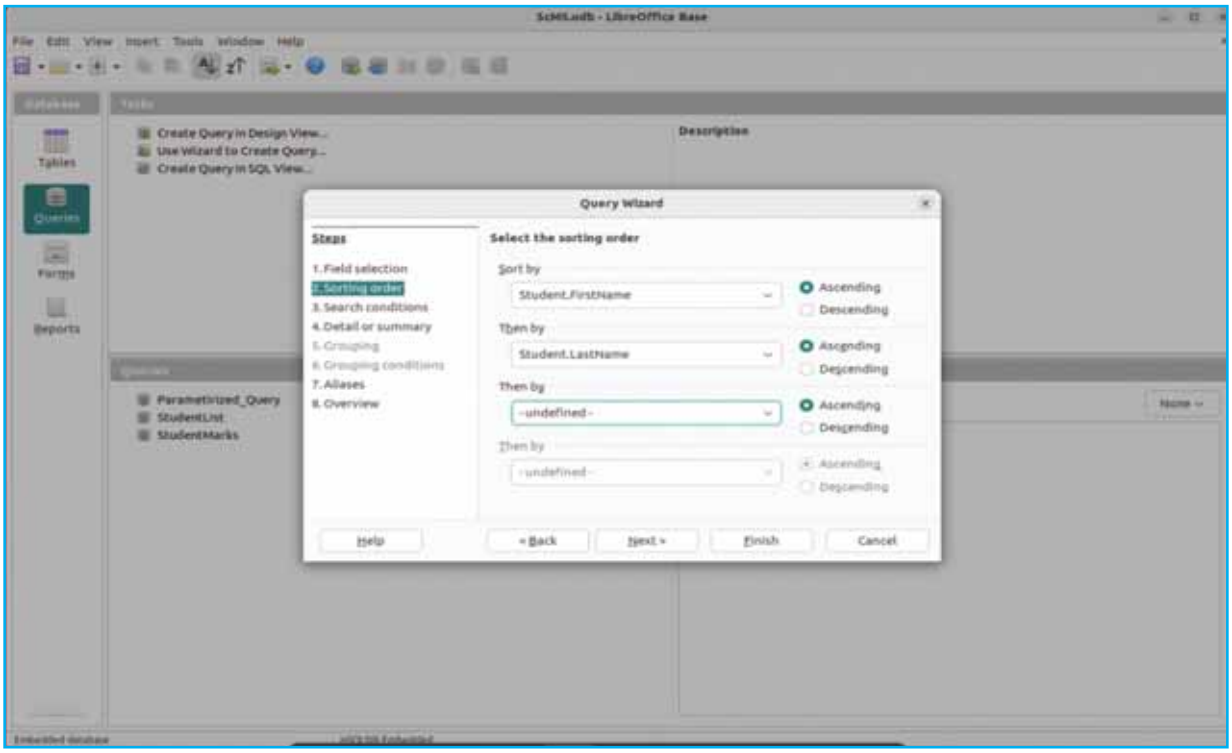


Figure 3.4 : Applying Sorting on a Field

Step 3: In the third step of the wizard, we set up the query. Here we have to select appropriate values for the *Fields*, the *Condition*, and the *Value* parameters. We can define a maximum of three search conditions in one query. In the case of displaying students name, if we want a list of male students, the criteria would be simple, we would select the *Gender* field from the drop down under the label

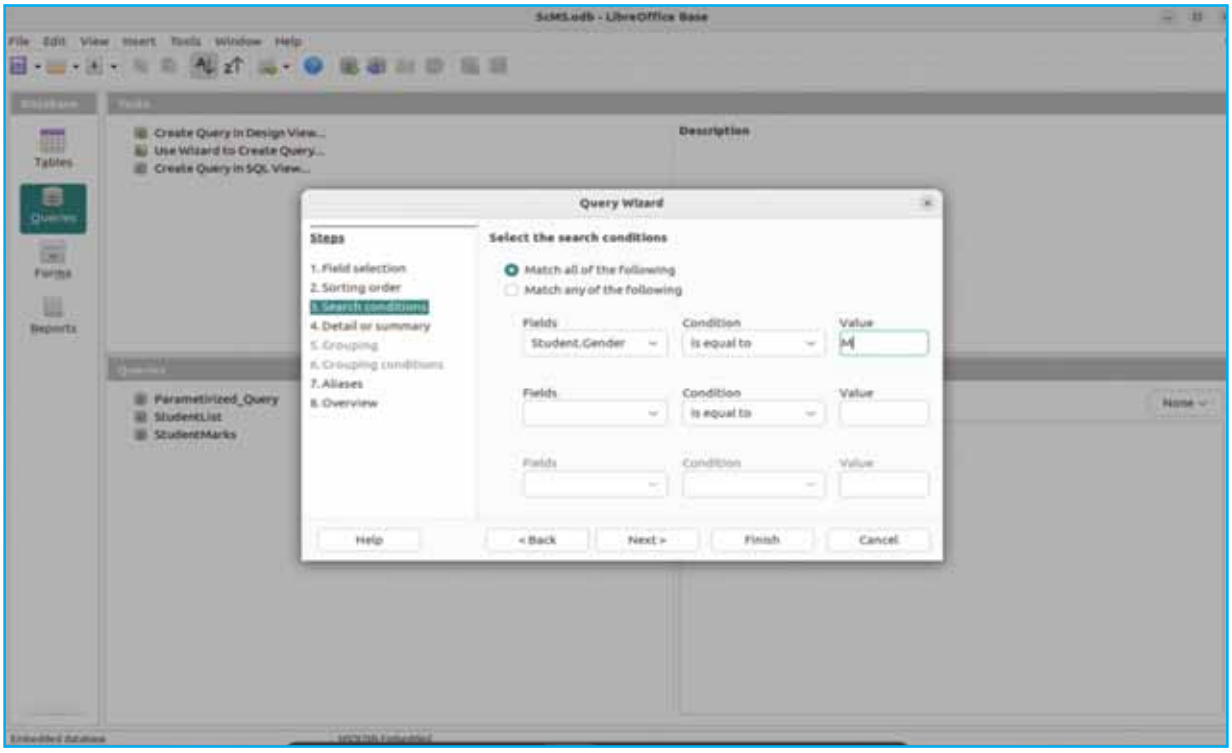


Figure 3.5 : Applying a Search Condition on the Field

Fields, then select *is equal to* from the drop down under the *Condition* label and finally in the text box under the label *Value*, type *M*. Observe that we have two options in this step, *Match all of the following* and *Match any of the following*. Since we have only one condition, we would not need to change the default setting. If multiple search conditions are to be set, like if we are looking for students with genders such as *M* or *F*, then we need to choose *Match any of the following*. Once the search conditions are finalized, click on the *Next* button. Figure 3.5 shows the search condition settings for listing all male students.

Step 4, 5, 6: These steps are specific to options to summarize and group the items. In our example query, the selected Student table does not contain any numeric fields, and so steps including options to summarize or perform numerical calculations are skipped.

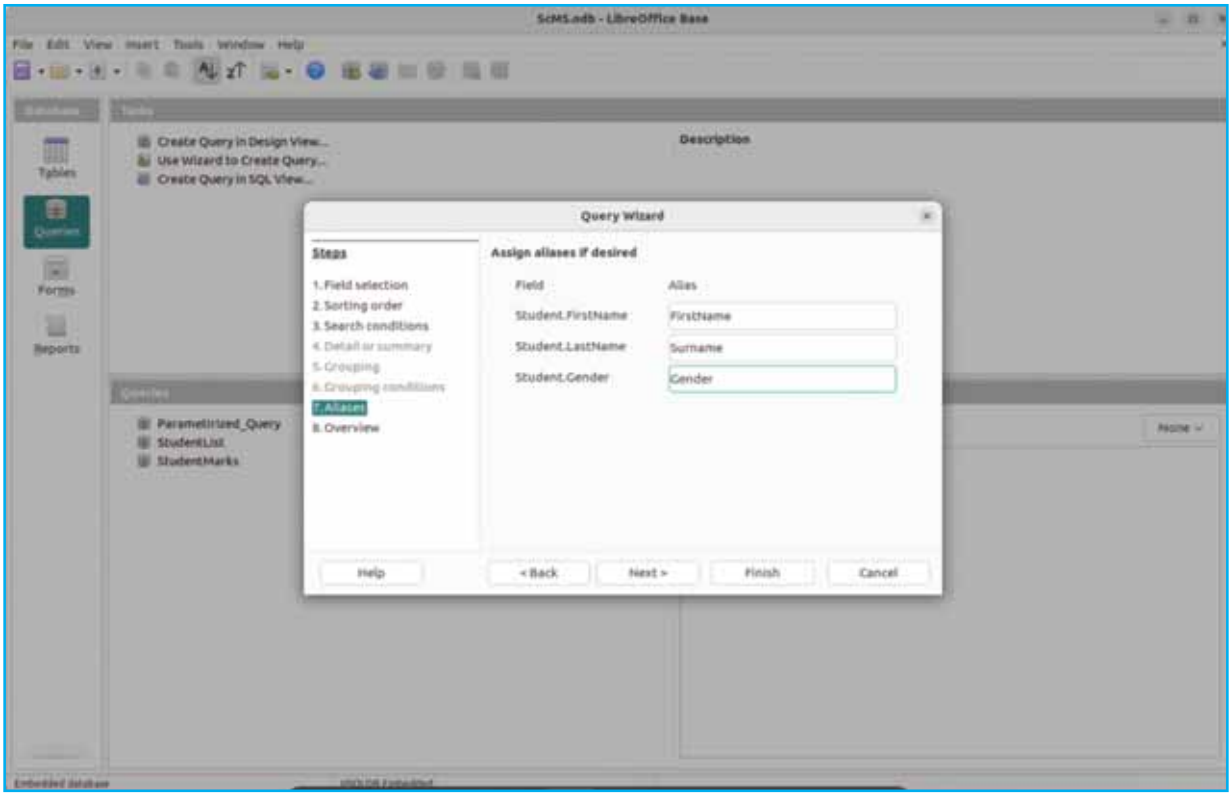


Figure 3.6 : Adding Aliases

Step 7: In the seventh step, *Base* expects aliases for selected field names. The actual field names, usually given by a database programmer, may not be in the format easily understandable to the user. For example, a database field name may have some special characters like *first_name,* or it can be an abbreviated form like *fName*; for such cases we may create an human understandable alias *First Name*. Thus the purpose of creating aliases is to make the *Query Wizard* display the field names in more human readable form. This step is also optional; we may add aliases only if required. If we do not add aliases, then the field names of a table will be displayed as it is. Once all aliases are decided, click on *Next* button. Figure 3.6 shows how to add aliases.

Step 8 : Finally, in the eighth step, we are given an overview of all the steps performed till now. Figure 3.7 gives us an overview of the query recently created. Assign a desired name to the query by typing it in the text box labeled *Name of the query*. In our case, we have named it *Query_Student* Take a moment to look over what we have done. In case changes are to be made, use the *Back* button to make the desired changes.

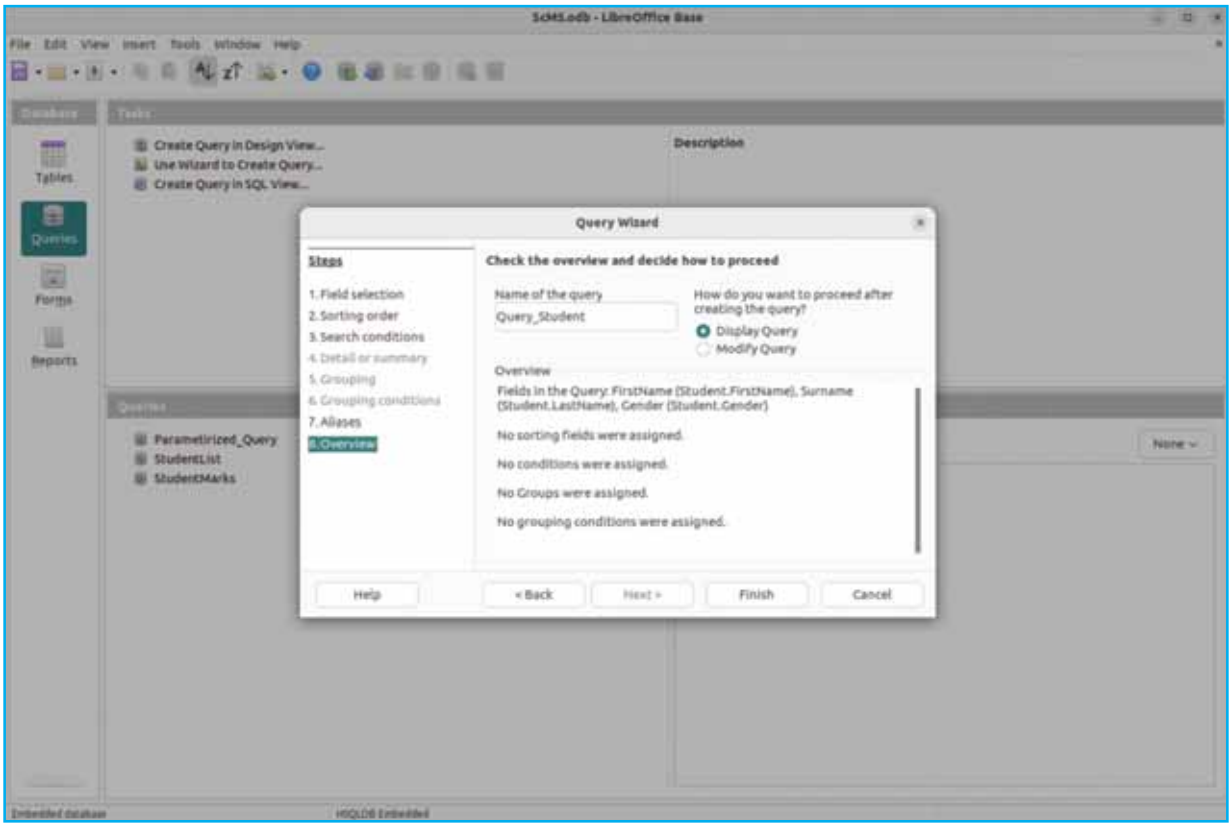


Figure 3.7 : Overview of Created Query

As we click on the *Finish* button, the query result is displayed in Datasheet view as shown in figure 3.8. (Note that the names 'Anthony Gomes, Rafiq Memon and Sunny Jain' are values fetched from the database table. If we have inserted other names like “Ramesh, Suresh” then it will show those names.)

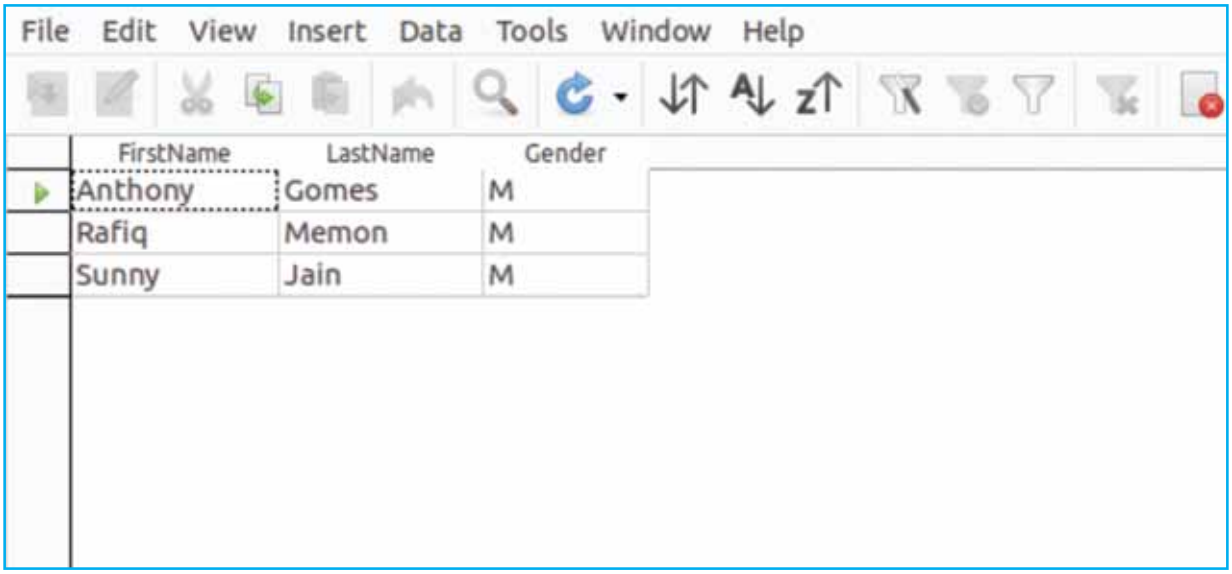


Figure 3.8 : Result of Query

Creating Query Using Design View

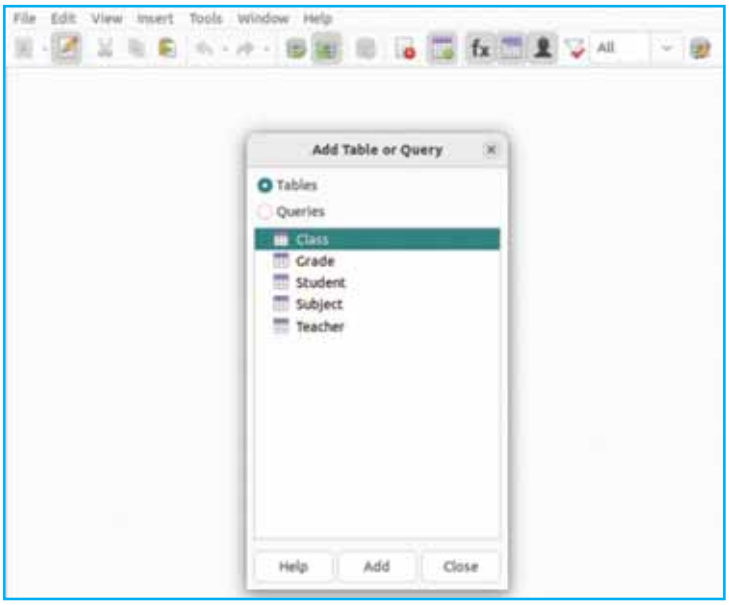


Figure 3.9 : Add Query or Table Dialog box

The *Query Wizard* which we have used in the previous section is simple and useful for the beginners. Design View is another option to create queries in *Base*. While the *Query Wizard* is a guided, step-by-step tool for creating basic queries, *Query Design View* offers more advanced control and customization options for query creation and modification. The wizard is helpful for simpler queries and for users who prefer a more guided approach, while the design view is useful for complex queries or when more fine-grained control is needed.

We will now create a query using Design View. In the *Queries* tab shown in figure 3.2, click on *Create Query in Design View*. It will open the *Add Table or Query* dialog box as shown in figure 3.9.

- Select the *Student* table and click on the *Add* button.
- Similarly, select the *Grade* and *Subject* tables. We can see three tables in the table pane, as shown in figure 3.10. *Base* also displays the relationships.
- Click on the *Close* button to close the *Add Table or Query* dialog box.
- Double click on *StudentID*, *FirstName*, and *LastName* from the *Student* table. Similarly, select the *SubjectName* field from the *Subject* table and the *Marks* field from the *Grade* table. The field names, along with their respective table names, will be displayed in a grid as shown in figure 3.10.
- Observe that we are also able to see some record (row) headings like *Alias*, *Sort*, *Visible*, *Function*, *Criterion*, and *Or*. By default the *Visible* option for each field is set to true. It indicates that all selected fields will be displayed in the output.
- As discussed in the previous section, *Alias* can be used for displaying meaningful names for the fields. For example, in place of *Firstname*, we would prefer to use *Name of Student* as a column title in the query result. Type *Name of Student* in the text box visible after the row heading Alias under the *FirstName* column.
- Sorting allows us to display records in ascending or descending order of the values. To display students' records in alphabetical order of names, select ascending in the *Sort* option under the *FirstName* column. Similarly, to display the records in the ascending order of *StudentID* numbers, select ascending from the drop-down box visible after the row heading *Sort* under the *StudentID* column.
- Click on the *Run Query* button on the *Query Design* toolbar. A query result similar to the one shown in figure 3.11 will be displayed.
- To save a query for later use, select the *Save* option from the *File* menu. Alternatively, click on the *Close* button, and *Base* will display a *Save* dialog box.
- Type the desired name, for example *StudentMarks*, and click on the *OK* button.

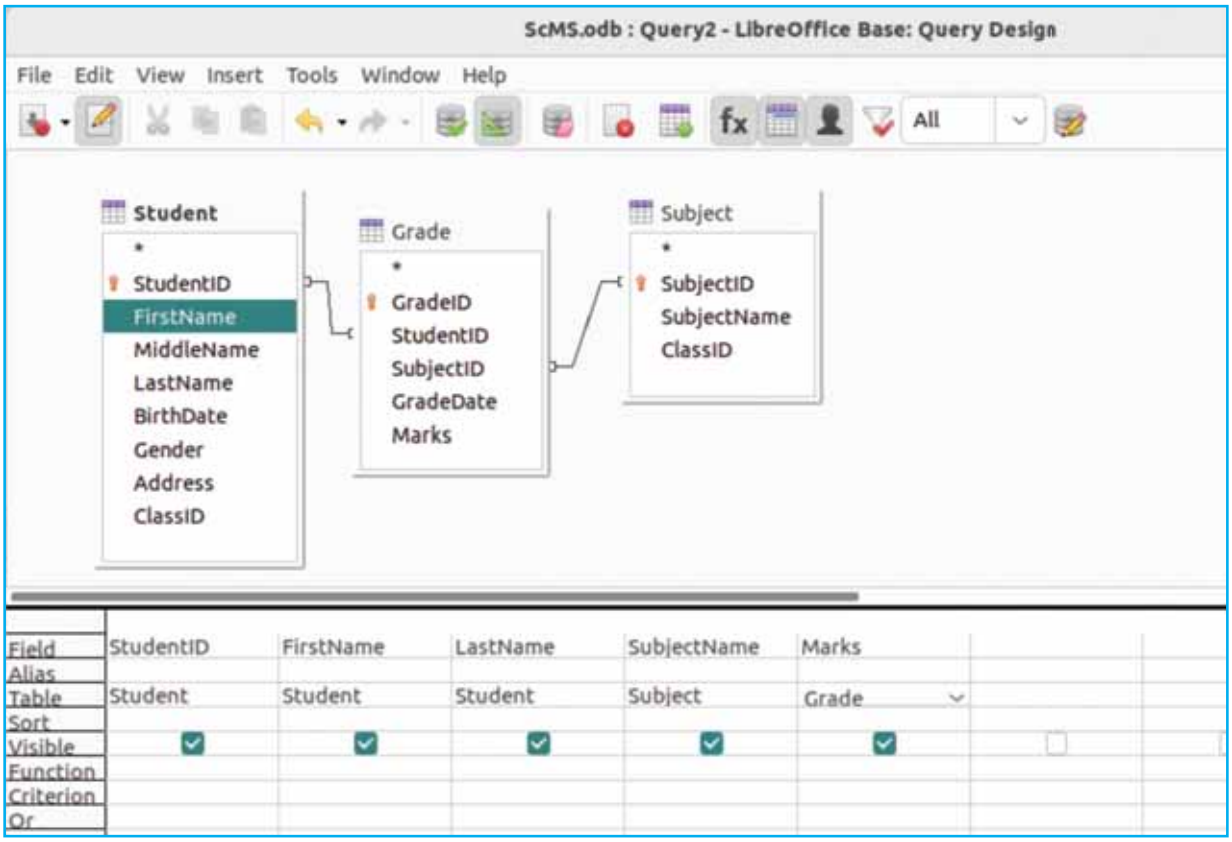


Figure 3.10 : Selection of Fields

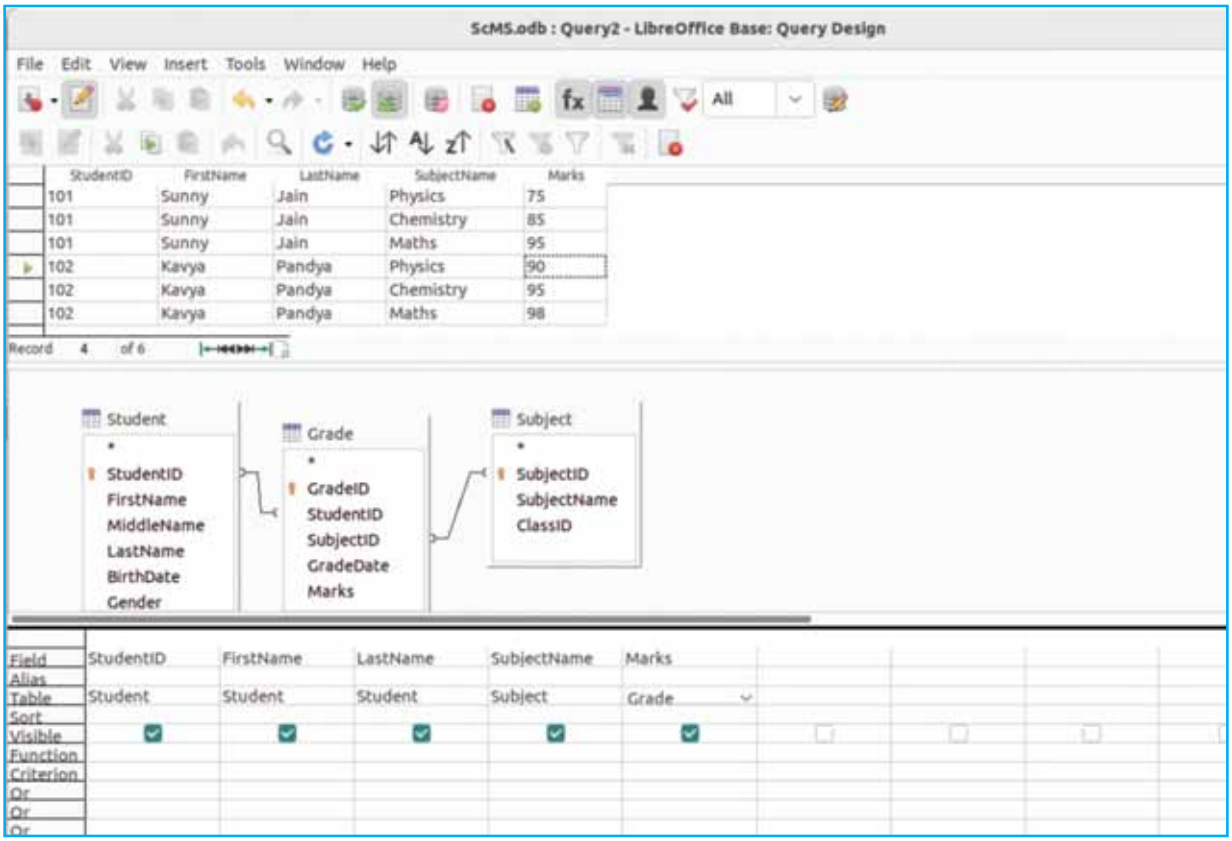


Figure 3.11 : Query Output

Queries with Criteria

We have seen that we can write a query that displays selected fields of a table. Now, suppose, instead of viewing all records, we wish to view the records which meet a certain criteria, e.g. details of only female students. This means we want *Base* to display a subset of selected records. To do this, we can specify a criterion that limits the records to only those, where the Gender field contains *F* as a value.

Using Single Field

In the *Criterion* cell of the *Gender* field type '*F*' as shown in figure 3.12. Note that, the criteria text must be enclosed within a quotation (' ') delimiter, while the number literals do not need any delimiters. If we fail to put delimiters, *Base* will not report any error; instead, it will apply delimiters on its own.

Save and run the query as discussed in the previous section, and we will get the list of female students as the output.

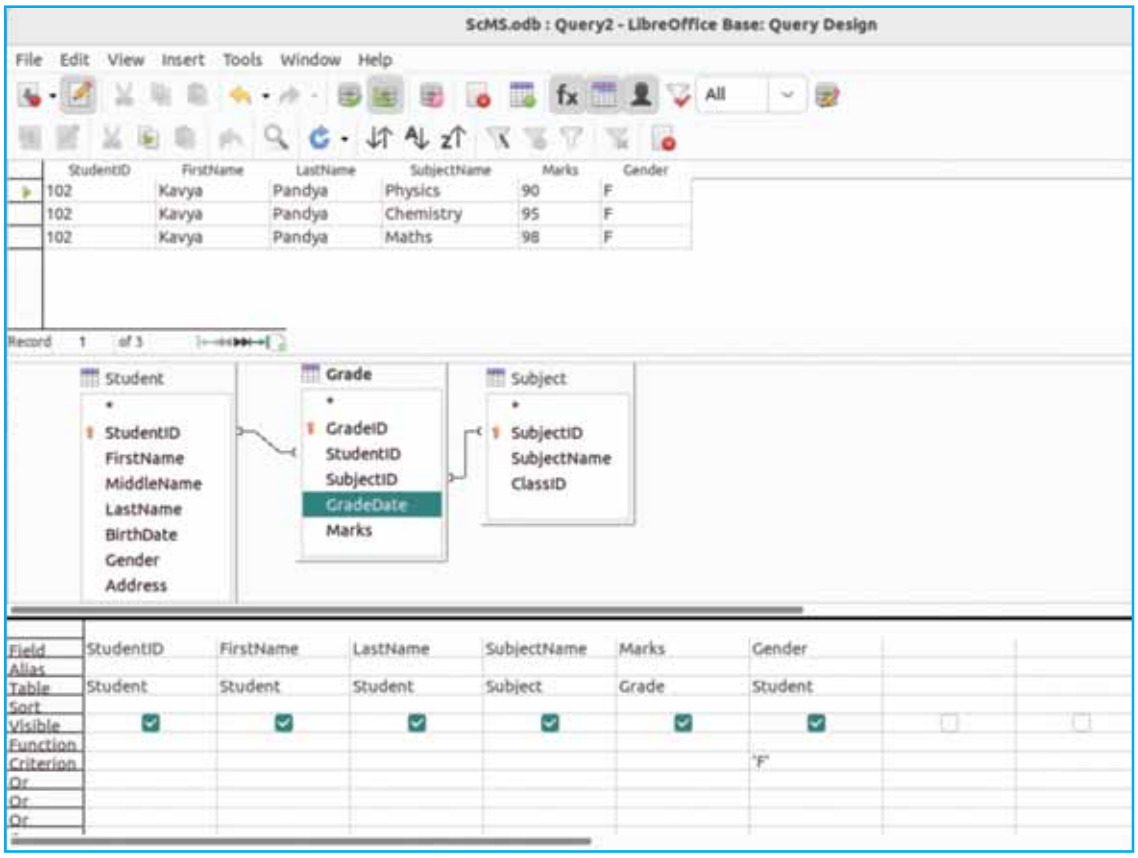


Figure 3.12 : Setting Criteria

Using Logical Operators

Apart from the constant values used as shown in figure 3.12, *Base* also allows us to design expressions for defining criteria based on different logical operators i.e. print names of students whose grade is more than 60%.

Suppose we want to display the list of students who are born on or after *1st January 2001*. Then create a new query in *design view*. Add the *Student* table. Type *">=#01/01/2001#"* in the *BirthDate* field's Criterion cell. Now, save and run the query. The output will be similar to figure 3.13

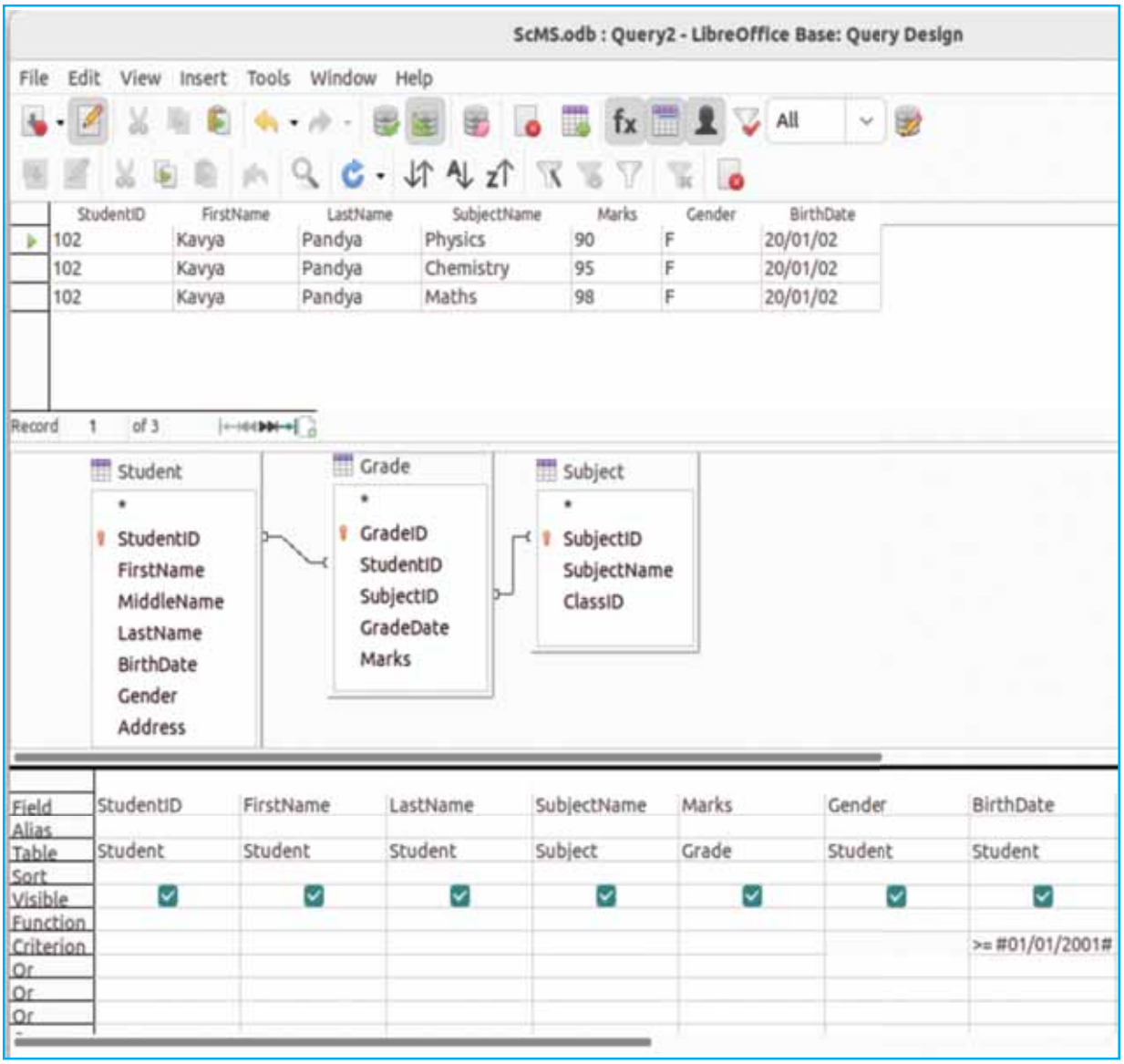


Figure 3.13 : Applying Criteria in Date Field

Similarly, to display students who borned between *1st January 2000* and *31st December 2001*, The Criterion in the *BirthDate* field can be set as *">= # 01/01/2000 # AND <= # 12/31/2001 #"*. *Base* also offers the *BETWEEN* operator to specify the same criteria as shown in figure 3.14.

Using Wild Card

Suppose we want to see the list of students whose first name starts with the alphabet *K*. In order to do this, create a new query using the table *Student*. Set the criterion *LIKE 'K*'* as shown in figure 3.15. The asterisk (*) used in the expression in the *Criterion* cell of the *FirstName* field in figure 3.15 is known as a wildcard character. A wildcard is a symbol that represents any character or combination of characters. *'K*'* represents a word whose first letter is K which might be followed by any group of characters. Similarly, the criterion *LIKE '*k'* will display students whose names are ending with the letter 'k'. Similarly, '?' can be used as a wild card for a single character.

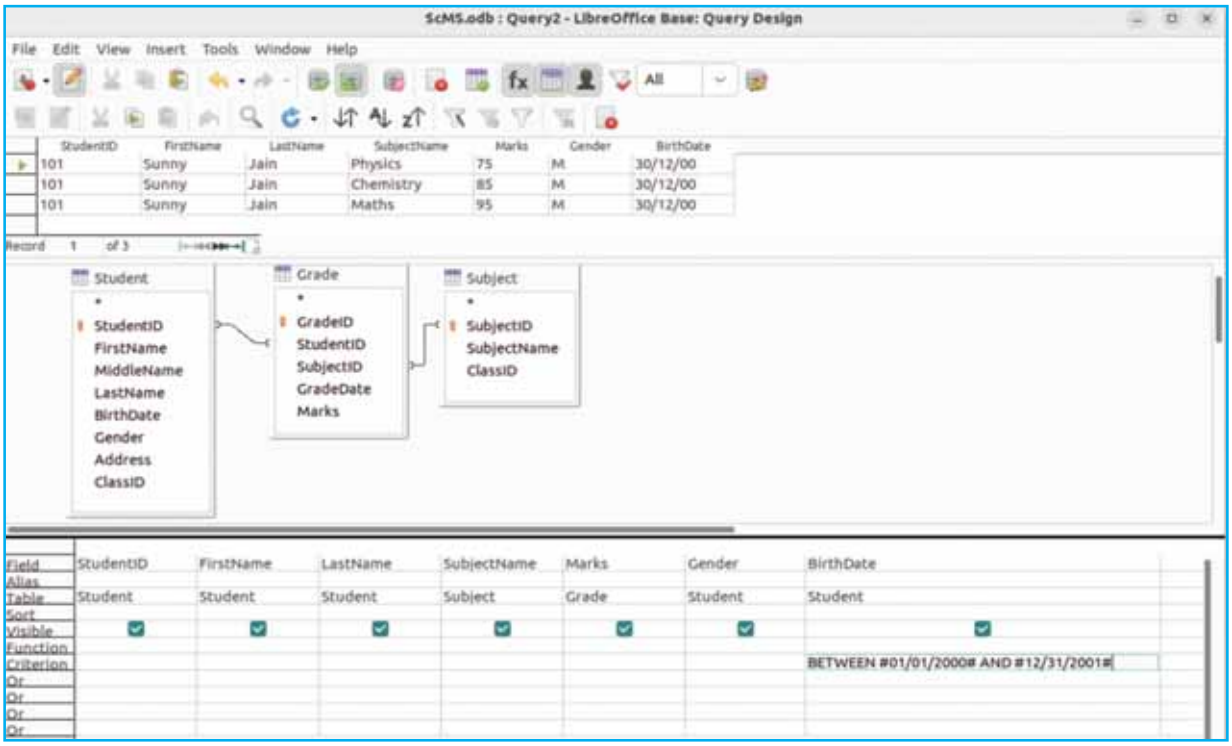


Figure 3.14 : Using the Between Operator

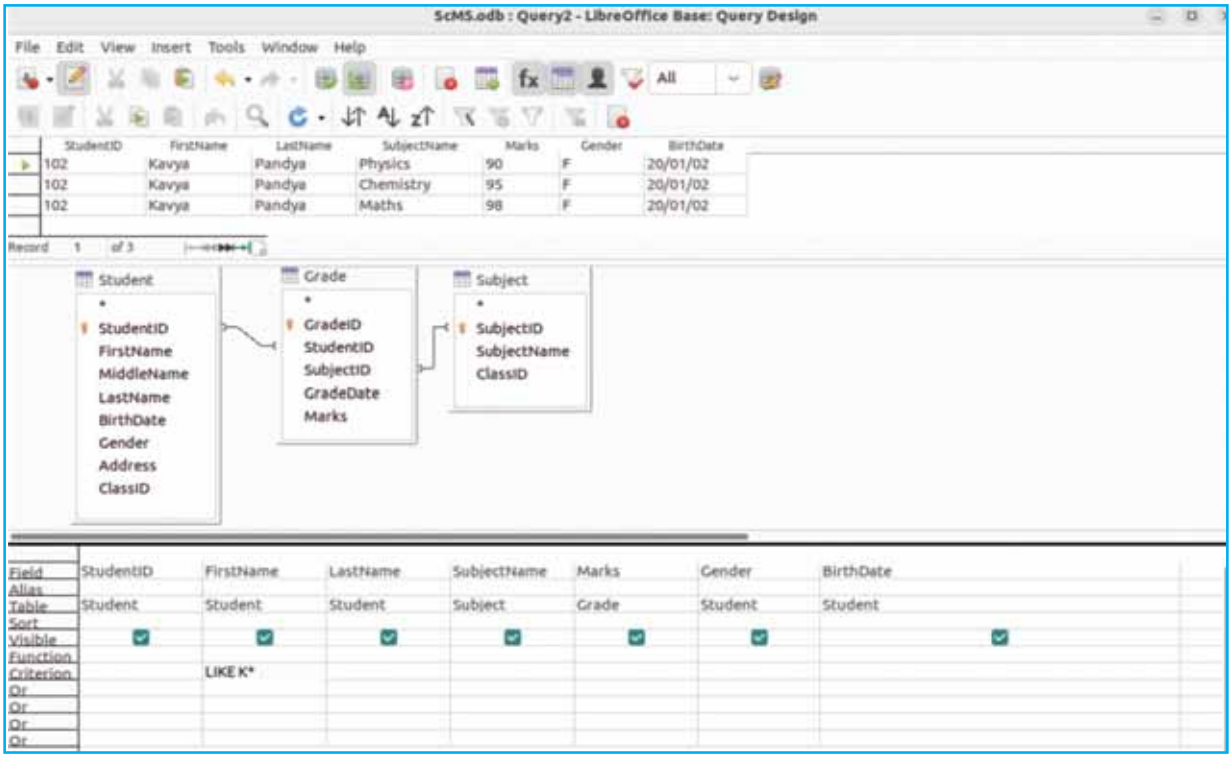


Figure 3.15 : Wild Cards

Queries with Parameters

So far, the queries that we created used fixed criteria. The criterion once defined will not be changed for every execution of the query. The output of the query may, though, vary depending on the current data in the table.

Parameter queries are designed to accept values from the user at run time. Generally, when we run a parameter query, it will display a dialog box asking the user to enter the values of the parameter. These values are then assigned as criterion values for retrieving the data.

Let us design a query to display the details of students available in the *Student* table. The following steps, when used, will give us the desired result.

- Open a new query in Design View.
- Add the Student tables.
- Type *:Address* (or [Address]) in the Criterion cell of the Address field.

The query will appear as shown in figure 3.16. Note that the criterion parameter must be preceded by a colon symbol (:).

- Click on the *Run* button to view query results. *Base* will display a dialog box as shown in figure 3.17.
- Type *'Goa'* in the text box under the label *Value* and click on the *OK* button. We will get the list of students residing in *Goa*, as shown in the figure 3.18.

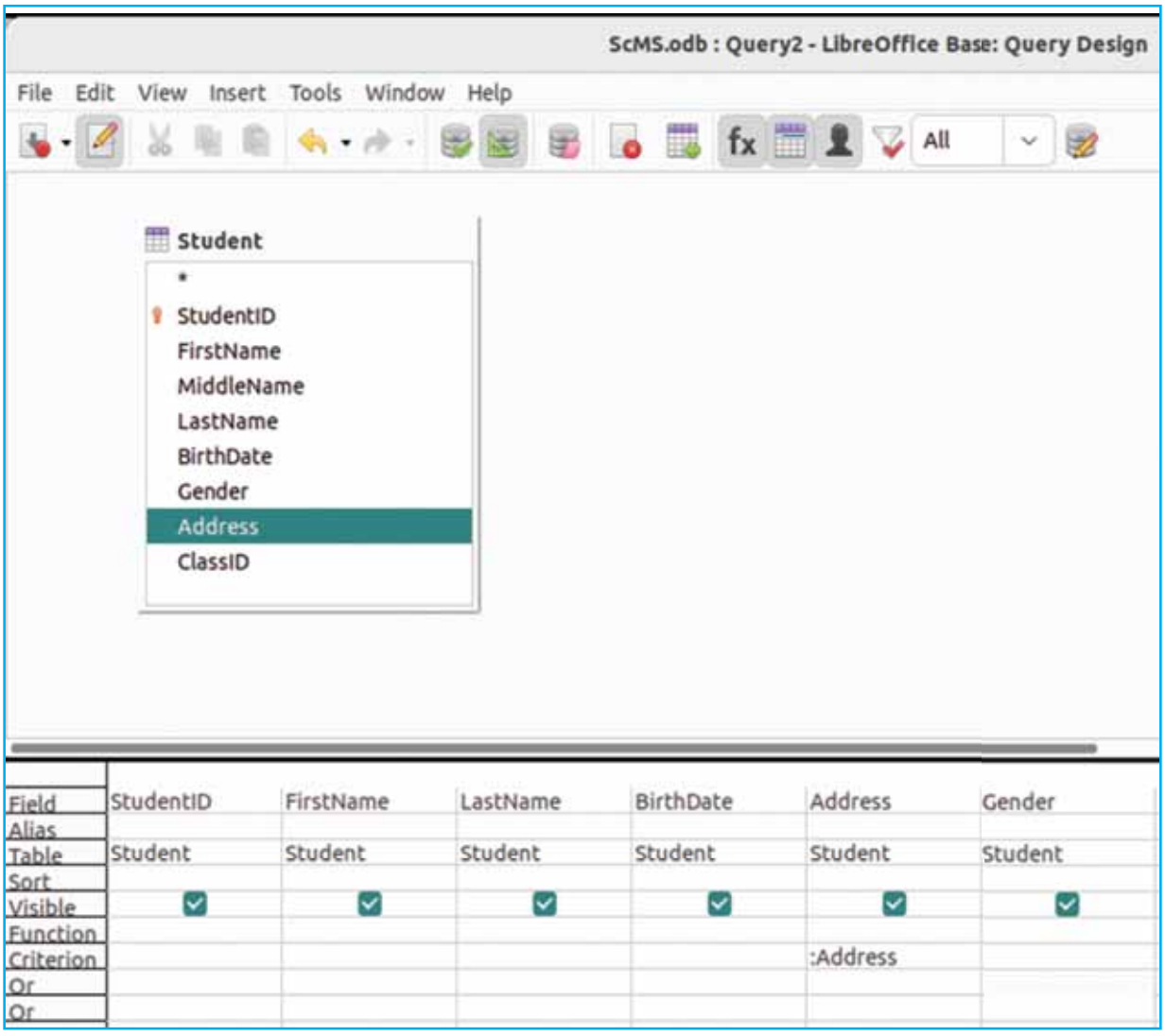


Figure 3.16 : Parameterized Query

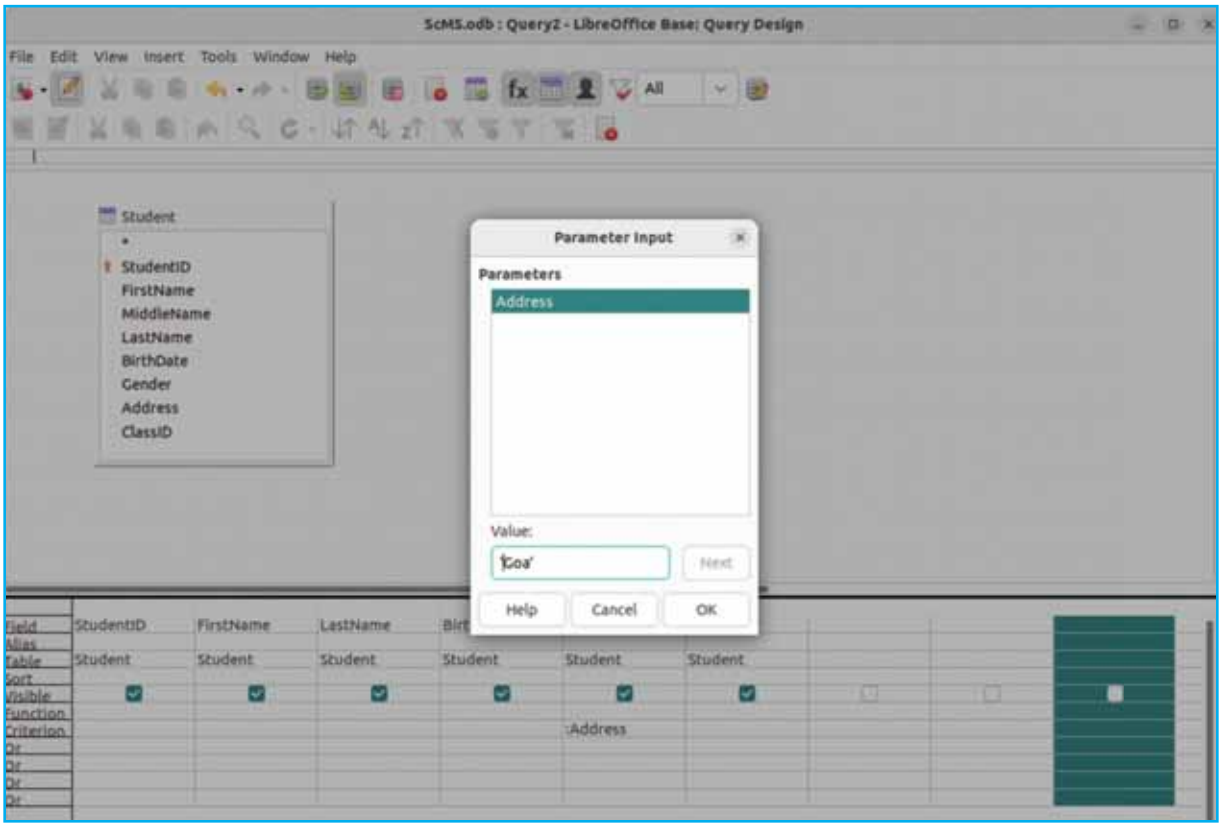


Figure 3.17 : Parameter Value

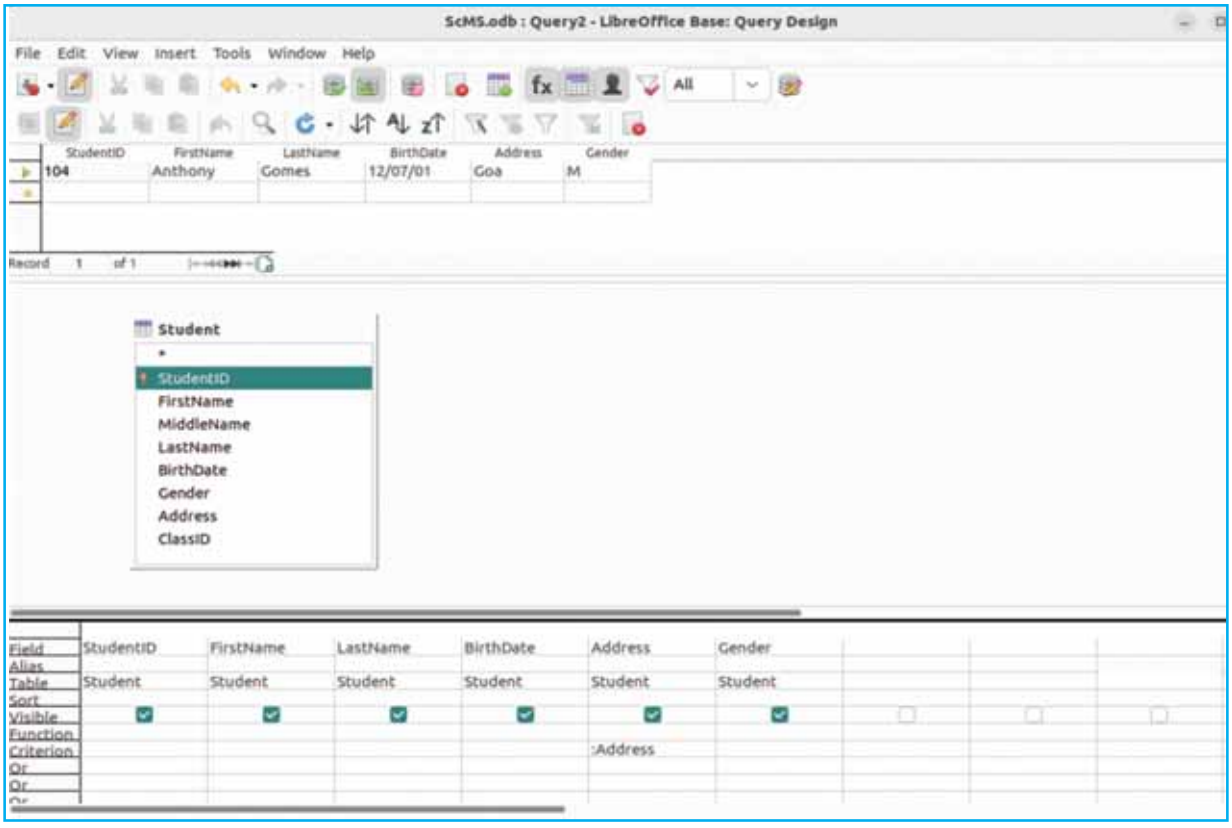


Figure 3.18 : Output

Summary

A database system provides a mechanism to store huge amounts of data. To retrieve the data useful to the end user from the database system is a challenging task. Database query allows a user to retrieve meaningful information from the database. *SQL* is a language to retrieve data from the database tables. Using *SQL* we can also retrieve data from the multiple tables. LibreOffice Base provides a very user-friendly interface to write database queries. Query wizard view provides a step by step guided approach to create a database query. On the other hand, the design view provides an efficient mechanism to write more complex queries, through which multiple tables can be combined to retrieve useful information.

EXERCISE

1. Explain the process of retrieving specific records from a table using a query in LibreOffice Base.

2. Describe how data can be retrieved from multiple tables?

3. Explain how to create a query using Design View.

4. What is the purpose of a query in LibreOffice Base?

5. Differentiate between a simple query and a parameter query.

6. Which view in LibreOffice Base provides a GUI to visually design a query?

7. Write an SQL statement to select all fields from a table called Students.

8. What is the output of this query?

SELECT "Name", "Age" FROM "Employees" WHERE "Age" > 30;

9. What is the use of wildcards in query?

10. What is the use of criteria in writing a query?

11. State whether true or false.

(1) Queries in LibreOffice Base can only retrieve data from a single table.

(2) We can use SQL statements in LibreOffice Base to retrieve data.

(3) Parameter queries allow users to input values at runtime to filter data.

(4) Wildcard characters like % can be used in queries for pattern matching during data retrieval.

(5) Queries in LibreOffice Base can be saved and reused multiple times.

12. Fill-in the blanks.

(1) A ……... is used to extract specific information from one or more tables in LibreOffice Base.

(2) The ……... view in LibreOffice Base allows users to build queries visually without writing SQL code.

(3) In a parameter query, user input is prompted using ……... brackets.

(4) The wildcard character ……... is used in SQL to match any sequence of characters.

(5) The results of a query can be displayed in a tabular format similar to a ……...

13. Multi-choice questions. Choose the most correct answer.

(1) Which object in LibreOffice Base is used to retrieve specific information from a database?

(a) Table (b) Form (c) Query (d) Report

(2) What does the * symbol in a SQL query represent?

(a) All rows (b) All columns (c) All tables (d) Primary key

(3) Which of the following views in LibreOffice Base allows graphical creation of queries?

(a) SQL View (b) Form View (c) Design View (d) Table View

(4) In a parameter query, what is used to prompt the user for input?

(a) Curly braces {} (b) Parentheses ()

(c) Square brackets [] (d) Quotes " "

(5) Which of the following is used for data retrieval in LibreOffice Base?

(a) Form (b) Query (c) Relation (d) Table Design

(6) Which wildcard character matches any sequence of characters in Libreoffice Base?

(a) – (b) # (c) * (d) ?

(7) Which SQL command is used to retrieve data from a database?

(a) UPDATE (b) INSERT (c) DELETE (d) SELECT

(8) To retrieve names starting with the letter 'A', which of the following queries would be used?

(a) LIKE 'A_' (b) LIKE '_A%' (c) LIKE 'A*' (d) LIKE '%A'

(9) What will SELECT "City" FROM "Customers" WHERE "Country" = 'India' return?

(a) Cities from all countries

(b) Cities of customers in India

(c) Countries of all customers

(d) Names of customers in India

(10) If a query does not return any rows, what does LibreOffice Base display?

(a) An error message (b) NULL

(c) A blank result grid (d) 0

Laboratory Exercise

1. Create a database table named BookList as shown in the below table

BookID	BookTitle	Author	Genre	Ratings
1	The Hobbit	J. R. R. Talkien	Adventure	4
2	The Adventures of Sherlock Holmes	Sir Arthur Conan Doyle	Suspense	5
3	Oliver Twist	Charles Dickens	Drama	3
4	adventures of huckleberry finn	Mark Twain	Children Fiction	4

2. Using the query wizard, write a query to retrieve the names of the book sorted in the alphabetical order from the table BookList created in exercise 1.
3. Using Design View, create a query to display a list of books for which ratings is at least 4.
4. Create a query with parameter "Genre" so that users can search books specific to a genre.

Working with Forms and Reports

Introduction

We covered database design and fundamental table operations using Table Datasheet View, as well as information retrieval via Query View in previous chapter. While effective, the standard row and column format of Datasheet View, with its spreadsheet-like data entry, can be visually unappealing and monotonous for users. This often 'black and white' presentation can make data interaction feel unengaging and tedious.We often use initials or abbreviated names when designing tables due to database management system naming conventions. However, these shorthand field names can sometimes lack clarity and are not always self-explanatory.

This chapter introduces an alternative and more user-friendly approach to entering and displaying database information: Forms and Reports. We will explore how to present your data in a formatted and intuitive manner using reports. Just like tables and queries, forms and reports are objects that appear in the left pane of the Database Window.

Forms

You have likely encountered many forms, like school admission forms, order forms or inquiry forms. These demonstrate how forms make it easy to collect data from users. In a database, a *Form* is an element that lets us interact with our data stored in table. Imagine it as a user-friendly window into our database, making it much simpler to view, enter, edit and delete the data. Instead of directly manipulating tables, forms present data in a more organized, visually appealing and intuitive way. Fields are often labelled clearly and the layout can be designed to mirror real-world documents or processes. In database terms, a form acts as the front-end for data entry and editing. You can design forms with different styles, colours, headings, names and even logos, making the process quite engaging.

Most forms are built upon one or more underlying tables. When we enter or edit data in a form, the changes are reflected in the corresponding table(s). Forms can also be based on queries. This is useful when we want to display, enter, or edit a subset of data or data combined from multiple tables. There are two primary ways to create a form:Using *Form Wizard* and using *Design view*.

Using a Form Wizard to create forms is both easier and quicker. The wizard automates much of the creation process, cutting down on manual effort by guiding you through a series of step-by-step questions. Let us create a form using the *Form Wizard*, based on a table we have already built. Open LibreOffice Base and then open the database we created in the second chapter. Now follow these steps to create a form based on Student table:

- Right click on *Student* table and click *Form Wizard* from the context menu as shown in figure 4.1. This will open a *Form Wizard* which offers eight steps to create a form.

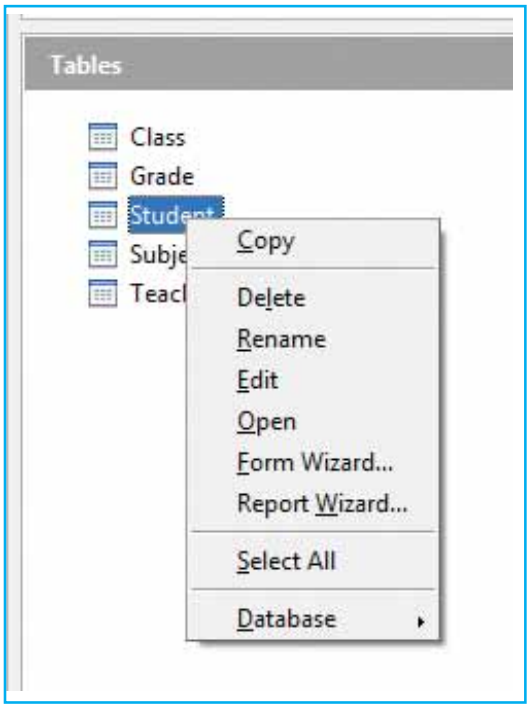


Figure 4.1 : Context Menu with Form Wizard

- Alternatively, we can open the *Form* section from left panel of Base interface and click *Use Wizard to create a Form* link as shown in figure 4.2.

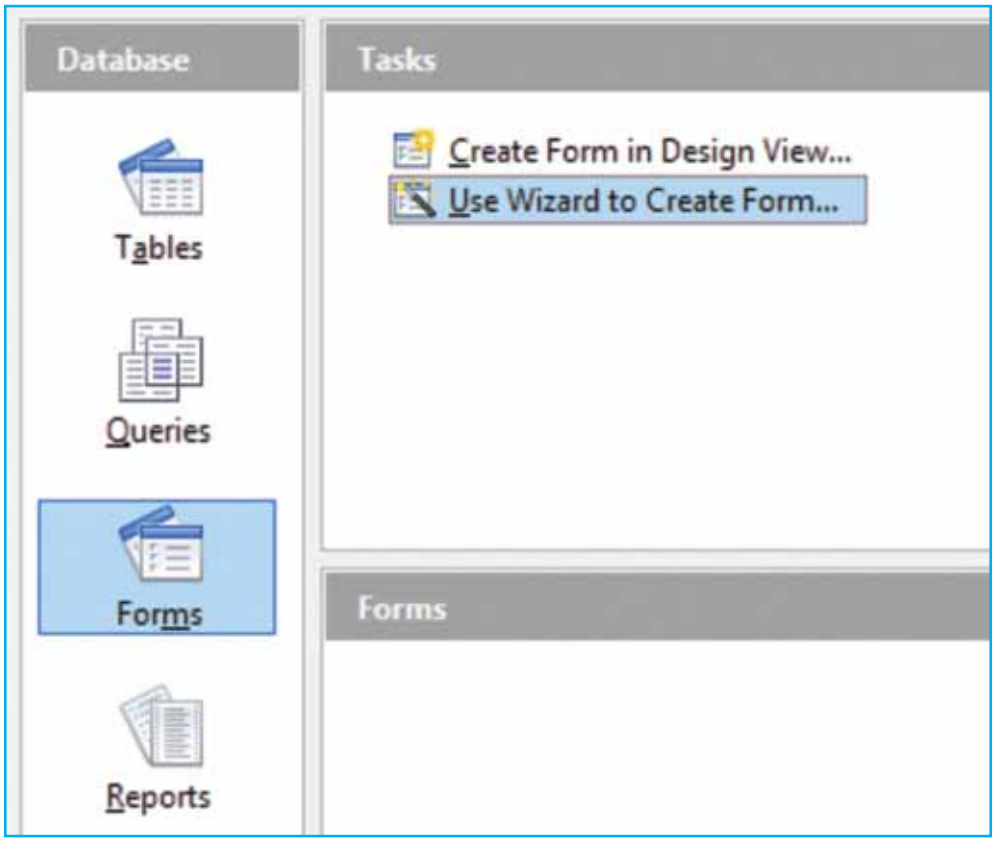


Figure 4.2 : Tasks Section Showing Form Wizard

- This action will open the *Form Wizard* as shown in figure 4.3, which guides us through eight steps to create our form.

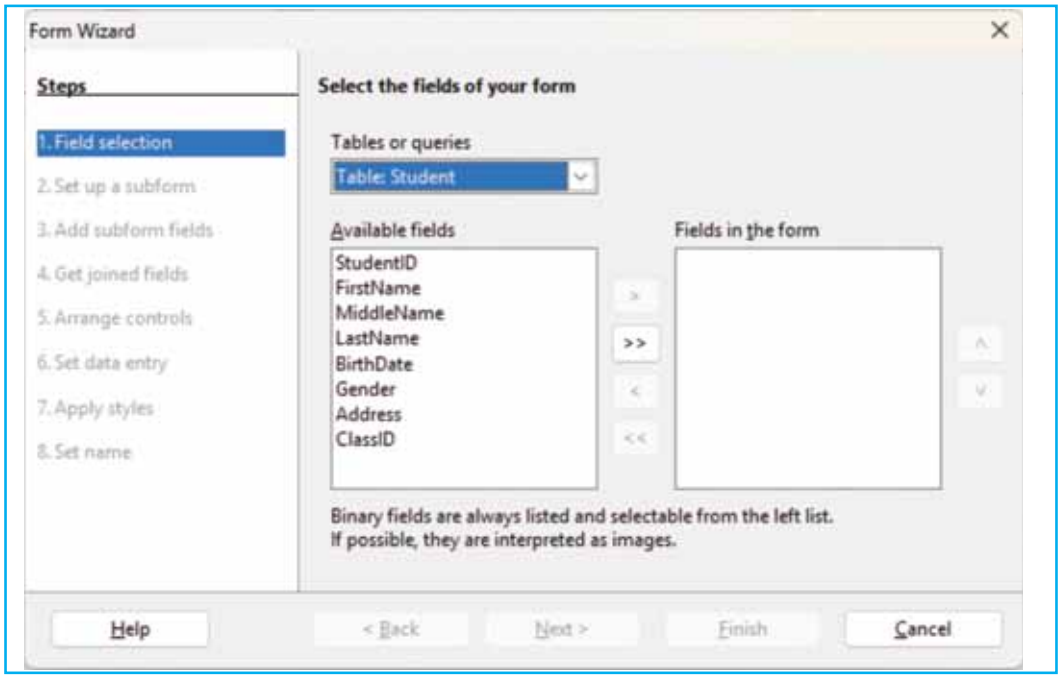


Figure 4.3 : First Step of Form Wizard – Selecting Table and Fields for Form

- First, we will need to choose the table we want to base our form on. From the *Tables or queries* dropdown menu, select the *Student* table. Once we do, all the fields from the *Student* table will appear in the *Available Fields* list.
- To add a specific field to our form, simply select its name in the *Available Fields* list and click the > button. This moves the selected field to the *Fields in the form* list.
- If we want to include all the fields from the table, we can click the >> button.
- To remove a single selected field from our form, use the < button and to remove all fields from the *Fields in the form* list, click the << button.
- We can also customize the order of our fields. Use the *Up* and *Down* arrow buttons to rearrange the fields in the *Fields in the form* list to our preferred sequence.

Once we have selected our table and fields, click the *Next* button to move on to the second step of the Form Wizard.

- The second step of the *Form Wizard* lets you create a sub form, but we will skip that for now. Just click *Next* to proceed directly to the fifth step as shown in figure 4.4.

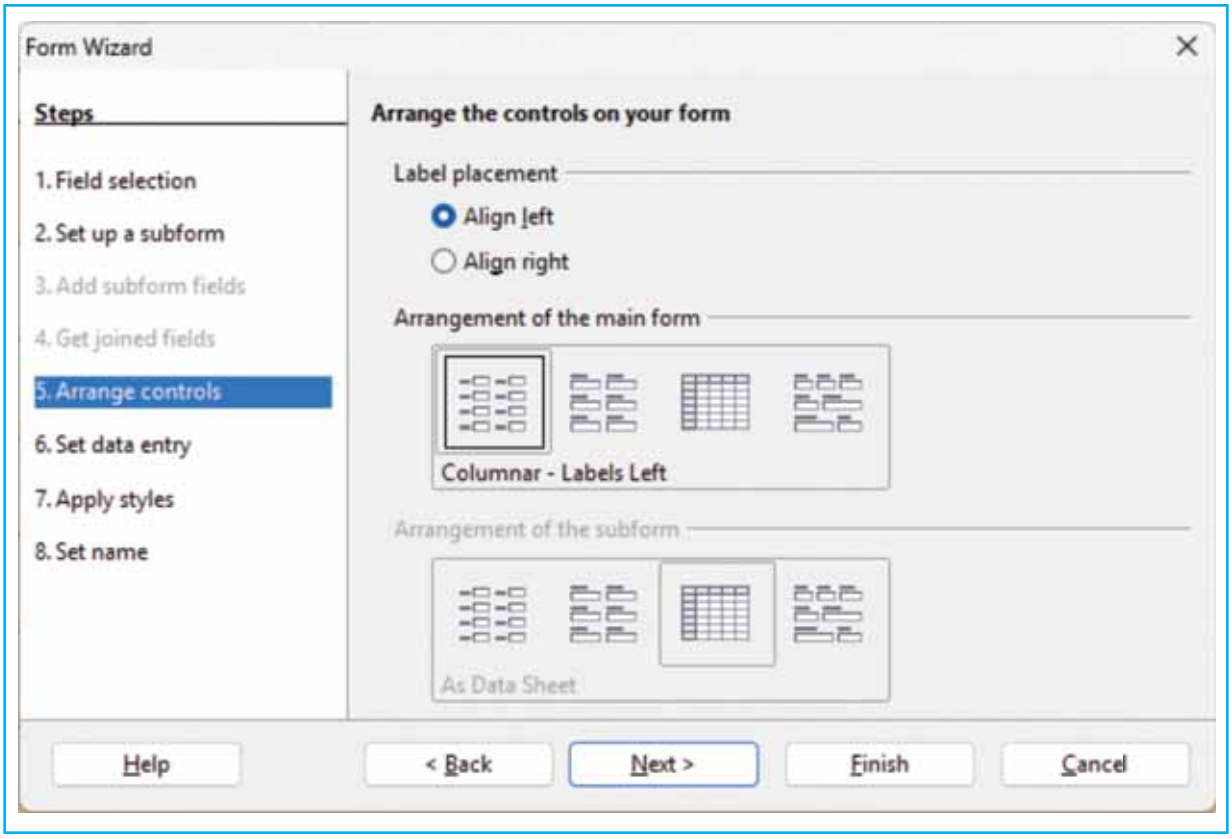


Figure 4.4 : Fifth step of Form Wizard – Arranging Controls in Form

- In the fifth step of the *Form Wizard,* we primarily focused on choosing the layout or arrangement of your main form. This step presents with different options for how our selected fields will be displayed on the form. The common layouts we will typically see are:

- **Columnar:** This is a very common choice for data entry forms. Each field appears on its own line, with the field label to the left and the input box to the right. This layout is excellent for displaying one record at a time, making it easy for users to focus on individual entries.
- **Tabular**: This layout arranges our fields in a row, like a spreadsheet. Labels usually appear as column headings, and each record occupies a new row. This is useful when you want to view or enter multiple records simultaneously, much like working directly in a table.
- **Datasheet**: This option creates a form that looks and behaves exactly like a table in Datasheet view. It displays multiple records at once in a grid format, allowing for quick data entry and review.
- **Justified**: This layout attempts to fill the available space on the form by arranging fields in a more compact multi-column format. Labels might appear above the text boxes, and fields are packed together.

- Pick your preferred layout from the options, and we will see it instantly applied to our form elements in the background. Click *Next* to proceed to the sixth step as shown in figure 4.5.

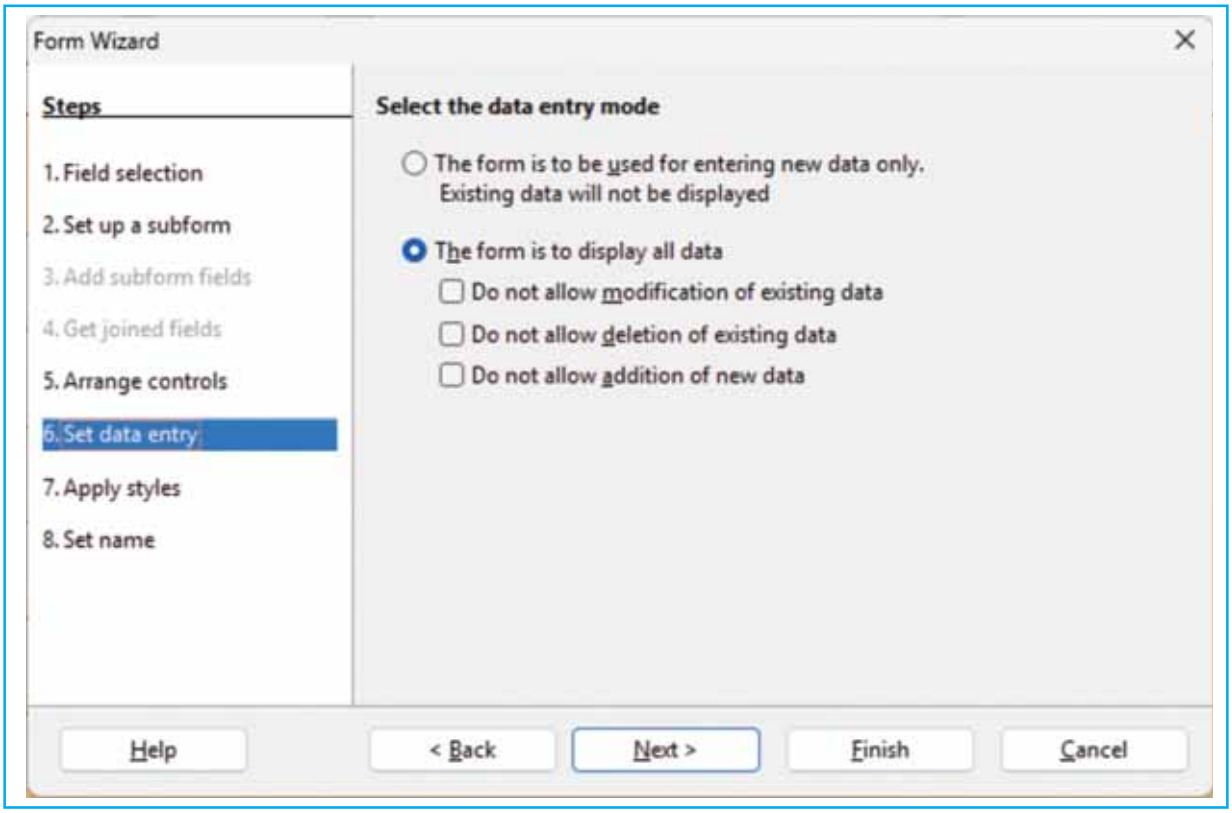


Figure 4.5 : Sixth step of Form Wizard - Setting Data Entry Mode

- The *Set Data Entry* step is where we define the behaviour and permissions for the data interaction within the form we are creating. This step decides how users will be able to view, add, modify and delete records through this form. Set Data Entry step offers two options:

- **The form is to be used for entering new data only. Existing data will not be displayed:** This creates a form that is exclusively for adding new records. Any existing data will not be shown.
- **The form is to display all data:** This creates a more general purpose form that allows users to interact with both existing and new data. Additionally, we can use some checkboxes that allow us to fine-tune what users can do with the existing data.

- Leave the default settings as they are and click *Next* to move to the next step of the *Form Wizard* as shown in figure 4.6.

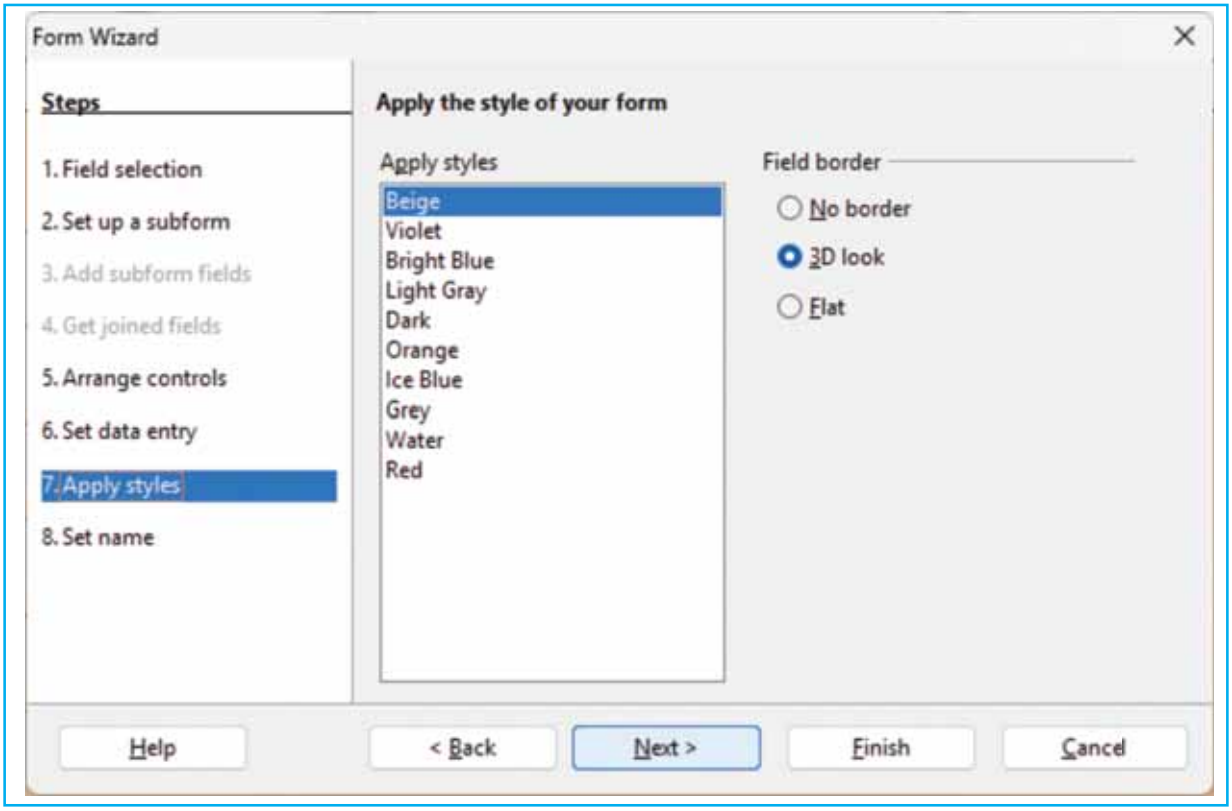


Figure : 4.6 : Seventh step of Form Wizard – Applying Styles to the Form

- This step is used to modify the visual presentation of our form. This is where we get to customize the look and feel of our form, making it more visually appealing and user-friendly. This offers mainly two options:
  - **Apply Styles:** This provides a selection of pre-defined colour options for our form.
  - **Field Border:** This option controls the appearance of the borders around our data entry fields. Following choices are available:
    - **No Border:** The fields appear without any visible outline.
    - **3D Look:** This gives the fields a raised or sunken appearance.
    - **Flat:** The fields have a simple, flat border.
- Choose your preferred options, then click *Next* to proceed to the final step of the *Form Wizard* as shown in figure 4.7.

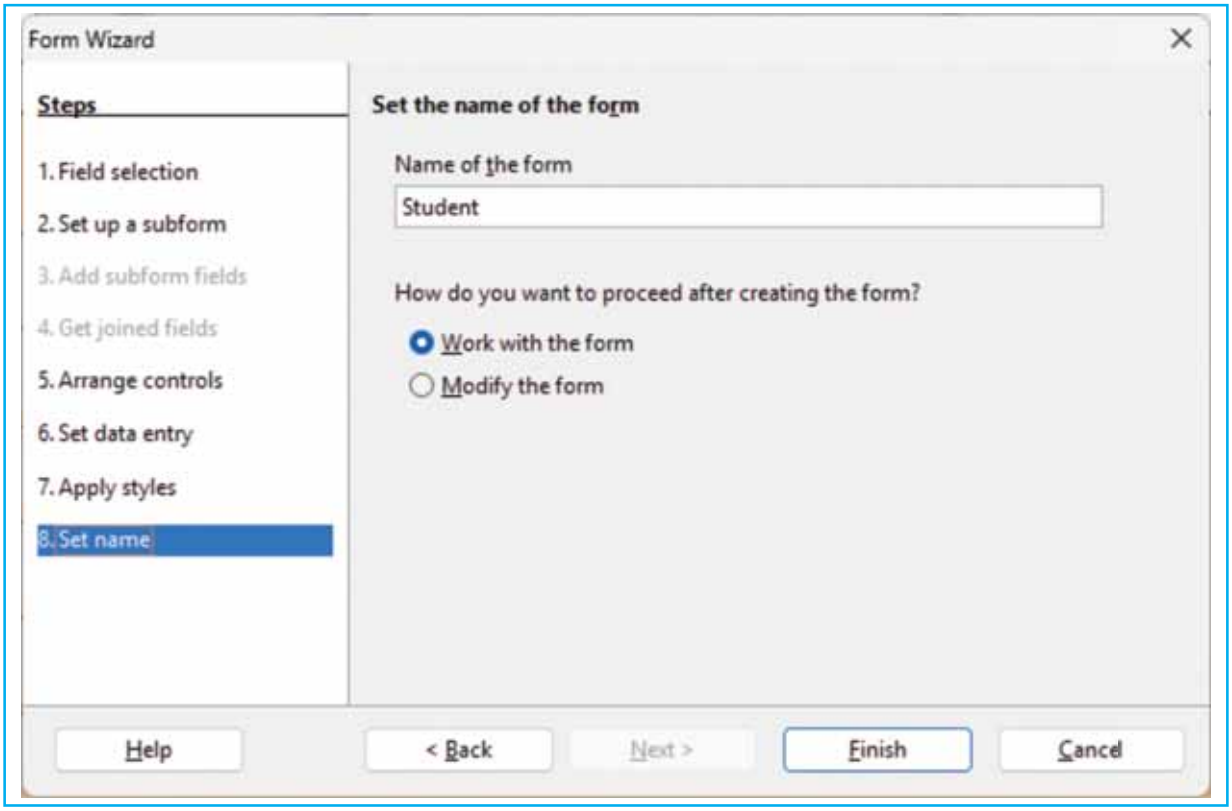


Figure 4.7 : Eighth step of Form Wizard – Setting a Name of Form

- The *Set name* step is the final stage of the *Form Wizard.* This is where we finalize our form creation by giving it a unique identifier and deciding what we want to do immediately after the wizard closes.
- This step also offers following two options:
  - **Work with the form:** If we select this, the wizard will close, and our newly created form will open immediately in *Form View*.
  - **Modify the form:** If we select this, the wizard will close, and our form will open in *Design View*.

A *FormView* for student table is shown in figure 4.8.

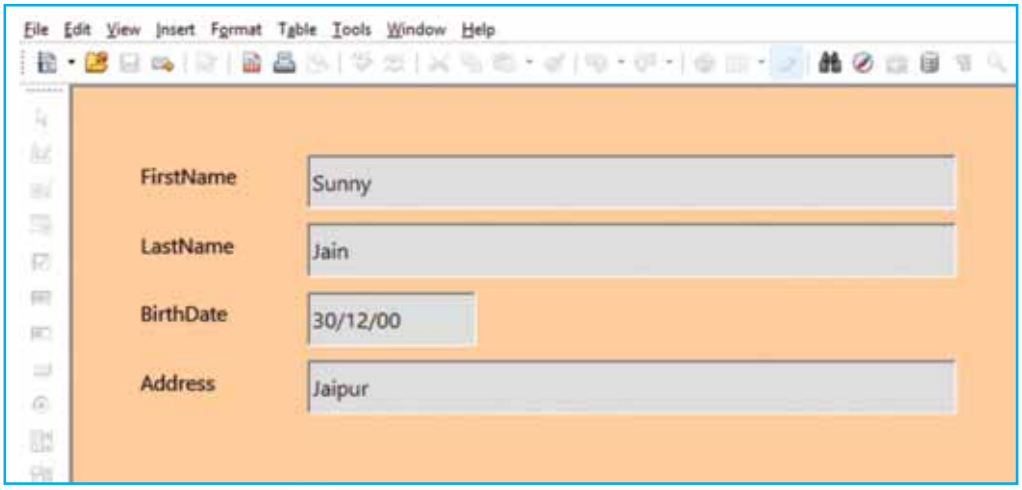


Figure 4.8 : Form View for Student table

Modifying Form

Once our form is created, we can easily modify its design to adjust the various parameters of its controls. To modify our form's design, right-click on the form name (Student) and select the *Edit* option. This will open the form in *Design View* as shown in figure 4.9.

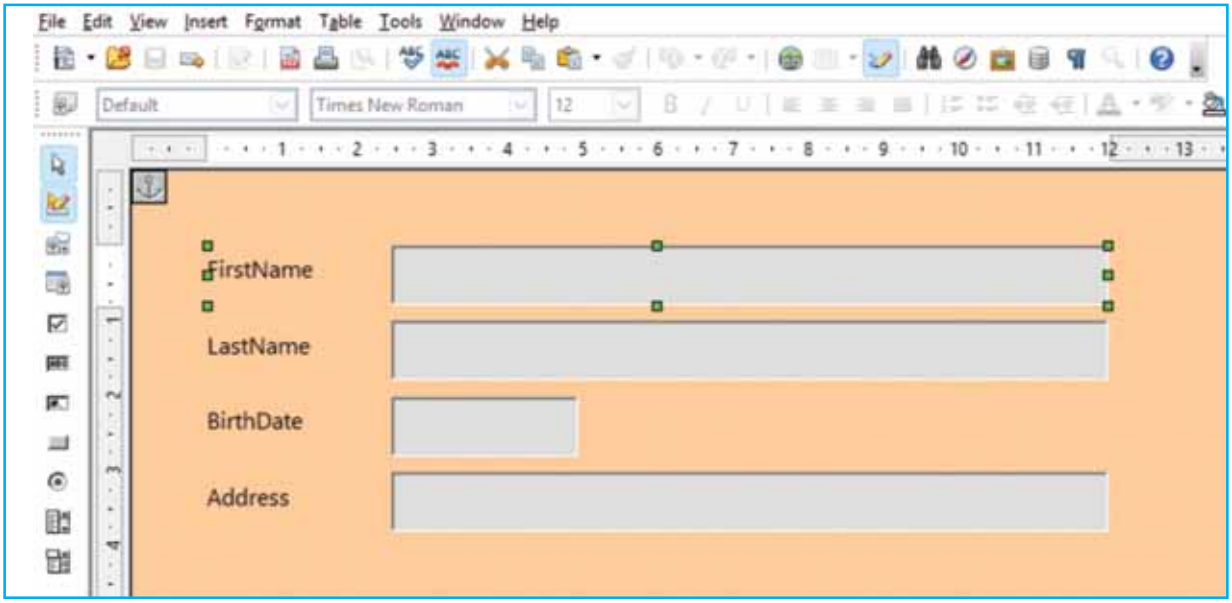


Figure 4.9 : Form Design View

Selecting Label and Textbox

In *Form Design View*, a control is typically composed of two parts: a Label and a Textbox. You can select a control by simply clicking on it. Once selected, small green squares, also known as edit points, will appear around the control as shown in figure 4.10.

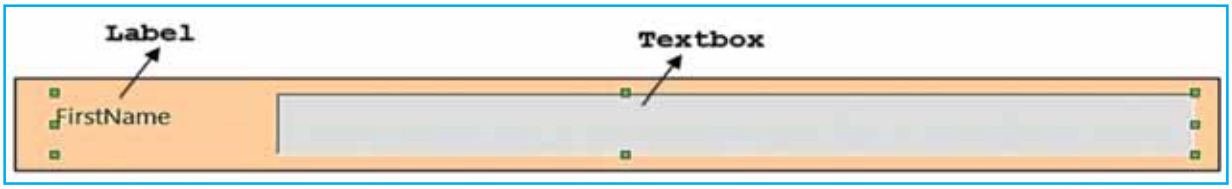


Figure 4.10 : Selecting Control in Form Design View

To select either the label or the text box independently, hold down the Control key while clicking on the desired individual element. This action will select just the label or just the text box, as depicted in figure 4.11.

Figure 4.11 : Selecting Individual Element of Control

Once we have selected an individual element of a control (like the label or text box), we can easily move or resize it. To move the element, simply drag it. To resize, drag the edit points (the small green squares) around that specific element.

To modify a control's properties in *Form Design View*, begin by selecting the specific text box. We can achieve this by holding down the Control key and clicking on the textbox. Then, right-click on the selected textbox and choose the *Control...* option from the context menu that appears. This action will open the *Properties* dialog box as shown in figure 4.12, which displays various configurable properties for that text box.

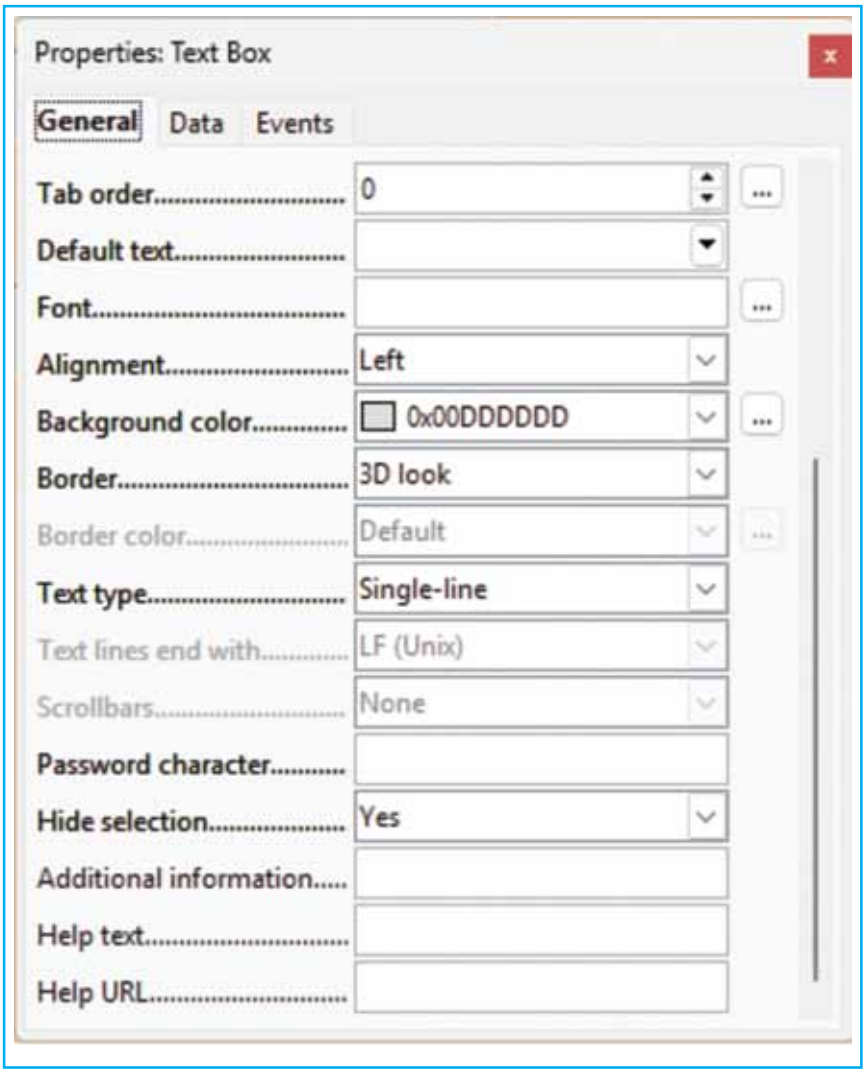


Figure 4.12 : Properties Dialog Box of Text box

Let us go ahead and change a property for this text box. Scroll down through the list of properties until you find the *Help Text* property. In the text box next to it, type "*Enter first name of student.*" This property adds a handy tooltip to the control.

Once we are done, close the *Properties* dialog box, save form design, and then open the form. When we hover mouse pointer over the *FirstName* field's text box, we will see the tooltip appear as shown in figure 4.13.

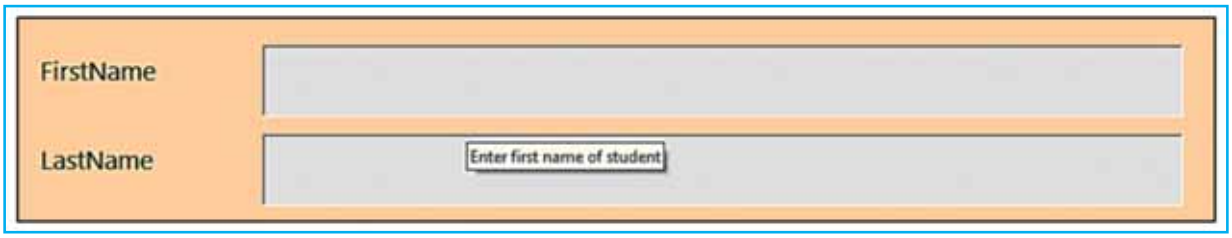


Figure 4.13 : Help Text Property Showing the Tooltip for Text box

Now, we will change the another property of form element. Select and right click on *BirthDate* text box and click *Control…* option to open *Properties* of *BirthDate* field as shown in figure 4.14. Change the value of *Dropdown* property from '*no*' to '*yes*'.

This will represent the calendar to the user while entering the birthdate of student in Form view. Close the *Property* dialog box, save the design view of form and open the form to check the data entry of date in *BirthDate* field. Output of this property is shown in figure 4.15.

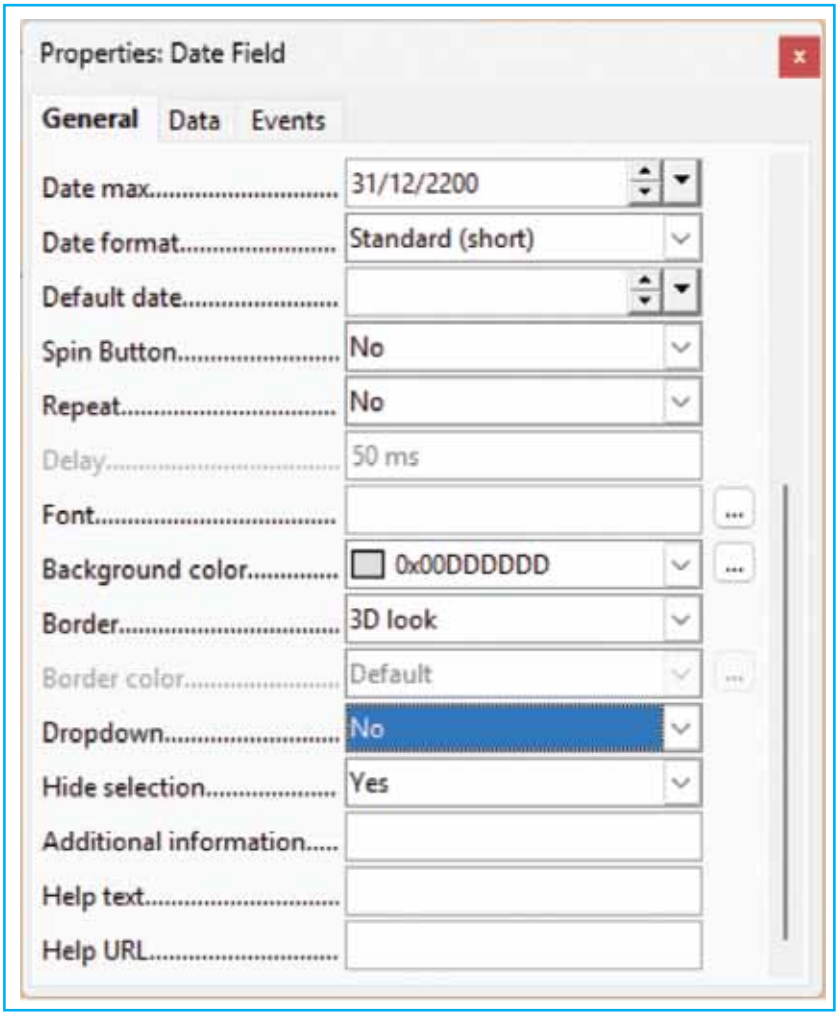


Figure 4.14 : Dropdown Property of Birthdate Field

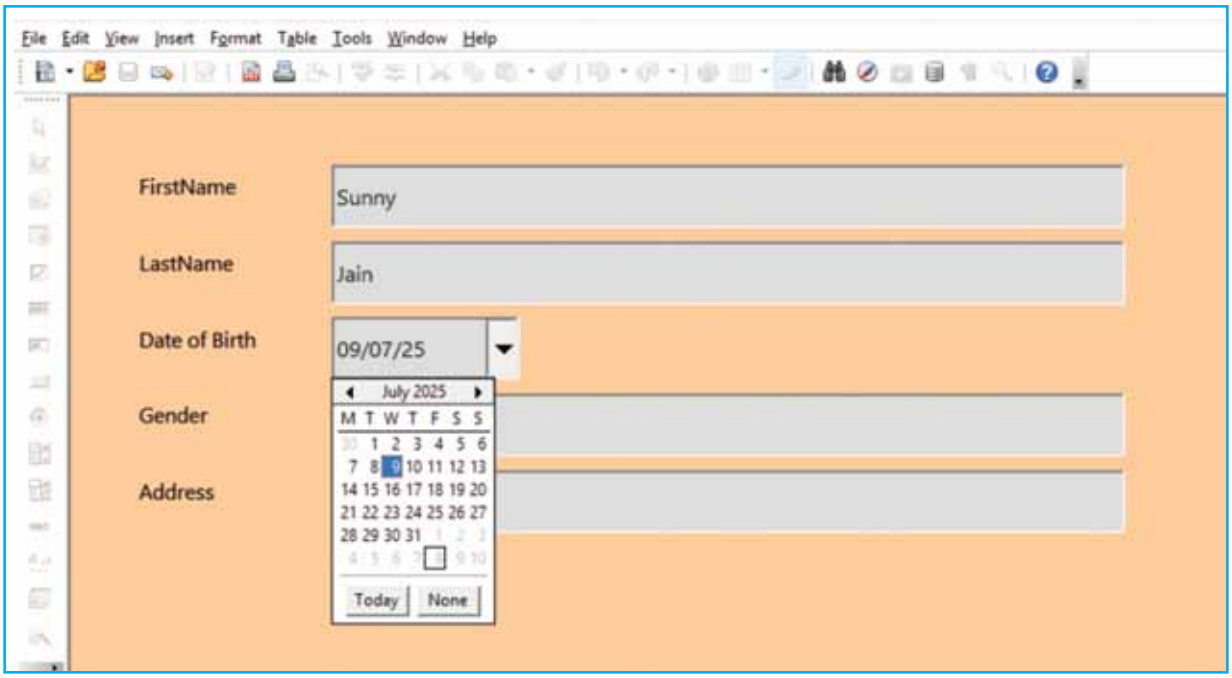


Figure 4.15 : Birthdate Field Showing Calendar

We can also change the text of a label, which is typically the field name displayed in the label box. To do this for the *BirthDate* label, right-click on it and open its *Properties* dialog box by selecting the *Control...* option.

In the *Properties* dialog box, locate the *Label* property and replace *BirthDate* with *Date of Birth* as shown in figure 4.16. After making this change, view the output to see the corrected text as the label for the *BirthDate* field.

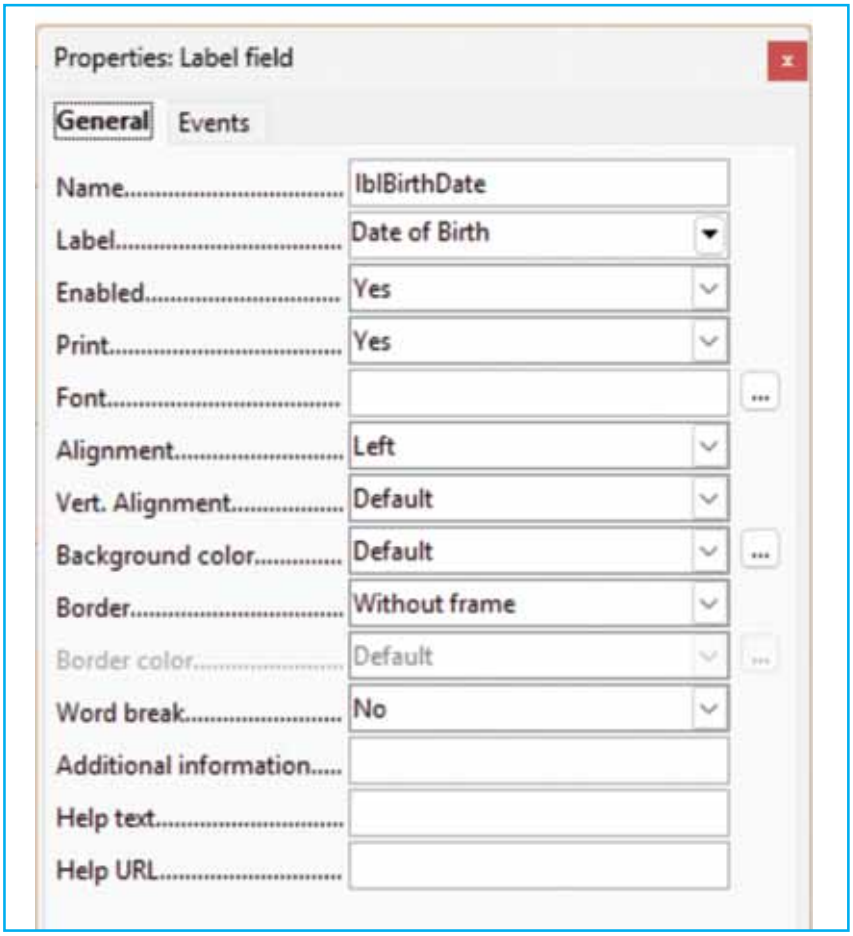


Figure 4.16 : Editing Label of Birthdate Field

We can also change the background of our form in *Form Design View*. Right-click on any blank space within the form. From the context menu, select *Page*. This will open the *Page Style* dialog box as shown in figure 4.17. Go to the *Background* tab. To apply a solid colour, simply choose from the various colours offered and click *OK*. Your Form's background will update immediately.

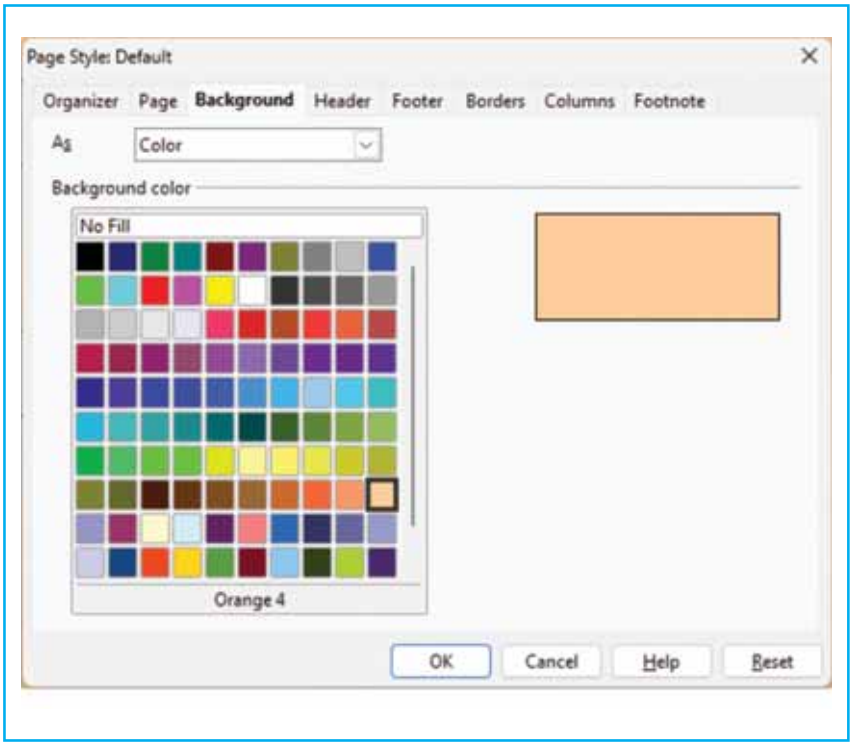


Figure 4.17 : Changing Background Colour of Form

To use an image as form background, select *Graphic* in the *As* dropdown menu. Click the *Browse* button and choose the desired image file. The form will then display selected image as the background.

With teacher's guidance, we may explore and modify more field properties in the *Properties* dialog box by ourselves.

Inserting and Deleting Records using Form

As we know, forms are primarily used for entering data into tables. When we open a form in *Form View*, we will see a navigation bar at the bottom of our screen as shown in figure 4.18. This bar has buttons that let us easily move through the records in our table.

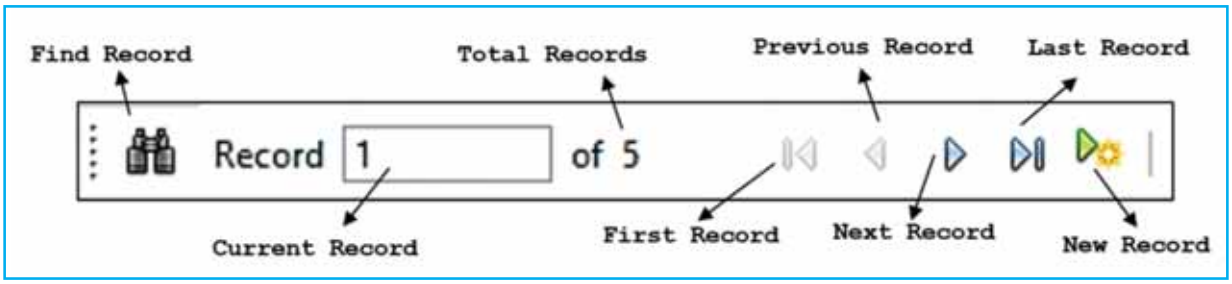


Figure 4.18 : Navigation Bar in Form View

To add a new record to our table using a form, just click the *New Record* button on the navigation bar. A blank record will appear, and we can then enter the values into the field text boxes.

To delete a record from our table, first locate the record we want to remove. Then, simply click the *Delete Record* button, which is located next to the navigation bar. A warning message will pop up asking you to confirm the deletion. Clicking *Yes* will permanently remove the record from the table.

Searching Record using Form

Often, we will need to locate a particular record within our form. Fortunately, Base includes a built-in search feature to help you find records directly from our table.

Here are the steps to search for a record in your form:

- Click the *Find Record* button on the navigation bar. This will open the *Record Search* dialog box as shown in figure 4.19.
- Enter the search term in the *Search for* box.
- In the *Where to search* section, we have two options: To search a specific field, select *Single field* and choose the field name from the dropdown menu. To search all fields in the table, select *All Fields*.
- We can also specify where the search term should appear within the record by using the *Position* dropdown menu.
- Click the *Search* button to begin. Base will then display any record containing the search term.

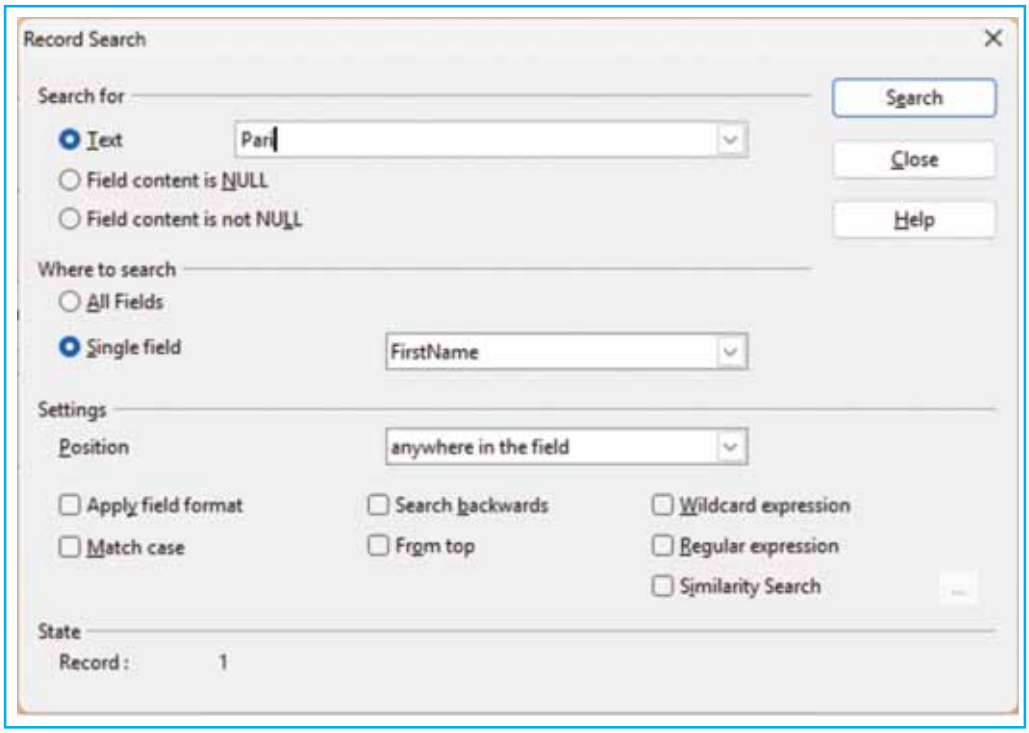


Figure 4.19 : Searching for a Record

Reports

Reports are an excellent way to present retrieved information in an attractive and organized format, often with the goal of creating a hard copy. The design of a report usually prioritizes how it will look when printed.

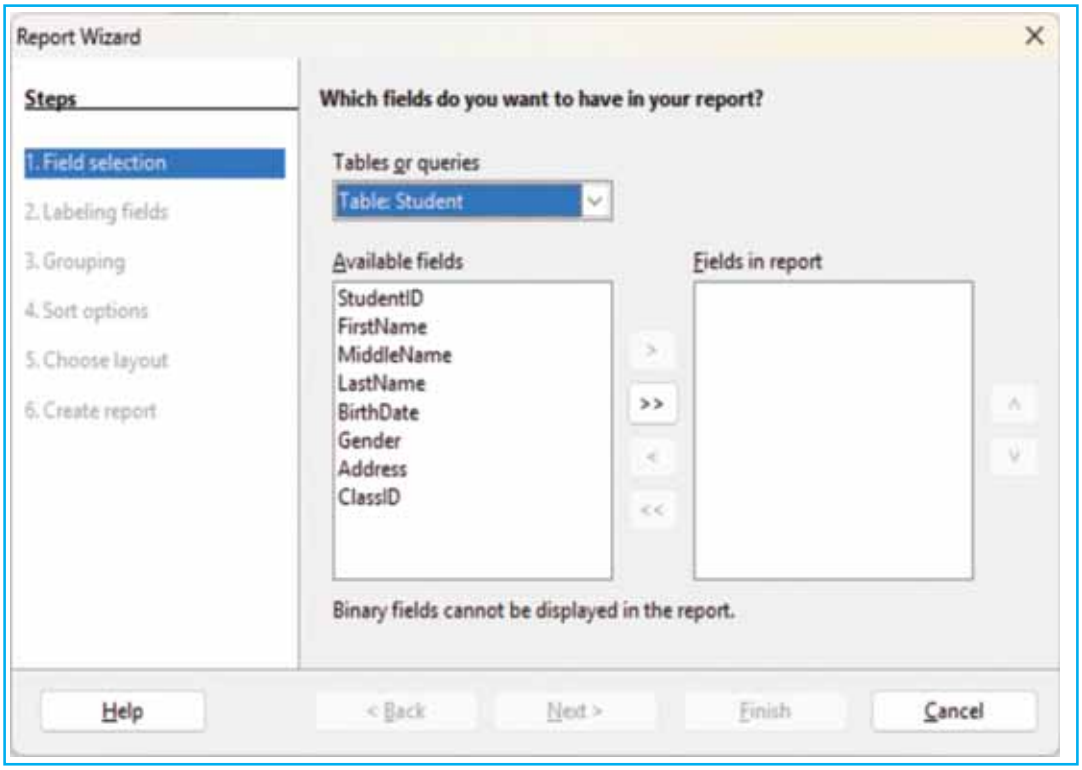


Figure 4.20 : Report Wizard

We can create reports using a query, a table or a combination of both. If our report needs to include fields from various tables, it is a good practice to first create a query that brings those fields together. Then, we can build our report using that combined query.

Let us create a report using *Student* table. First, click the *Reports* icon in the left-hand panel. We will then see the option *Use Wizard to Create Report*…. Click this, and the *Report Wizard* will appear as shown in figure 4.20, guiding us through six distinct steps to prepare our report.

To begin creating our report, the first step in the *Report Wizard* is to select the data source. From the *Tables or Queries* list, choose the *Student* table. After selecting the table, we will need to pick the specific fields we want to include in our report. Use the > button to move these fields from the available list to our report.

Just like in the Form Wizard, we have several options for managing the fields in our report. The >> button lets us quickly add all available fields to our report. To remove a selected field, simply use the < button. If we want to remove all fields from report at once, use the << button.

After configuring the field selections, click the *Next* button to proceed to the *Labeling Fields* step of the *Report Wizard* as shown in figure 4.21.

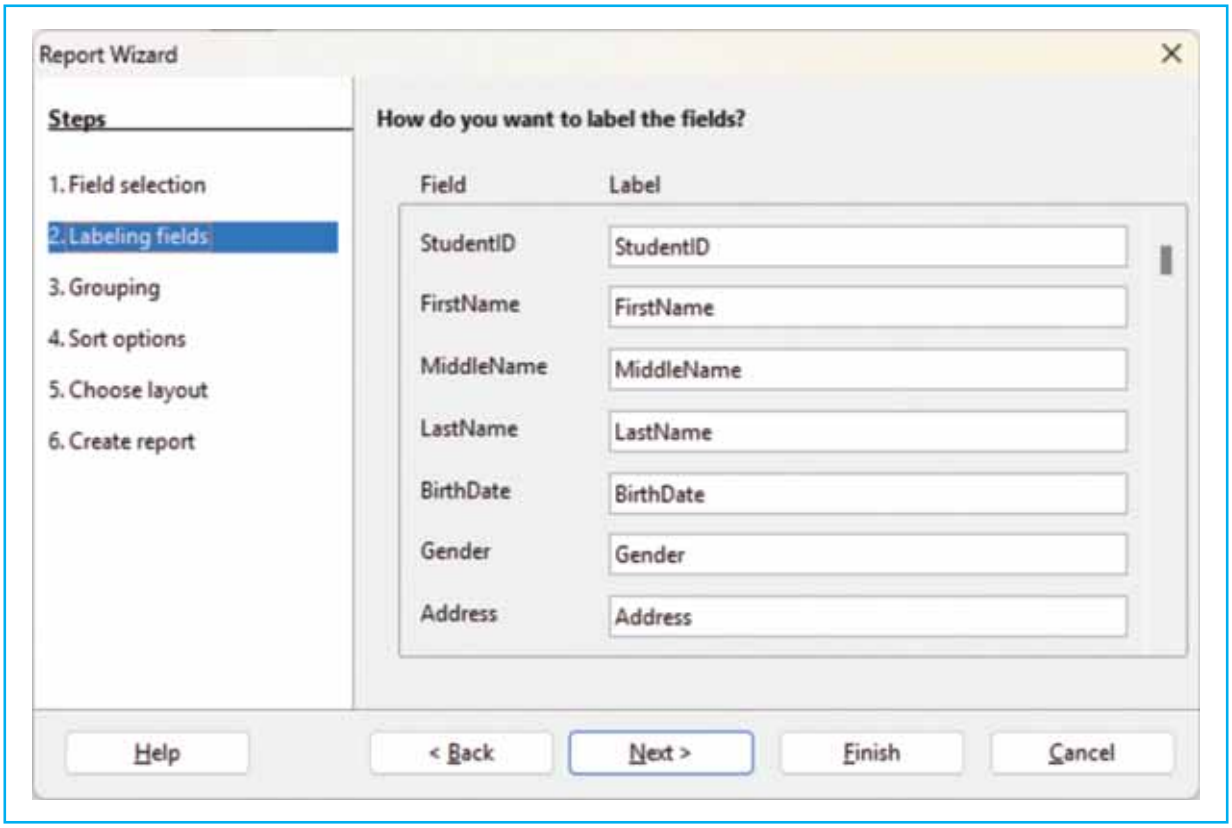


Figure 4.21 : Second step of Report Wizard – Labeling Fields

This step allows us to customize how those fields will appear as labels in the final report. By default, Base will use the exact field names from the table or query as the labels in the report. This step gives us the opportunity to make them more user-friendly and readable. We can add spaces, change capitalization, or use more descriptive terms for our labels.

Click *Next* to proceed to the *Grouping* step of the *Report Wizard* as shown in figure 4.22.

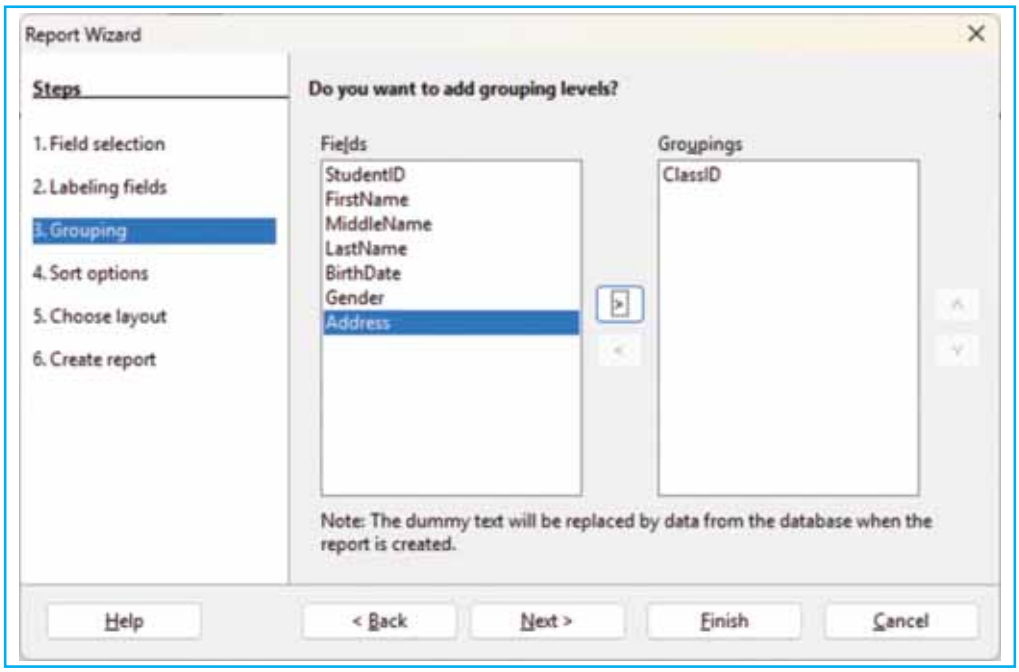


Figure 4.22 : Third step of Report Wizard – Grouping

Grouping involves collecting rows of data that have the same value in a specific field and presenting them together. When we group data, the report will display a header for each unique value in the

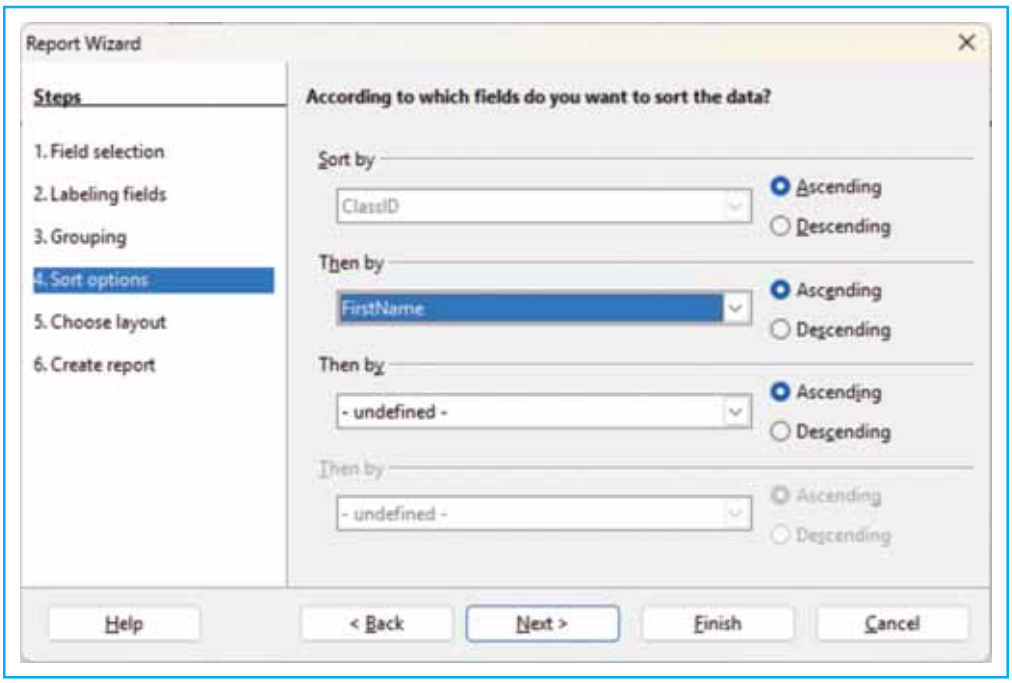


Figure 4.23 : Forth step of Report Wizard – Sort options

chosen grouping field, followed by all the detail records that belong to that group. Choose one or more fields from the available list and move them to a *Grouping* list. Here, the *ClassID* field has been added as a grouping field.

Click *Next* to go to the next step of *Report Wizard* → *Sort Options* as shown in figure 4.23.

In this step, we define the order in which the detail records within our report, and specifically within any defined groups, will be presented. This step is crucial for making report data logically ordered and easy to follow. In the *Sort by* dropdown menu, select the *FirstName* field. Then, click *Next* to proceed to the *Choose layout* step of the *Report Wizard* as shown in figure 4.24.

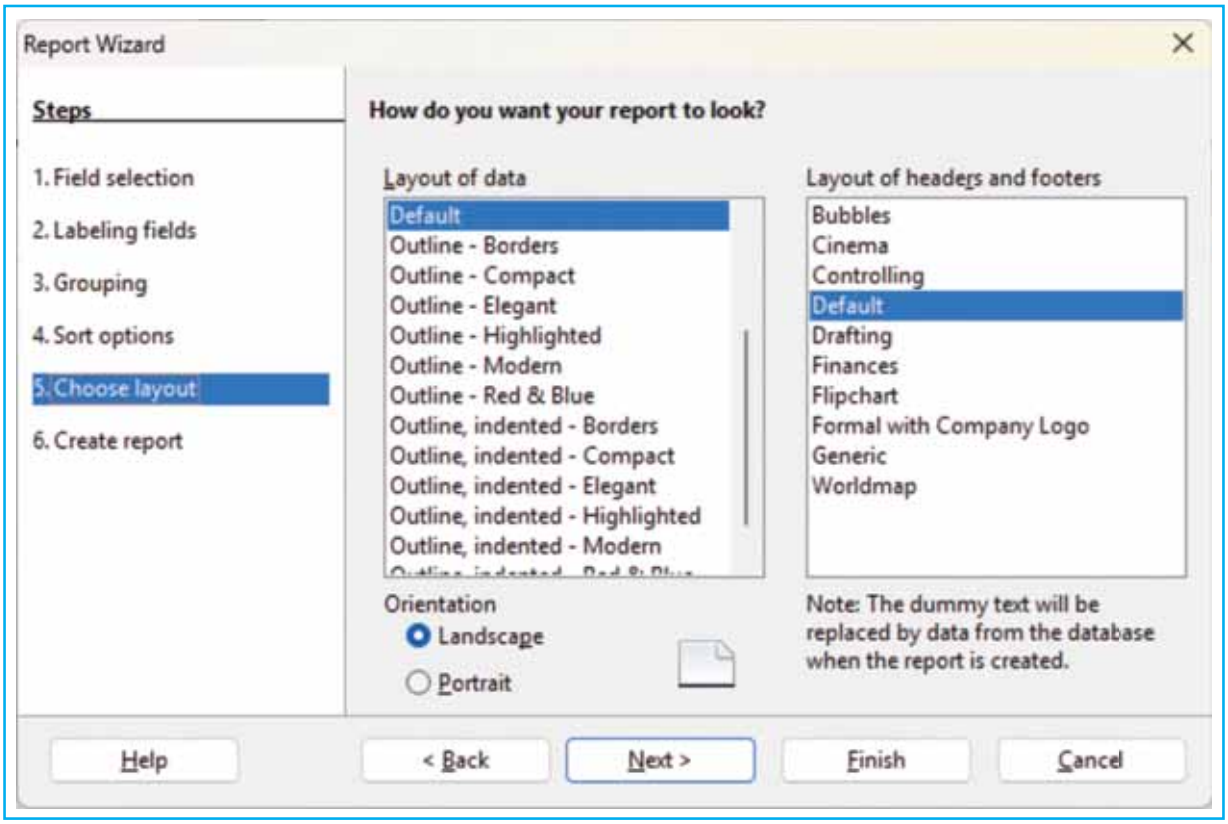


Figure 4.24 : Fifth step of Report Wizard – Choose layout

The fifth step of the *Report Wizard* lets us customize our report's layout. You can choose from various colour combinations and adjust the positioning of fields and text alignment using the *Layout of data* selection list. This step also allows to set the orientation of report to either *Landscape* or *Portrait* and select a generalized header and footer layout from the *Layout of headers and footers* list box.

Once we have chosen our desired report layout, simply click *Next* to proceed to the final step of the *Report Wizard: Create Report* as shown in figure 4.25.

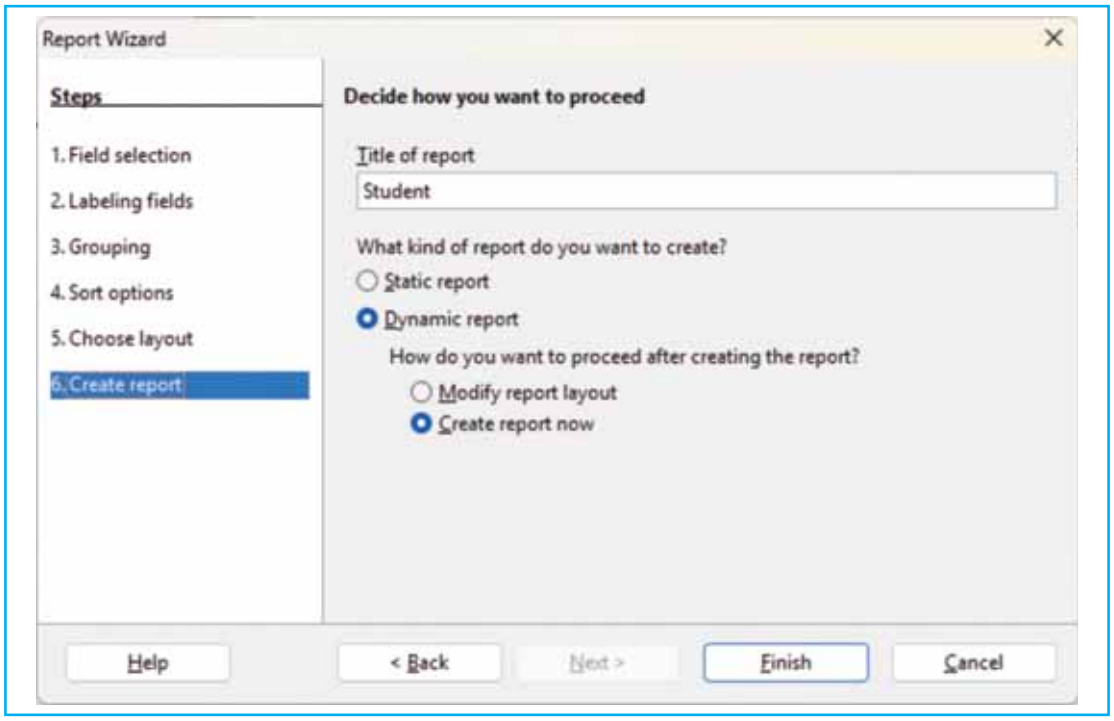
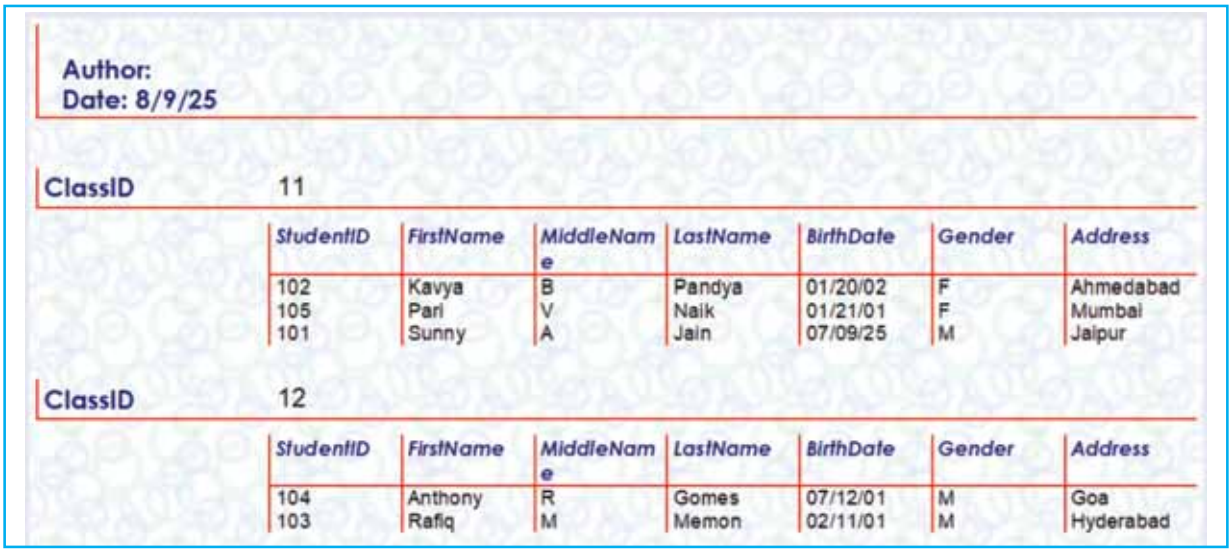


Figure 4.25 : Last step of Report Wizard : Create Report

In the last step of *Report Wizard,* we will be offered following options:

- **Title of Report**: We will need to name the report we are creating.
- **What kind of report do you want to create?**: We will be presented with two options here:
  - **Static report**: Once created, the data displayed in a static report does not change, even if the underlying tables or queries in our database are updated with new information.
  - **Dynamic report**: A dynamic report is designed to reflect the current data in our database every time it is opened or refreshed.

Click *Finish* to complete creating our report. Figure 4.26 displays a similar view of the final report.



Author:
Date: 8/9/25

ClassID 11

StudentID	FirstName	MiddleNam e	LastName	BirthDate	Gender	Address
102	Kavya	B	Pandya	01/20/02	F	Ahmedabad
105	Pari	V	Naik	01/21/01	F	Mumbai
101	Sunny	A	Jain	07/09/25	M	Jaipur

ClassID 12

StudentID	FirstName	MiddleNam e	LastName	BirthDate	Gender	Address
104	Anthony	R	Gomes	07/12/01	M	Goa
103	Rafiq	M	Memon	02/11/01	M	Hyderabad

Figure 4.26 : Report based on Student Table

Please note that the report groups the *Student* table data by *ClassID* and sorts it by *FirstName*. Once our report is created, we can easily modify its design. Simply right-click on the report name and select the *Edit* option. This will open the *Report Design View*, where we can make changes in a similar way to how we would modify a form.

We can obtain a hardcopy of our report by printing it via *File*→*Print* option. Alternatively, to create a PDF copy, simply click the *Export directly as PDF* button on the standard toolbar.

Summary

Forms and Reports are vital to any effective database management system. They play distinct yet complementary roles, transforming raw data into accessible, usable, and valuable information for end users. In this chapter, we have learned how to create both forms and reports for our database. We saw that forms act as user interfaces, allowing direct interaction with the database without needing to manipulate tables or queries. We also prepared reports, which are essential tools for presenting and summarizing data in an organized and meaningful way, turning raw data into actionable insights crucial for decision-making. With this, we conclude our exploration of the basics of database management.

EXERCISE

1. Why are forms more convenient for interacting with database data?

2. Which form creation options are available in Base?

3. Explain any three steps of Form Wizard.

4. How can we change the background of our form?

5. How do we resize form controls?

6. What is a report, and what role does it play in data representation?

7. What are the steps to modify the Label property of a control in Form Design view?

8. What is the difference between static and dynamic reports?

9. Which buttons are displayed on the navigation bar of a form?

10. How do we search a specific record in a form?

11. State whether True or False.

(1) It is possible to modify data in Form view.

(2) We can create a form based on a table or query.

(3) It is required to create a subform when creating a form with the Form Wizard.

(4) Reports are typically read only.

(5) Reports can be exported as PDF files.

12. Fill-in the blanks.

(1) Forms can be created based on ……... or ……... .

(2) Using a ……... to create forms is both easier and quicker.

(3) Form Wizard contains total ……... steps.

(4) ……... lets us easily move through the records in our table.

(5) ……... are an excellent way to present retrieved information in an attractive and organized format, often with the goal of creating a hard copy.

13. Mutichoice questions. Choose the most correct answer.

(1) Which element enables interaction with data stored in a table?

(a) Design view (b) Relationship (c) Form (d) Report

(2) Which database object can be used to create a form?

(a) Table (b) Query (c) Report (d) Table or Query

(3) How many steps are involved in the Form Wizard?

(a) 6 (b) 7 (c) 8 (d) 10

(4) In which step of the Form Wizard do we choose the underlying table or query?

(a) First (b) Third (c) Forth (d) Eighth

(5) Which button is used to add all available fields of a table to a form?

(a) > (b) >> (c) < (d) <<

(6) Which key should be held down while selecting a control to choose either the label or textbox?

(a) Shift (b) Control (c) Alter (d) Space

(7) Which property in Form Design view is used to add a tooltip to a control?

(a) Tooltip (b) Add text

(c) Help text (d) There is no such property

(8) Which button in the Report Wizard is used to remove a selected field from the report?

(a) > (b) >> (c) < (d) <<

(9) In which step of the Report Wizard do we choose the page layout orientation?

(a) Third (b) Forth (c) Fifth (d) Sixth

(10) Which kind of report does not change even if the underlying tables are updated?

(a) Static (b) Dynamic (c) Programmatic (d) Automatic

Laboratory Exercise

1. Create report for all the tables created in previous chapters.
2. Modify the forms according to the specified requirements.:
   - Change the background of any form.
   - Insert tooltip for at least two controls in any form.
   - Change 'Date' type control of form to dropdown menu.
   - Change label of any two controls in any form.
   - Adjust the size of all controls on all forms as necessary.
3. Search for a student record by providing the student's name, birthdate, and address.
4. Generate static and dynamic reports based on two distinct tables.

Problem Solving: Flowchart and Algorithm

Introduction

From the stone age to modern technology era, human come across the wide variety of problems every day throughout the life. It includes very simple tasks like counting different items, making total of bill for purchase of different items in different quantity having different prices etc. to complex problems like managing the stock of the hundreds of items in a store of a large factory producing home-use items. People are used to with how to solve their day to day and simple problems but when it comes to large and complex problems where completion in time with accuracy is mandatory, computer comes to the help.

Computer science provides us the convenient way to solve the problems or we can say the problem solving is at the heart of the computer science. It can be done by writing a program for the task we want to handle. Once a program is written for the task, it can be used for long time without solving problem again and again. For example, performing the banking operations, selecting the item online or ordering the items online and making the payment online using UPI (Unified Payment Interface).

In this chapter, we will be focusing on understanding the concepts of problem solving and then learning two widely used methods 1) Flowchart and 2) Algorithm with number of examples. After learning the chapter students will be able to understand the given problem statement and develop the flowchart and/or writing algorithm to solve the problem with clarity.

Overview of Problem Solving

Problem solving is a step-wise solution of a problem. That means, understanding the problem completely and organizing it into smaller steps in proper order. The broader steps for problem solving can be as under:

- Define the problem with all details
- Understanding the problem
- Identifying or getting the inputs
- Providing the necessary steps to solve the problem
- Testing and validating the output

The step "Providing the necessary steps to solve the problem" varies from problem to problem. For simple problem, it includes few steps in sequence only but for moderate to large problems, it may include decisions providing alternative paths or repeating some steps again and again. Hence, this step mainly incorporates the following steps with different combinations:

- **Sequence :** one by one step
- **Decision making :** Decide to follow one of the steps from two more alternatives
- **Repetition/Looping :** Performing set of steps again and again i.e. multiple times

Let us understand them with the help of examples. The steps for adding two numbers are as follows:

1. Accept two numbers N1 and N2
2. Add them as SUM = N1 + N2
3. Print the result i.e. SUM

The above problem is purely sequential and to solve it, we have to follow the steps in sequence as 1, 2 and 3.

Following are the steps for finding larger of two numbers.

1. Accept two numbers N1 and N2
2. Compare them and if N1 > N2, then N1 is larger, jump to step-4
3. Otherwise N2 is larger
4. Print larger

Observe that above problem includes decision at step-2. After accepting two numbers N1 and N2 in step-1, step-2 first compares them with ">" and if N1 is found larger, then it skips the step-3 and goes directly to step-4 to print N1. If the comparison is false, rest of the part of step-2 is skipped as N2 is larger means step-3 is performed and then step-4 prints N2. The important thing is we choose between two alternatives based on comparison of N1 or N2.

Following are the steps for making the sum of 1 to 10.

1. Define SUM with 0
2. Define I with 1
3. Repeat step-4 and step-5, 10 times
4. Add I to the SUM
5. Increment I by 1
6. Print SUM

Carefully, observe the steps of above problem. After step-1 defines SUM with 0 and step-2 defines I with first value 1, we repeat step-4 to add current value of I to the SUM and then increment I to next value in step-5. They are repeated 10 time hence I will go from 1,2, … 10 and every time it is added to previous value of SUM. Finally, it produces SUM = 1+2+3+4+5+6+7+8+9+10 = 55 which is printed by step-6.

Above example demonstrated use of all three important elements of processing steps of the problem solving. The problems we encounter in real-life are not simple as above. They are involving many computations and are complex. To solve them, we have to use above steps i.e. sequence, decision making and repetition in different combinations depending on the requirement of the problem.

Following section will introduce you with two widely used problem solving methods namely flowchart and algorithm. We will understand how to use them to solve the problem with many examples involving use of above steps and finally, we will compare both flowcharts and algorithms.

Flowchart and its symbols

Let us understand the flowchart in detail with the different symbols used to develop the flowcharts with their meaning and application.

Flowchart

Flowchart is a pictorial representation of the solution of a problem. It shows the steps of a process or task in a simple and visual way. Flowchart uses different symbols like ovals, rectangles, diamonds to represent various actions for the various steps. Arrows are used to connect different steps. As flowchart uses visual representation, all the possible paths and steps are clearly visible and it makes the understanding of the solution very easy. Figure 5.1 shows the flowchart for multiplying two numbers.

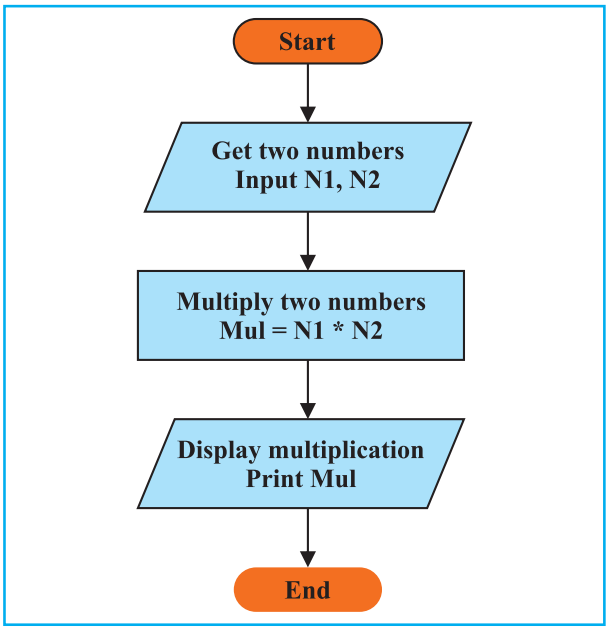


Figure 5.1 : Flowchart to Multiply Two Numbers

As shown in figure 5.1, flowchart starts with "Start" and ends with "End". Second step represented by parallelogram accepts the two numbers N1 and N2. Third step uses rectangle to show the process of multiplying them as Mul = N1 * N2. Forth step is again parallelogram for printing the output i.e. Mul that is multiplication of given two numbers. The flowchart is very simple as problem is sequential. The flow i.e. order of the steps is clearly visible as arrow after each step points to the next step. Let us understand all the symbols used by flowchart before we develop the flowchart for different problems involving decision making and looping.

Flowchart Symbols

Figure 5.2 shows the various symbols used to develop the flowcharts. Figure shows the symbol, its name and the short description for quick understanding. Let us now understand each of them one by one with their use in flowchart.

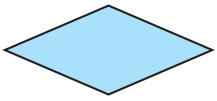
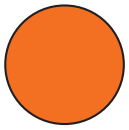
Symbol	Name	Description
	Oval	Represents start or end point
	Arrows	Connect two symbols with direction
	Parallelogram	Represents input or output
	Rectangle	Represents process, used to compute values or perform operations
	Diamond	Represents decision making, provides alternative paths
	Circle	Used as connector, used to connect different parts of flowchart

Figure 5.2 : Flowchart Symbols

Start/End: Start and End are the terminal symbols represented by oval. The Start denotes the beginning of the flowchart and is always the first symbol. The End denotes the completion or end of the flowchart and is always the last symbol. The steps for performing process or task are enclosed between Start and End symbols as shown in figure 5.1. Figure 5.3 shows the use of the Start and End symbol with arrows.

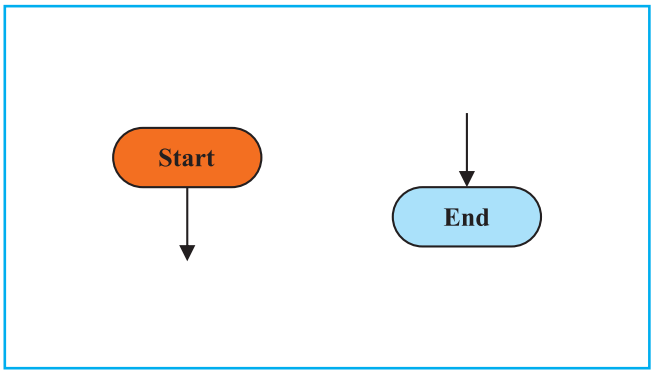


Figure 5.3 : Use of Start and End symbols

Arrow: Arrow is used to denote the flow lines. We can use arrow in all the directions depending on the where next symbol is drawn in the flowchart. Most common use is down arrow as shown in figure 5.1. Arrow is most important symbol used to decide the execution path within the flowchart or it indicates where the control moves next. However, arrow to left or right is also used whenever needed specifically in large flowcharts having many steps in a single page.

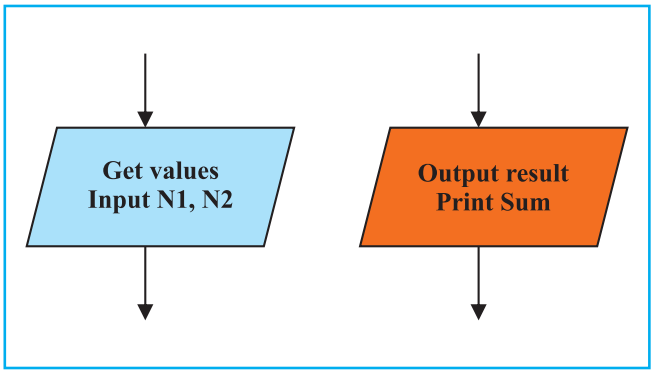


Figure 5.4 : Use of Input/Output symbols

Input/Output: Input and Output are represented by parallelogram as shown in figure 5.2. The Input symbol is used to receive the values or data from the user or real world on which further process is done. The Output symbol denotes that the result produced by the solution is sent to the user or real world for their use. Figure 5.4 shows the use of the Input and Output symbols. Input symbol is denoted with "Input N1 and N2" and Output symbol denoted by "Print Sum".

Process: It uses the rectangle. Whenever we want to compute or process some values, it is used. Process step can be used to perform variety of computations like arithmetic operations, logical operations, evaluating formulas based on given values etc. It is possible to compute more than one value using set of steps in a single process box represented by a rectangle. Figure 5.5 shows two examples of process step: one to add two numbers (Sum = N1 + N2) and second to compute area and periphery of the circle.

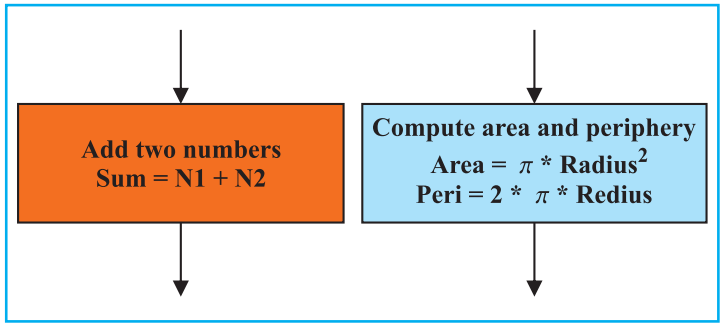


Figure 5.5 : Use of Process Symbol

Decision: Decision making is most important and performed regularly by human beings in their life. Decision uses the diamond symbol in the flowchart. Within a diamond, we normally use condition to decide one of the alternatives. Let us say, if number is greater than 0, we negate it, otherwise don't take any action. Another example compares two numbers and if first is greater than second, we take choice-1, otherwise we take choice-2. It is shown in figure 5.6.

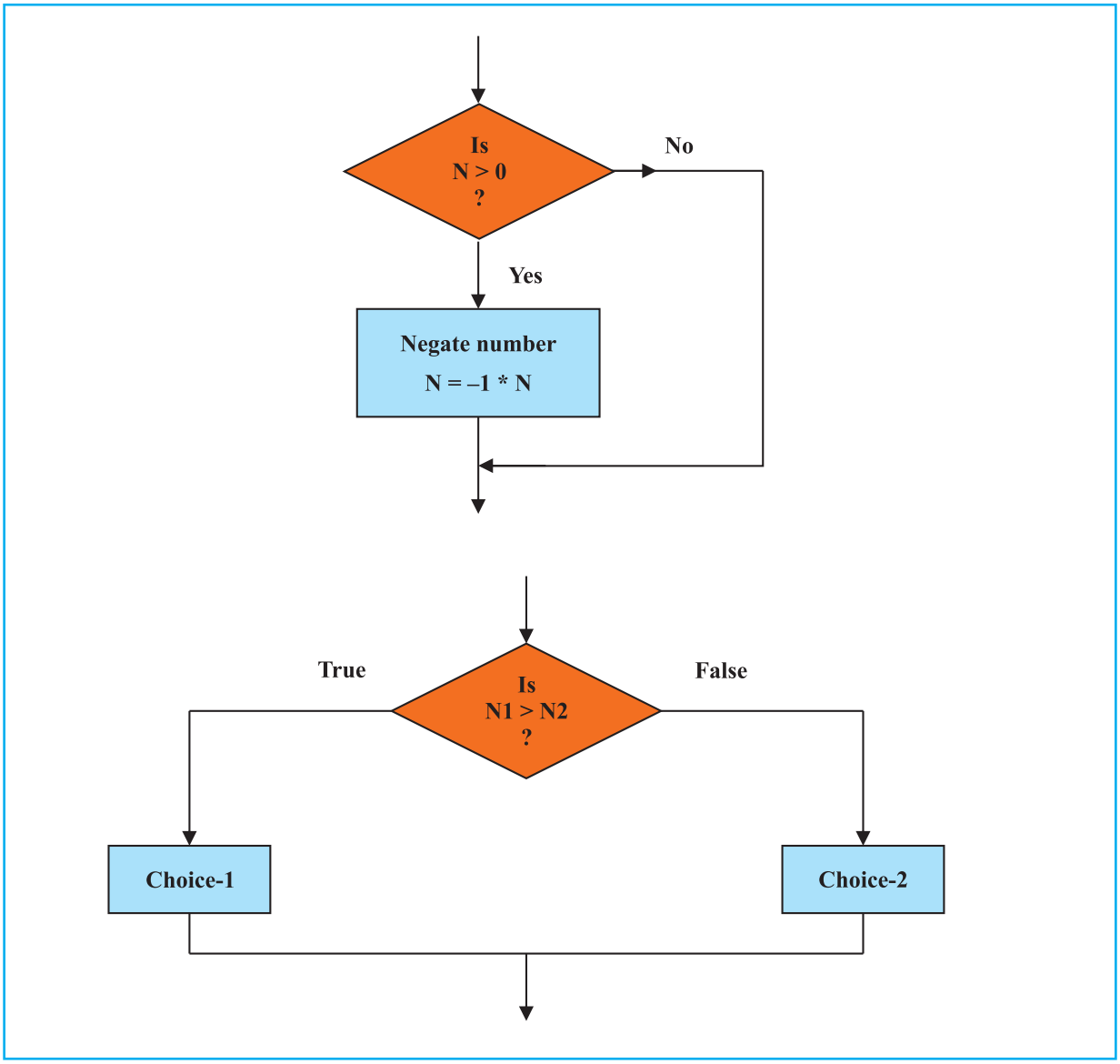


Figure 5.6 : Use of Decision Making

Sometimes, we have multiple conditions and with each of them, some statements are associated. Figure 5.7 shows such a situation. It is a multi-choice structure as we have to choose one alternative from multiple alternatives based on which one of the conditions is true. In figure 5.7, first the Condition1 is tested. If it is true, then statement1 is processed and control moves to the end of structure. If it is false, then Condition2 is evaluated. If Condition2 is true, then statement2 is processed and moves to the end, otherwise it moves to the Condition3. It is repeated until all the conditions are tested. If all the conditions are false, then Default-statement is processed and completes. In short, only one of the statements will execute depending on which condition is true or Default-statement executes.

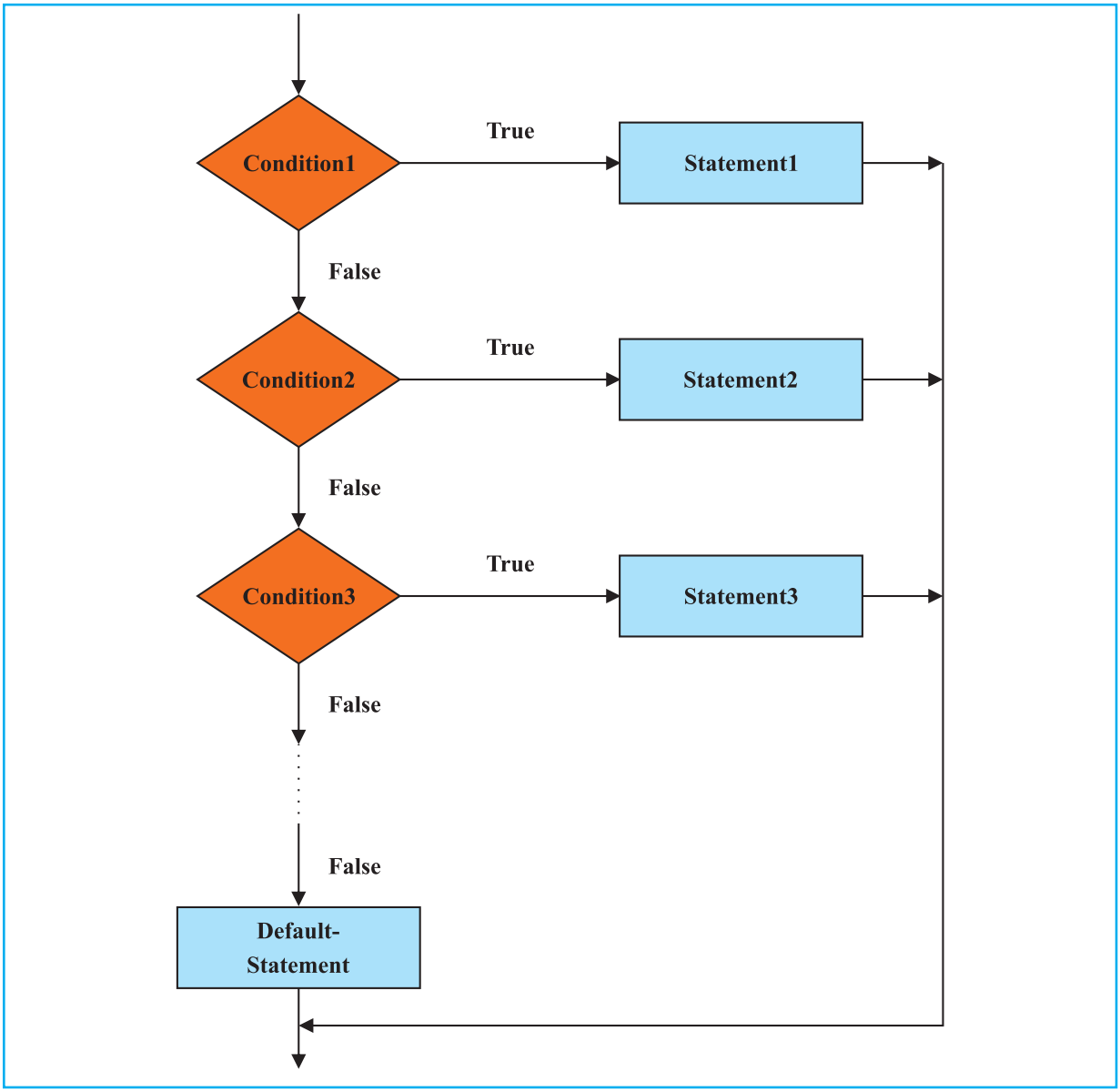


Figure 5.7 : Use of Multi-choice using decision

Connector: Small circle with single capital letter is used as connector. When a problem is relatively large and complex, flowchart is either complex or may not fit into a page. Connectors are used to connect two parts of flowchart within a page or in another page. Connector is always used in pair where circle having letter say "A" with inward arrow (coming out of some symbol) shows that flowchart continues in same part of same page or in another page denoting circle with same letter "A" but outward arrow connecting to the other part in same page or other page. Figure 5.8 shows the pair of connectors with letter "A".

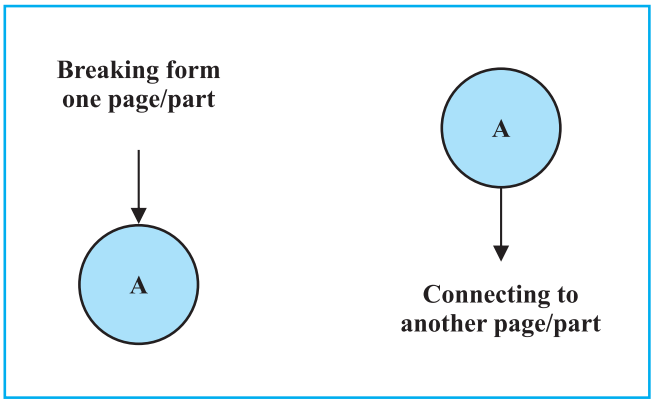


Figure 5.8 : Connectors

Sequential and Decision-making Examples

Let us now use symbols learnt in previous section to develop the flowcharts for various problems including simple sequential steps and decision making.

Problem 1 : Draw a flowchart for adding two given numbers.

Solution :

Problem is very simple and needs to receive two numbers, add them and finally produce the addition of them. Figure 5.9 shows the flowchart.

As shown in figure 5.9, flowchart starts with Start symbol. Then it receives two numbers using variables N1 and N2. Third step is the process step denoted by rectangle to add N1 and N2 into variable Sum. Forth step prints the addition stored in variable Sum. Last step is End to finish the flowchart.

We can easily draw the flowchart for the subtraction by making small changes in figure 5.9. First change is to replace the process step (third symbol) by

Diff = N1 – N2

and then changing the forth step with

output the difference

Print Diff

Problem 2 : Draw a flowchart to compute the simple interest using

I = (P x R x N)/100

where P = Principal amount, R = Rate of interest and N = Period.

Solution :

Let us take an example where P = 100000, R = 7% and N = 1 year.

Then interest I = (100000*7*1)/100 = 7000.

Figure 5.10 shows the flowchart. As seen in the flowchart, three variables P, R and N are used to receive the input and then process box computes the interest I, using above mentioned formula and finally interest calculated is printed.

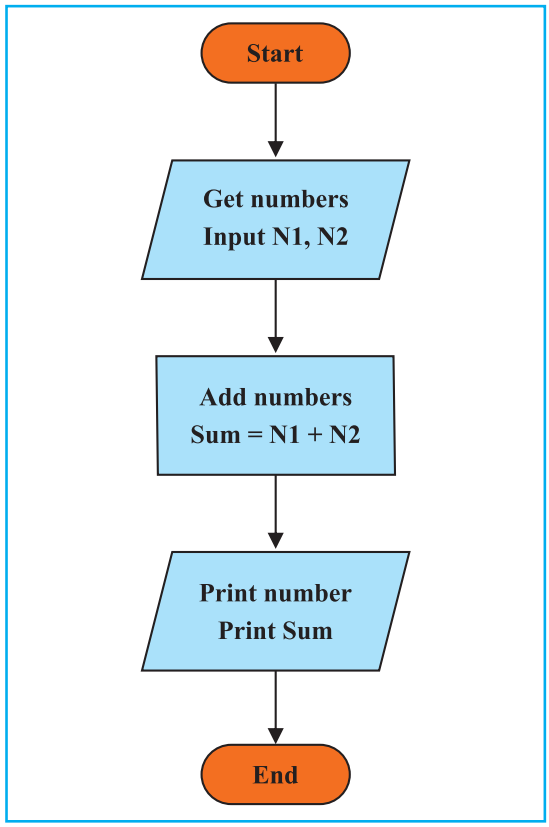


Figure 5.9 : Flowchart to add two numbers

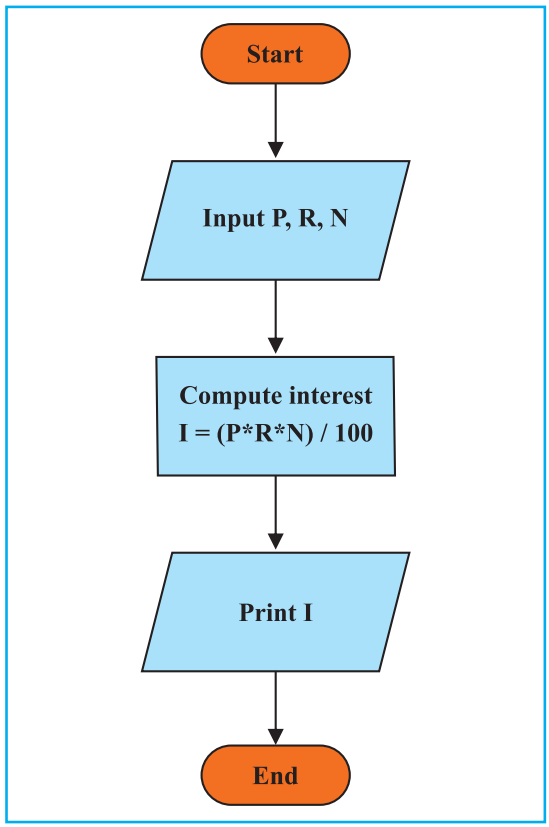


Figure 5.10 : Flowchart to compute simple interest

Problem 3 : Draw a flowchart to compute area and periphery of the circle.

Solution :

To compute, area and periphery of the circle, we need to accept radius as input. We can compute area using $A = \pi R^2$ and the periphery using $P = 2\pi R$. Figure 5.11shows the flowchart. Observe that we are computing two values in a single processing step. We can compute as many as value in a single process box if they are in sequence.

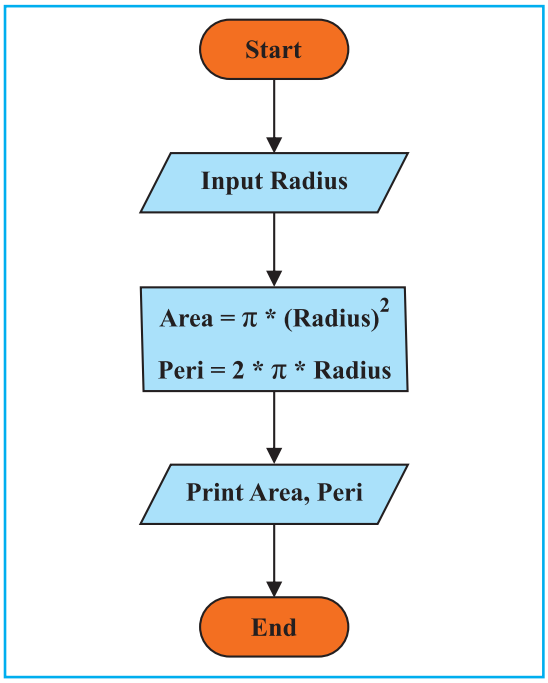


Figure 5.11 : Flowchart to compute Area and Periphery of Circle

Problem 4 : Draw a flowchart to find the maximum of two numbers.

Solution :

This flowchart uses diamond symbol to compare two numbers using ">" (greater than) sign and then prints one of them whichever is larger. Figure 5.12 shows the flowchart.

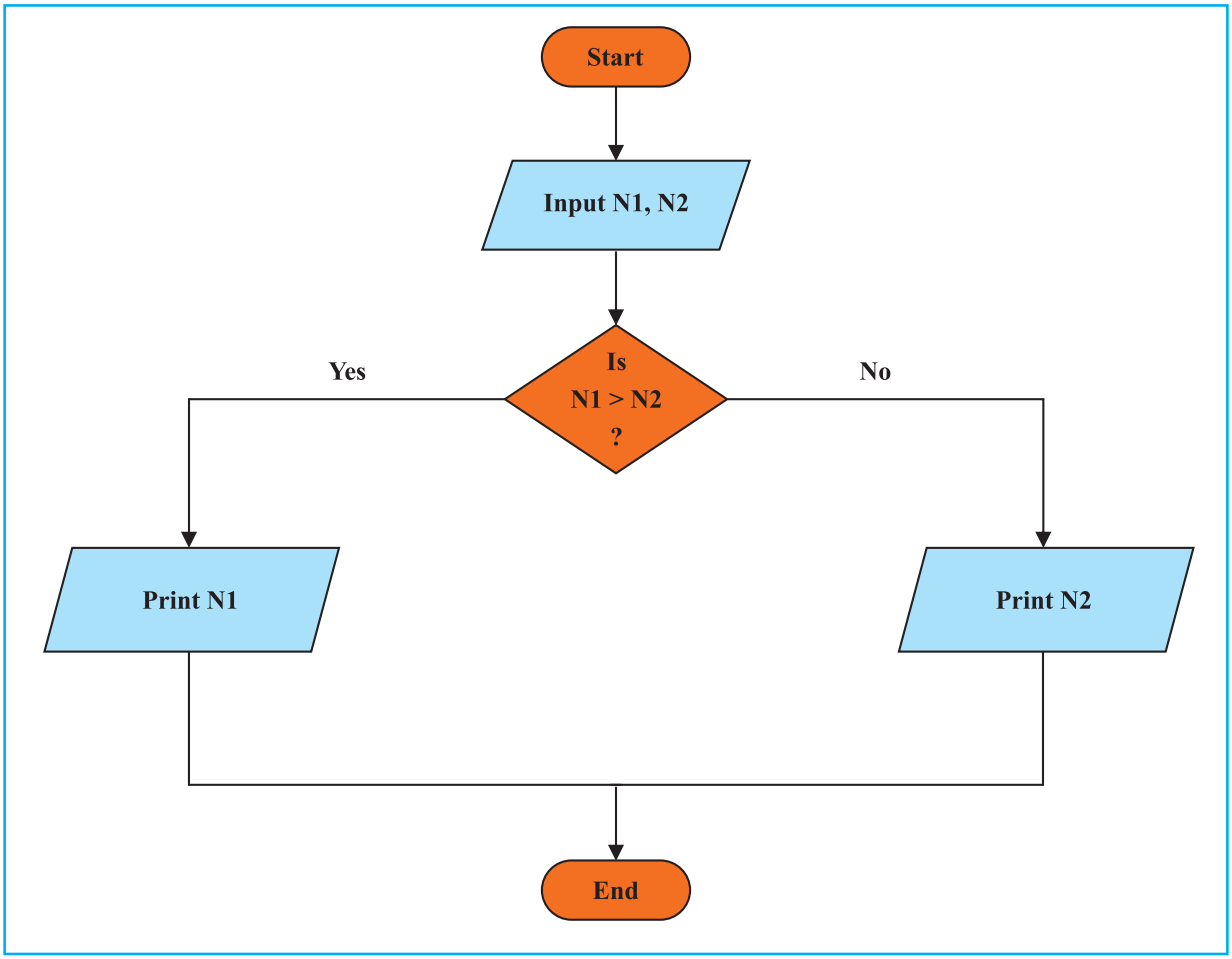


Figure 5.12 : Flowchart to find Maximum of Two Numbers

As shown in flowchart, after receiving two numbers N1 and N2, they are compared in decision box using condition N1 > N2. The decision box creates two paths, one if the condition is true (printing N1) and another if the condition is false (printing N2). After printing one of them flowchart ends.

If we simply change the condition to N1 < N2 in decision box of the flowchart in figure 5.12, then it will print the minimum of N1 and N2.

Problem 5 : Draw a flowchart to find the maximum of two numbers and check whether the maximum number is odd or even.

Solution :

Figure 5.13 shows the flowchart for the problem. As seen in flowchart, first maximum number is found (refer figure 5.12) and stored in the variable Max. Then, we use decision box to test whether Max is odd or even using modulo by 2 (Modulo i.e. % operator gives the remainder after division). If the Max%2 = 0, then it is Even, otherwise (=1) it is odd. Finally, we print the Max with label "Max is Even" or "Max is Odd".

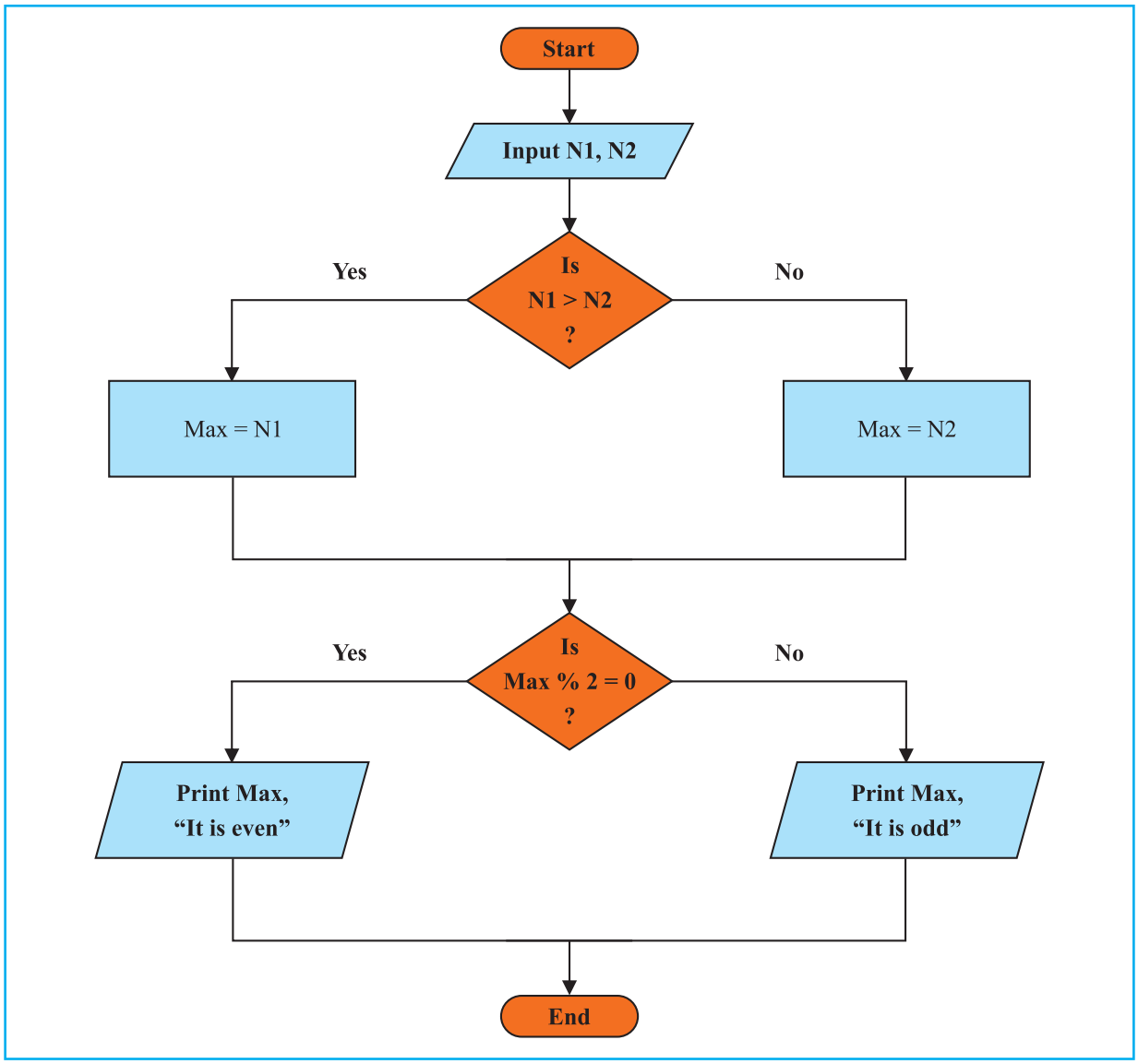


Figure 5.13 : Flowchart to find Maximum of Two is Odd or Even

Nested Conditions and Looping with Examples

Nested condition means condition inside the condition. Sometimes, it is necessary that after applying one condition, we need to apply another condition based on the outcome of the first condition either true or false. Consider the situation shown in figure 5.14. First, we evaluate the Condition1. If it is true, then we apply Condition2 and continue further based on whether Condition2 is true or false. Same way, if Condition1 is false, we will apply Condition3 and continue further based on its outcome either true or false.

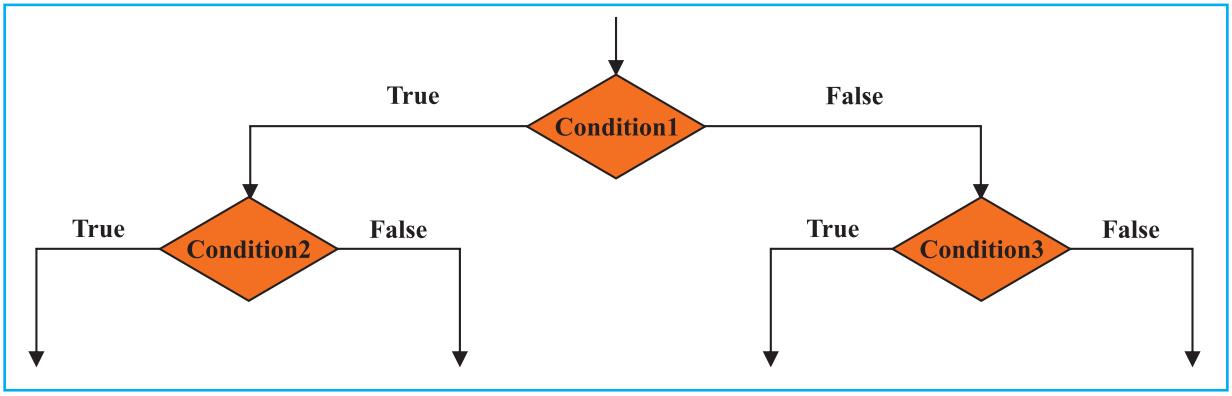


Figure 5.14 : Nested Conditions

Problem 6 : Draw a flowchart to find minimum of three numbers.

Solution :

Figure 5.15 shows the flowchart for finding the minimum of three numbers. Flowchart first accepts three numbers using variables N1, N2 and N3. Then decision box compares N1 and N2 (N1 < N2). If N1 is less than N2, then nested decision box (left side) compares N1 with N3 (N1 < N3) to find whether N1 is minimum or N3. If N2 is less than N1, then another nested condition box (right side) checks N2 < N3 to find whether N2 is minimum or N3. Finally, the minimum of all three is printed.

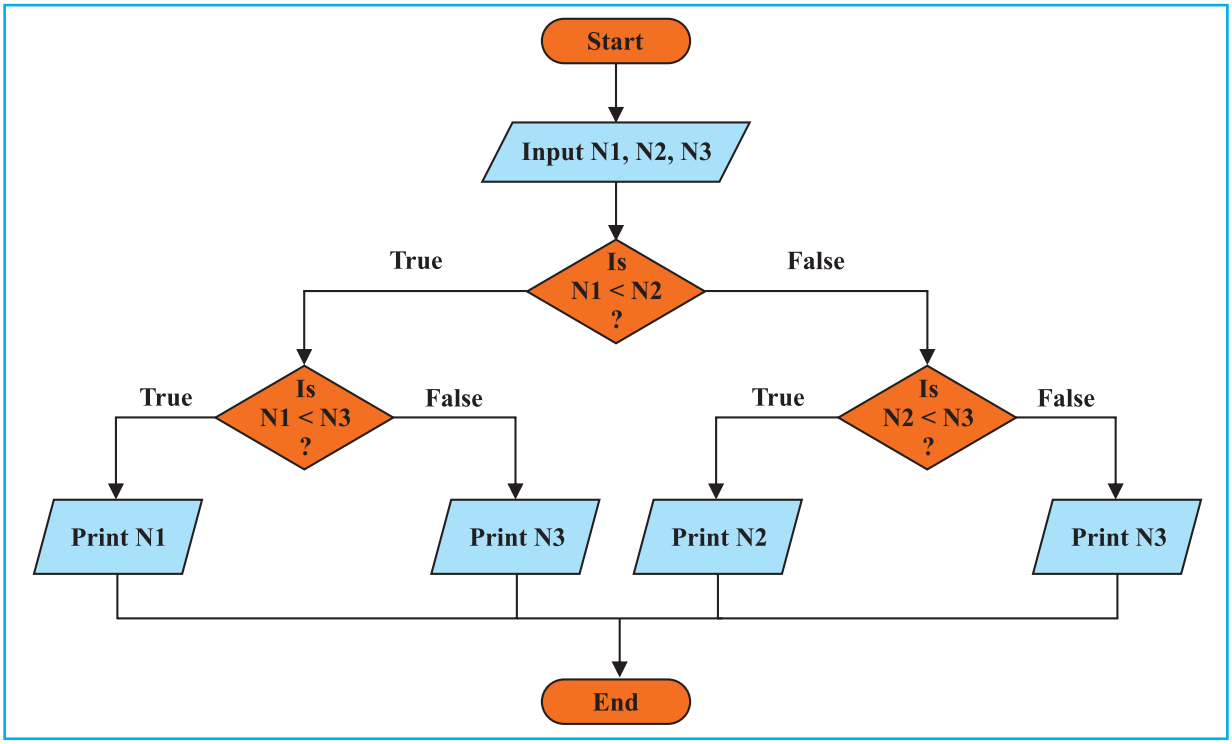


Figure 5.15 : Flowchart to Find Minimum of Three Numbers

For example, assume that N1 = 5, N2 = 2 and N3 = 7. Condition N1 < N2 (5 < 2) is false. Control moves to the nested condition on right hand side which checks N2 < N3 (2 < 7) which is true and finally minimum of three N2 = 2 is printed. You can test it with various combinations of N1, N2 and N3.

It is also possible that a task needs to repeat (looping) some set of statements based on some condition. Earlier we have taken an example to add 1 to 10 numbers, in which we have to repeat statement to add next value to Sum and increment I by 1 every time as long as I is less than or equal to 10. Let us understand it through following example.

Problem 7 : Draw a flowchart to add 1 to 10.

Solution :

Flowchart is shown in figure 5.16. First process box initializes the Sum = 0 and I = 1. Then decision making statement I <= 10 is performed and if it is true then process box containing Sum = Sum + I and I = I + 1 is executed. After execution, the arrow points back to decision box and condition I <= 10 is again evaluated with updated value. It is repeated as long as I <= 10. When I = 11, condition becomes false and control moves to the next step after loop and finish.

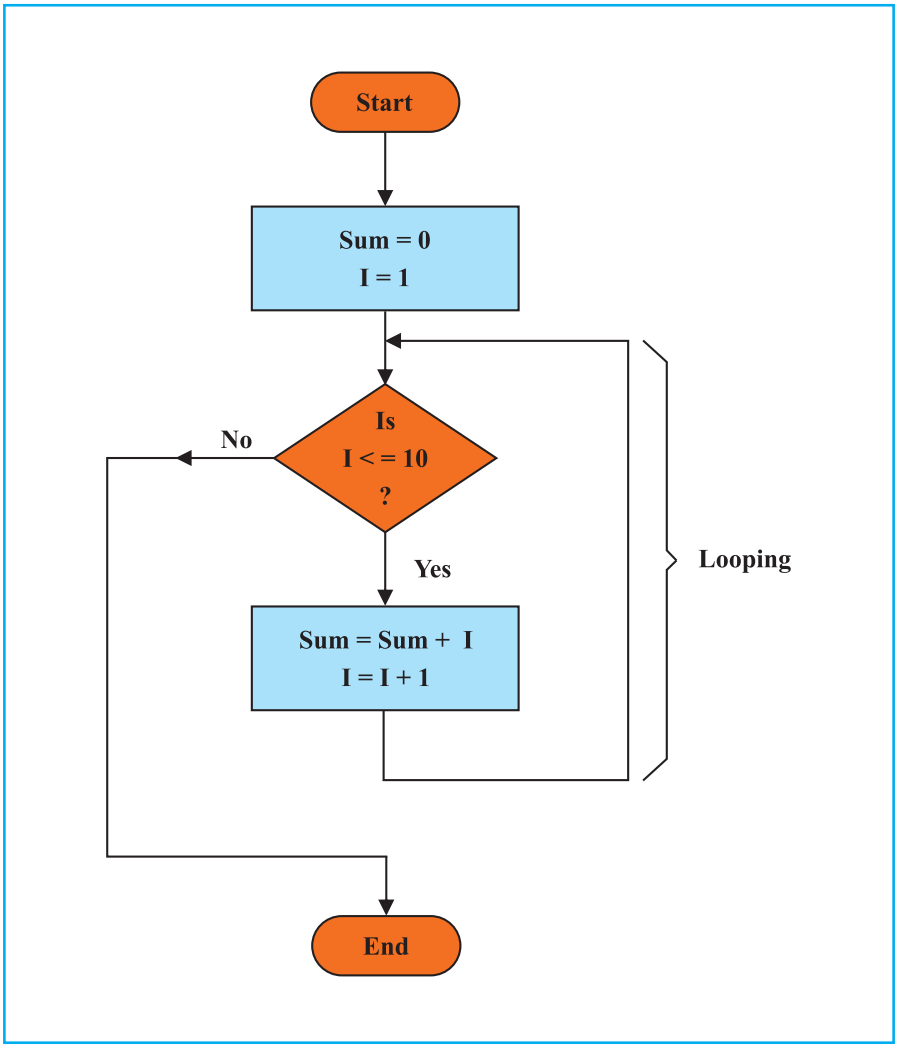


Figure 5.16 : Flowchart to Add 1 to 10

Problem 8 : Draw a flowchart to sum the first 100 even numbers.

Solution :

First 100 even numbers are covered in the range 1 to 200 and summing them means

$2+4+6+\ldots.+200$

Figure 5.17 shows the flowchart. We need three variables: Sum for storing addition of all even numbers, Count to keep track of when to finish and Even for storing next even number. They are initialized as Sum = 0, Count = 0 and Even = 2 (first even number). We will add next even number in Even to Sum and then increment Even by 2 and Count by 1. After processing it, flowchart checks if Count is less than 100, then we will repeat the process otherwise task is finished.

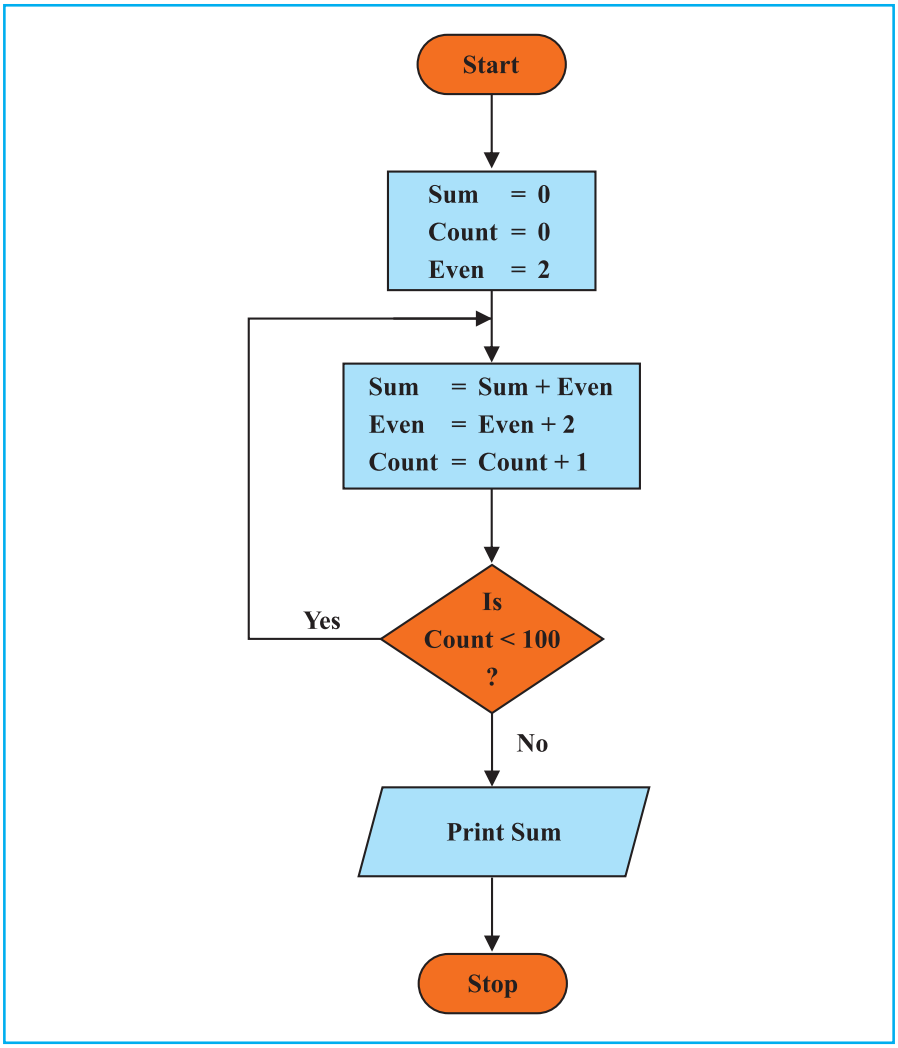


Figure 5.17 : Flowchart to Add First 100 Even numbers

Overview of Algorithm

Algorithm is a stepwise solution of a given problem in logical order. It is just like a recipe of any food describing exactly what to do in specific order. Algorithm can be written using descriptive phrases or pseudo codes/instructions. Pseudo codes use words and symbols to represent operations or steps.

They are very compact and easy to understand. Algorithm gives the decomposition of problem into small steps which are easily understandable. Algorithm looks very similar to computer program except it uses common syntax like pseudo codes rather than specific syntax of a particular programming language. It is very easy to convert an algorithm to a computer program as whole solution is already developed using steps in a logical order. Writing algorithm before developing program ensures clarity and correctness of the program.

Following is an example of an algorithm to find the area of rectangle.

1. START
2. Input Length, Breadth
3. Compute Area = Length * Breadth
4. Print Area
5. STOP

Observe that each step in an algorithm is given number starting from 1. Algorithm starts with START. The pseudo codes Input, Compute, Print denotes the input, computation and output operations respectively. The last step of the algorithm is always STOP denoting the end of the algorithm. Above algorithm first receives Length and Breadth, then it computes Area by multiplying Length and Breadth and finally prints the Area as an output.

Examples of Algorithm

Let us write the various algorithms including sequence, decision making and repetition so that we can understand it in depth.

Problem 9 : Write an algorithm to find the square of a given number.

Solution : The algorithm is as follows.

1. START
2. Input N
3. Compute Square = N * N
4. Print Square
5. STOP

It is very simple and follows the steps in sequence from 1 to 5. It accepts N, computes Square and then prints the Square.

Let us take the examples which includes the decision making and repetition/looping.

Problem 10 : Write an algorithm to find maximum of two numbers.

Solution : The algorithm is as follows.

1. START
2. Input N1, N2
3. If N1 > N2, goto 6
4. Print N2
5. goto 7

6. Print N1
7. STOP

First algorithm reads two numbers using variables N1 and N2. Step 3 to step 6 implements the decision making where step 3 compares N1 and N2 (N1 > N2) and, if true, then it jumps to step 6 to print N1 followed by STOP. If it is false, then follows the step 4 to print N2 and step 5 sends control to step 7 to STOP. If we don't use goto 7 in step 5 then after printing N2, it will also print N1 which is wrong. Hence, care has been taken to avoid it. Flowchart for the same is given in figure 5.12. Compare the steps in algorithm and flowchart so that it will be very clear that how we can convert algorithm to flowchart and vice versa.

Problem 11 : Write an algorithm to add first 100 even numbers.

Solution : The algorithm is as follows.

1. START
2. Initialize Sum = 0
3. Initialize Count = 0
4. Initialize Even = 2
5. Compute Sum = Sum + Even
6. Compute Even = Even + 2
7. Increment Count by 1
8. If Count < 100, goto 5
9. Print Sum
10. STOP

Steps 2, 3 and 4 initializes the variables Sum, Count and Even respectively. Step 5 to step 8 implements the repetition or loop. Step 5 adds current Even value to Sum. Step 6 adds 2 to the Even to get next even number. Step 7 increments Count by 1 as already one even number is added to the Sum. Step 8 checks if Count is less than 100. If it is, then it jumps to step 5 and loop is repeated again. When Count = 100, condition is false and it follows step 9 to print the Sum. Thus, step 5 to step 8 are repeated 100 times and each time next even number (2,4,6, …, 200) is added to Sum and Count goes from 0 to 99 and when it is 100, the loop is over. Flowchart for the same is given in figure 5.17. Compare it with above algorithm to understand how a loop is implemented.

Flowchart Vs. Algorithm

We have studied the flowcharts and algorithms with examples and understood that how sequence, decision and looping can be implemented which are most important components of problem solving. Any problem in real life is different combinations of these three elements. We can use either flowchart or algorithm or both to represent the process i.e. problem solution in structured and logical order step-by-step depending on the need or choice. Although both flowchart and algorithm are used as problem solving methods, they have their own merits and demerits. Let us compare flowchart and algorithm. It is given in following table.

Flowchart	Algorithm
Provides visual representation of process using symbols.	Uses pseudo codes / instructions to represent the process.
All the paths are visible and easy to follow.	Paths are hidden and needs to read the steps to understand.
Shows the flow of the process which is good for visual learners.	Provides logic of the process using steps which is good for logical thinkers.
Good for planning and explaining the process.	Good for writing computer programs as algorithm is very similar to computer program.
Good for beginners.	Good for computer programmers.

Summary

We started this chapter with the introduction to problem solving and its importance. The problem solving is a logical and step-by-step process to find the solution of the problem. Problem solving is at the heart of the Computer Science and it is foundational skill for the Computer Science as well as for the day to day life. We have studied two widely used methods for problem solving which uses structured approach to solve the problems. They are flowcharts and algorithms. Flowchart is a pictorial representation of the solution using visual diagrams to show the flow of the solution.We have developed the flowcharts for many small and simple problems for better understanding. Examples also provided opportunities to trace the flowchart to follow different paths based on decisions. Algorithm provides the step-by-step solution of the problem. For better understanding and clarity, chapter provides examples of writing algorithms for the same problems used to develop the flowcharts. Chapter ends with the comparison between both the problem-solving methods i.e. flowchart and algorithm. These methods reinforce logical thinking and clarity on expressing the solutions. It also allows students to test the solution by providing various inputs and following paths based on decisions. This chapter laid down the strong foundation for learning programming with C in coming chapters.

EXERCISE

1. What is problem solving? Give an example.
2. Mention the problem-solving methods with their benefits.
3. What is flowchart? List the symbols used by flowchart with their function.
4. What types of operations are normally part of process box of the flowchart?
5. Which symbol is used for decision making? Explain its use with an example.
6. How would you create a looping using flowchart symbols?
7. What is algorithm? Give a simple example.
8. Give an example of repetition using algorithm.

9. Give the drawbacks of algorithm.

10. Compare flowchart and algorithm.

11. State whether true or false.

(1) Arrow is used to connect two steps in flowchart.

(2) All the possible paths are visible in algorithm.

(3) Problem solving is logical and step-by-step process to solve the problem.

(4) Multi-choice can be implemented by diamond.

(5) Flowchart always contains Start and End steps.

12. Fill-in the blanks.

(1) ……. is pictorial representation of the solution.

(2) Algorithms use …….. code.

(3) Connectors use ……. symbol in flowchart.

(4) Normally arithmetic and logical operations are placed in …… box.

(5) …….. and …….. are structured methods for problem solving.

13. Multi-choice questions. Choose the most correct answer.

(1) Which of the following symbol is used for Start and End of a flowchart?

(a) Rectangle (b) Diamond (c) Oval (d) Circle

(2) Which of the following provides visual representation of problem solution?

(a) Algorithm (b) Flowchart

(c) Program (d) Process

(3) Which of the following symbol is used to denote the process in flowchart?

(a) Rectangle (b) Diamond (c) Oval (d) Circle

(4) Which of the following is used to distinguish connector pairs?

(a) Direction of arrows (b) Letter in circle

(c) Size of the circle (d) Colour of the circle

(5) Comparison is used in which of the following?

(a) Process (b) Decision-making

(c) Input (d) Output

(6) Which of the following is best for identifying different paths within solution?

(a) Flowchart (b) Algorithm

(c) Flowchart and Algorithm both (d) We can not see the path

(7) Which of the following uses pseudo codes to represent steps of solution?

(a) Program (b) Flowchart (c) Algorithm (d) Process

(8) Which of the following is normally used to perform computations?

(a) Process box (b) Decision box

(c) Input box (d) Output box

(9) Which of the following is used when flowchart is split into more than one pages?

(a) Letter with arrows (b) Connectors

(c) Arrows (d) Special symbols

(10) When we need to compute two values, we can

(a) Put them in one process box one by one

(b) Put them in two separate process box one after another

(c) Both A and B

(d) Must be in two separate box one after another

Laboratory Exercise

1. Draw a flowchart to convert Fahrenheit to Celsius using C = (5.0/9.0) * (F – 32).
2. Draw a flowchart to check whether given positive number is odd or even.
3. Draw a flowchart to add only odd numbers from 1 to 100.
4. Draw a flowchart to accept the students mark of a subject from 100 and print "Fail", if marks is <= 35, "Second class" if marks is > 35 and <= 60, print "First class" if marks > 60 and <= 70, otherwise print "Distinction".
5. Draw a flowchart to guess a number. It accepts number repeatedly from user until number is 7. When number is 7, it prints "Congratulations!" and finish.
6. Draw a flowchart to find minimum from given 10 numbers.
7. Write an algorithm to find sum and difference of two given numbers
8. Write an algorithm to find maximum of three numbers.
9. Write an algorithm to add only even numbers from given N numbers as input.
10. Write an algorithm to add following series : 1-2+3-4+5-6+ ……N.

Introduction to C Programming

Introduction

In the previous chapter, we learned how to design algorithms and draw flowcharts. These help us plan and understand a solution step by step before we start coding. But planning is only the first part. To make our solution work on a computer, we must express these logical steps in a specific programming language such as Python, Java or C. Among these languages, C is very important. It is fast, powerful, and forms the base for many other programming languages. In this chapter, we will start learning C programming. We will learn how to write simple C programs, understand the main components of a C program, and explore the basic characters, numbers, and symbols used in C. This is our initial step into the world of computer programming.

Why Do We Need a Programming Language?

Natural languages often suffer from ambiguity, where a single sentence can have multiple interpretations. This is a significant challenge when communicating with computers, which require instructions that are precise and unambiguous. Unlike humans, computers cannot conclude meaning from unclear instructions. Programming languages address this by enforcing strict rules. These rules ensure that every instruction has exactly one intended meaning. Learning a programming language is a new form of communication. We can effectively command computers to perform specific tasks.

Understanding Programs and Programming

Before exploring C, let us start with core programming concepts. A program is a specific sequence of instructions directing a computer to perform tasks, such as calculations or data display. Since computers process only binary codes (0s and 1s), we use structured programming languages like C that enforce unambiguous sentence structure. Specialized tools called compilers then translate this human-readable code into executable machine language (binary codes). Programming languages exist on a spectrum: High-level languages prioritize human readability and abstraction, while low-level languages offer hardware control at the cost of complexity. C uniquely bridges these categories, providing both efficiency and expressive power.

History of C Language

The C Language was developed by computer scientist Dennis Ritchie at Bell Labs in 1972. The C programming language was created to overcome limitations found in earlier languages like B and BCPL. Ritchie aimed for a language that was simpler, faster and could access computer hardware directly. Despite its age, C remains widely used globally because it is relatively easy to learn, could run efficiently on many different computer systems and allowed programmers to write

powerful and efficient code. Its core ideas heavily influenced later major computer languages. Today, C continues to be a fundamental language taught in schools and used in industry to build software systems.

Block diagram of a C program

Figure 6.1 shows the basic flow of a C program. It starts with the Preprocessor directives like *#include* followed by the main Function int *main()*. Then, variables are declared. Output is produced using *printf()*, and the program ends with a return statement, usually return 0;, indicating successful execution. First step to write a C program is to type it using an editor. We can use any available editor on our computer systems. Compiler like GCC can be used for compiling a C program.

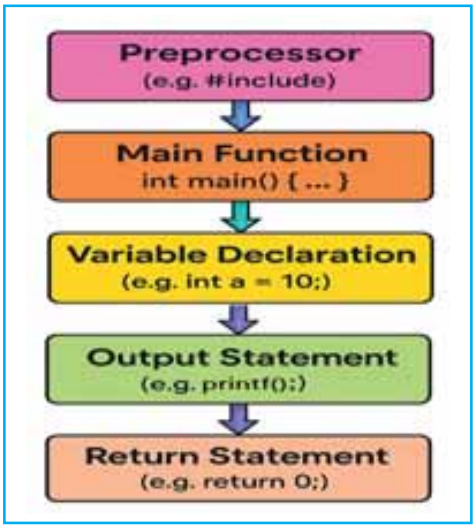


Figure 6.1 : Basic flow of a C Program

Step-by-Step: Create, Save, Open, and Run - hello.c

1. Open an editor
2. Type your code
3. Click on **File → Save As**
4. Name the file as: hello.c
5. Make sure to **save it with ".c" extension** (not .txt)

Following C program demonstrates how to print "Hello, World!" on screen.

```
/* C Program to print Hello World! */

#include <stdio.h>
int main()
{
      printf("Hello, World!\n");

      return 0;
}
Result:
Hello, World!
```

This is a basic program that beginners write to understand how C programming works. C program begins with the line **#include <stdio.h>,** which tells the computer to include the Standard Input/Output library. This allows us to use functions like *printf()* to display output on the screen. The second line, **int main()**, marks the starting point of the program. Following this, the opening curly brace { indicates the beginning of the main function's body where the program instructions will be written. Inside the body, the line **printf("Hello, World!\n");** is used to print the message "Hello, World!" on the screen and the \n in the string moves the cursor to the next line. The line **return 0;** tells the computer that the program has successfully finished its execution. Finally, the closing curly brace } marks the end of the main function. This simple program demonstrates the basic structure and functionality of a typical C program. Steps to write, save, compile and execute C program are given at the end of this chapter.

Character Set in C Language

In C programming, the character set refers to the collection of characters that are used to write programs. These characters include letters, digits, special symbols, white spaces and other control characters. Understanding the character set is essential because every program is made up of valid characters that form variables, functions, operators and other elements of the language. Table 6.1 shows the character set of C language.

Character Set Category	Description	Examples
1. Letters	Alphabets used to form identifiers (variables), keywords and other names in C. Both uppercase and lowercase letters are allowed while declaring identifiers.	• Uppercase letters (A–Z) • Lowercase letters (a–z) **Example:** To declare variable names. • int Age; char name;
2. Digits	Used to represent numeric values, array indices, etc.	• 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 **Example:** To define numeric constants, array index, etc. • int num = 45; • int num[5];
3. Special symbol / characters	Symbols that have a special meaning in C (used to specify operators, delimiters, structuring code and defining syntax).	• Arithmetic operators like +, -, *, / etc. • Address of operator (&) • Logical AND (&&), Logical OR (||), etc. • Curly braces to define code blocks { } • assignment operator (=) and • Many more like , ==, (), [], #, %, !, ^ etc.

Table 6.1 : Character Set in C

White Spaces

In C programming, white space characters are used to separate words, symbols and elements in the source code. They do not affect the execution of the program but are important for making the code readable and organized. Common white space characters include spaces, tabs and newlines. Table 6.2 describes different types of white space characters used in C.

White Space Character	Usage/Description
Space	Used to separate keywords, identifiers and operators (e.g., `int a = 10;`)
Tab (\t)	Used to indent code for better structure and readability
Newline (\n)	Moves the cursor to the next line; often used in output
Carriage Return (\r)	Returns the cursor to the beginning of the line
Form Feed (\f)	Used in printing to move to the next page (rarely used)

Table 6.2 : White Spaces

Keywords

In C programming, keywords are predefined, reserved words that have special meanings to the compiler. They are used to perform specific tasks and control the flow of the program. Keywords cannot be used as names for variables, functions or any other identifiers because they are part of the C syntax. C has a total 32 keywords. Table 6.3 shows some of the commonly used keywords.

Keyword	Meaning/Usage
int	Used to declare integer variables. Like 1, 500, 78
float	Used to declare floating-point variables. Like 784.5, 6.78
char	Used to declare character variables
return	Used to return a value from a function
if	Used for conditional branching
else	Used with 'if' for alternative execution
while	Used to create loops that run as long as a condition is true
for	Used to create loops with initialization, condition and increment/ decrement in one line

void	Specifies that a function returns no value
break	Terminates loops or switch statements

Table 6.3 : Keywords

Identifiers

In C programming, identifiers are the names used to identify variables, functions, arrays, structures and other user-defined elements. An identifier allows the programmer to refer to specific data or functionality in the code. Identifiers must follow specific rules in C.

1. an identifier must start with a letter (A–Z or a–z) or an underscore (_)
2. After the first character, digits (0–9), letters or underscores can be used
3. Keywords cannot be used as identifiers

As we discussed earlier, C is case-sensitive, so 'Value' and 'value' are different identifiers. Table 6.4 provides examples of valid and invalid identifiers:

Identifier	Valid/Invalid and Reason
totalMarks	Valid - starts with a letter and contains only letters
_value	Valid - starts with underscore
count1	Valid - letters and digits allowed after the first character
1value	Invalid - starts with a digit
float	Invalid - 'float' is a keyword
user-name	Invalid - hyphen is not allowed
Value	Valid - different from 'value' due to case sensitivity

Table 6.4 : Identifiers

Variables and Constants

Variables and constants are fundamental concepts used to store data within a program. A variable is a name assigned to a storage location where data is stored during the program's execution. The value of a variable may change during the execution of a program. We must declare a variable by specifying its data type (e.g., *int, float, char*) and assigning it a name before using it. For example,

int age = 25;

declares an integer variable named age and initializes it with the value 25. This value can be used in program later.

A constant is a fixed value that, once defined, cannot be altered by the program. Constants ensure that important values are not accidentally modified. A common way to define a constant is to use the *#define* pre-processor directive or the const keyword. For example,

#define PI 3.14159

declaration ensures the value of PI remains 3.14159 (constant) throughout the program. Variables and Constants allow programmers to manage and manipulate data effectively in a program.

Executing C Program in Ubuntu (Linux)

Let us understand the process of writing, compiling and executing a C program on Ubuntu operating system (OS). We can write and execute C programs using the Ubuntu OS terminal.

To run a C program in Ubuntu, follow these steps:

1. **Write the Program**

   Open any text editor like **gedit** as shown in figure 6.2 and write the following C code in **gedit** editor:

```
/* C program execution in Ubuntu OS */
#include <stdio.h>
int main()
{
   printf("Welcome to C Programming \n");
   return 0;
}
```

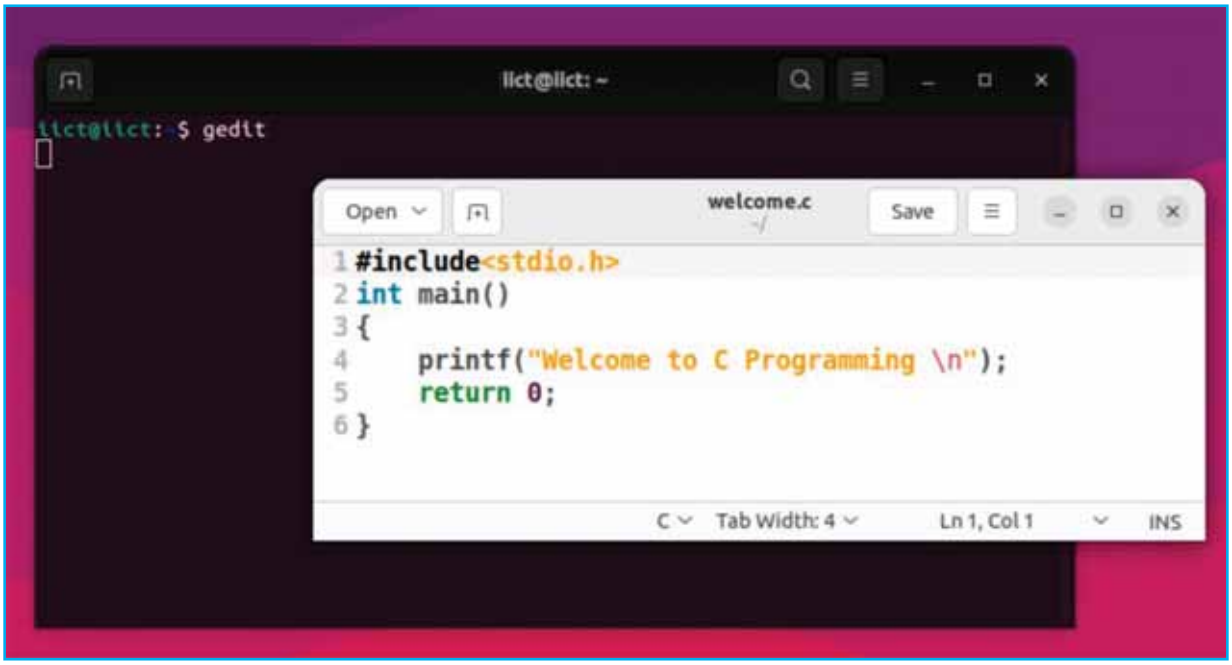


Figure 6.2 : Writing C Program in gedit Text Editor

2. **Save the Program**

Save the file with the name "**welcome.c**". It is crucial to ensure the file has a **".c"** extension to be recognized as a C source file.

3. **Compile the Code**

We need GCC (GNU Compiler Collection) to compile the C Code. Most Ubuntu versions come with GCC which includes the C compiler. Launch your Ubuntu terminal and compile the program using the GCC command as shown below (line 2 of Figure 6.3):

```
gcc welcome.c -o welcome
```

This command compiles "**welcome.c**" and generates an executable file named "**welcome**".

Figure 6.3 : Compile and Execute C Program on Ubuntu Terminal

4. **Execute the Program**

To run the compiled program, enter the command as shown below (line 3 of Figure 6.3):

```
./welcome
```

Upon execution, you can see the following output on your terminal:

```
Welcome to C Programming
```

By following these steps, you can successfully write, compile, and execute a C program on an Ubuntu operating system.

Consider following important points while executing C program on the Ubuntu (or any Linux-based) OS:

1. **GCC Installation:** By default, Ubuntu operating systems have GCC installed.
   - To check installation, run **gcc --version** command on your Ubuntu terminal as shown in Figure 6.4.
   - If not installed, run **sudo apt install gcc** command to install GCC in your computer as shown in Figure 6.4.

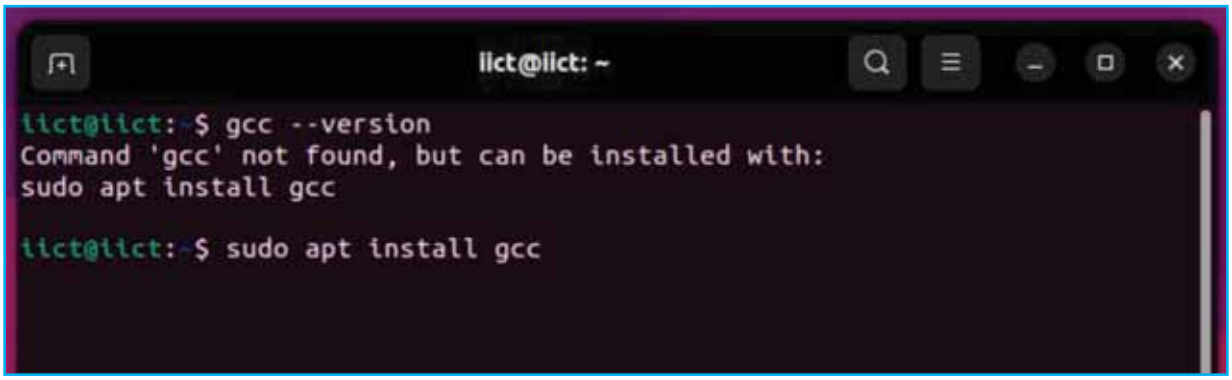


Figure 6.4 : GCC Installation in Ubuntu

2. **File Extensions:** Save your program with a **.c** extension (e.g., welcome.c) so the compiler recognizes it as C code.
3. **Compilation Errors:** If there are syntax errors in your code, GCC will display error messages with line numbers. Carefully read them and fix the mistakes.
4. **Case Sensitivity:** Linux is case-sensitive. File names like **Welcome.c** and **welcome.c** are treated as different files.
5. **Good Practice:** Always use meaningful file names for source files (like welcome.c) and output files.

Summary

This chapter introduces the core concepts of C programming, a language known for its speed, efficiency and portability. We studied the importance of instructions in programming and how C serves as a bridge between human logic and machine execution. The basic structure of a C program is discussed, including character sets, keywords, and identifiers—essential elements for writing valid C code. Additionally, we cover the history of C, which was developed by Dennis Ritchie, and its continued relevance in modern software development. Key tools such as the GCC compiler and text editors are introduced for writing and running C programs. Practical examples are provided to demonstrate how to compile and execute C code on Linux systems.

EXERCISE

1. What is the use of the #include directive in a C program?
2. Why is the main() function important in C?
3. What does the return 0; statement signify in C programming?
4. What is the role of curly braces { } in a C program?
5. How do you display output in C programming?

6. Explain the history of C programming.

7. List out character set in C programming.

8. Explain Case Sensitivity in C programming with example.

9. List out the steps to run C program in Linux.

10. List out the steps to save C program file.

11. State whether true or false.

(1) The main() function is optional in a C program.

(2) Keywords like int and float can be used as variable names.

(3) The #include directive is used to include libraries in a program.

(4) C is not a case-sensitive programming language.

(5) printf() is used to display output in C.

12. Fill-in the blanks.

(1) The ..... function marks the starting point of a C program.

(2) The ..... directive includes a header file in a C program.

(3) The ..... symbol is used to terminate a statement in C.

(4) ..... is used to print text on the screen.

(5) Keywords cannot be used as ..... in C programming.

13. Multi-choice questions. Choose the most correct answer.

(1) Which symbol is used to include a header file in C?

(a) $ (b) @ (c) # (d) %

(2) What is the output of printf("Hello \t World!");?

(a) Hello World? (b) Hello World

(c) Hello World! (d) Error

(3) Which keyword is used to declare an integer variable?

(a) int (b) float (c) char (d) include

(4) Which of the following is not a valid identifier in C?

(a) totalMarks (b) 1value (c) _value (d) value1

(5) What is the purpose of return 0; in C?

(a) Start the program

(b) End the main function

(c) Include header

(d) Print output

(6) Which of the following is a valid variable name?

(a) 123abc (b) _count (c) float (d) -value

(7) Which is used to declare real number?

(a) int (b) float

(c) char (d) None

(8) Which escape sequence is used for a new line?

(a) \t (b) \n (c) \ (d) \r

(9) Which of the following is used to define code blocks?

(a) () (b) [] (c) {} (d) <>

(10) Which of the following is a keyword in C?

(a) loop (b) int (c) print (d) name

Laboratory Exercises

1. Write a C program that prints your full name using the printf() function.

2. Write a C program to print “Hello, India!”.

3. Write a C program to print a message in the following manner.

Hello,

World!