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.

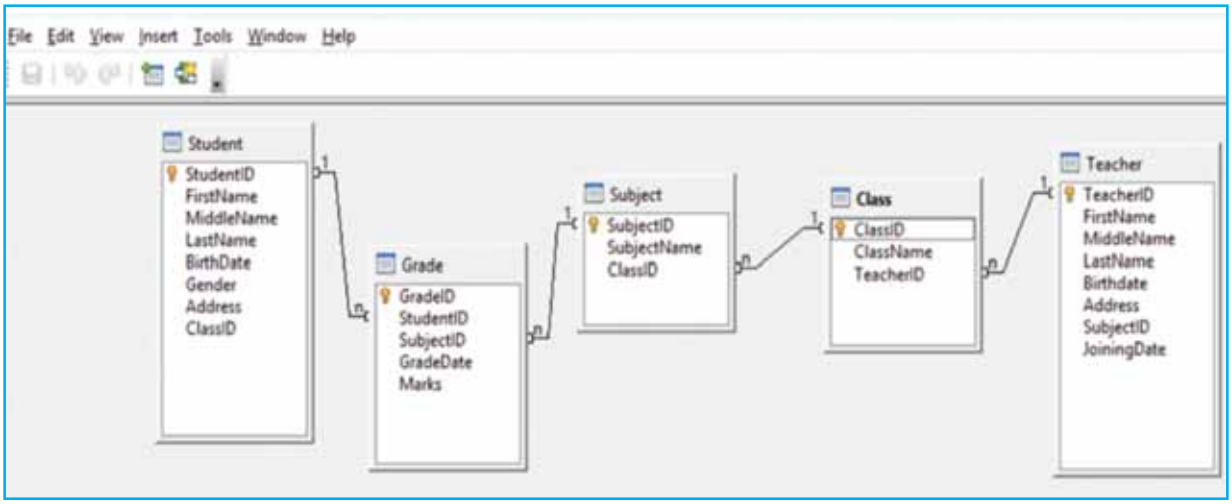

Figure 3.1 : Student Database

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.

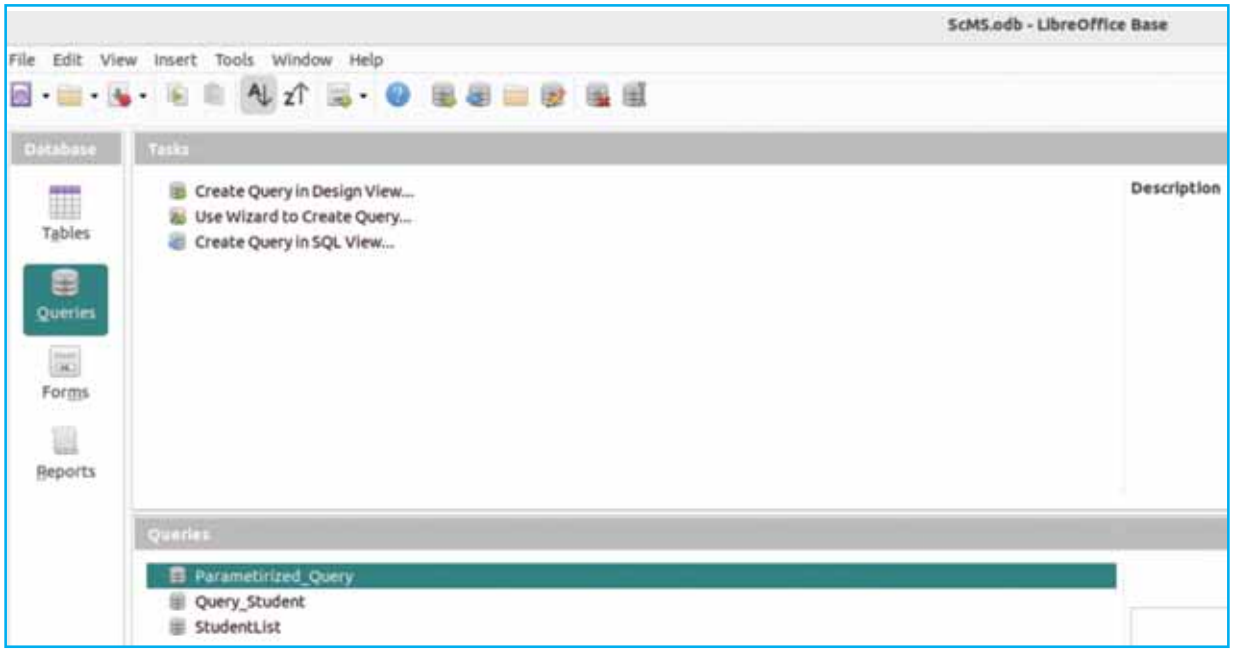

Figure 3.2 : Queries Window

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.

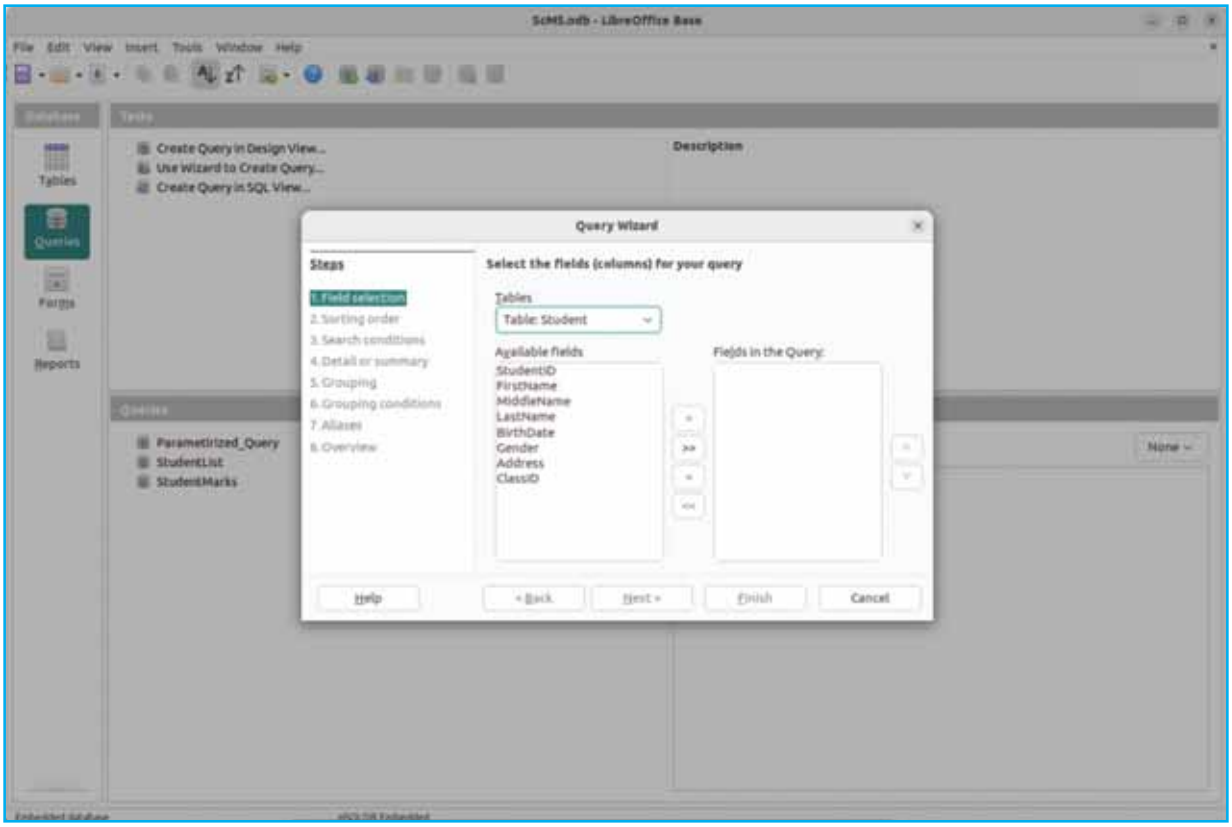

Figure 3.3 : Selection of Table and Fields for Query

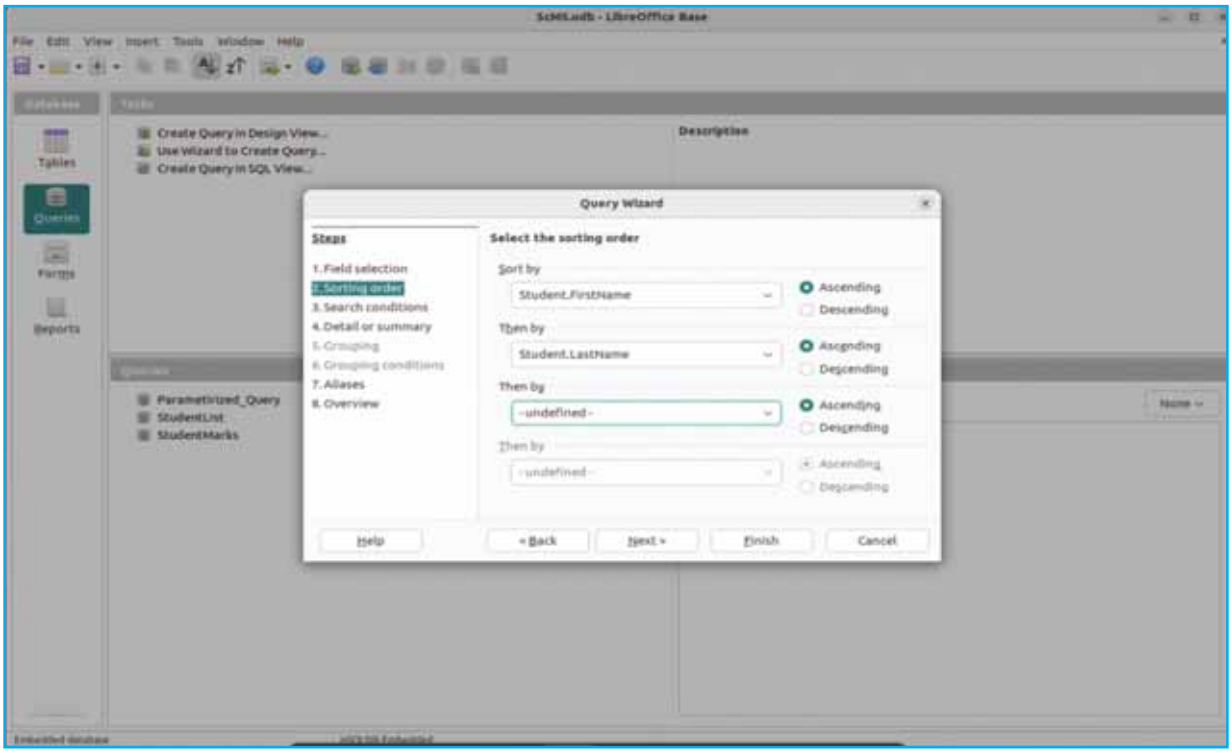

Figure 3.4 : Applying Sorting on a Field

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

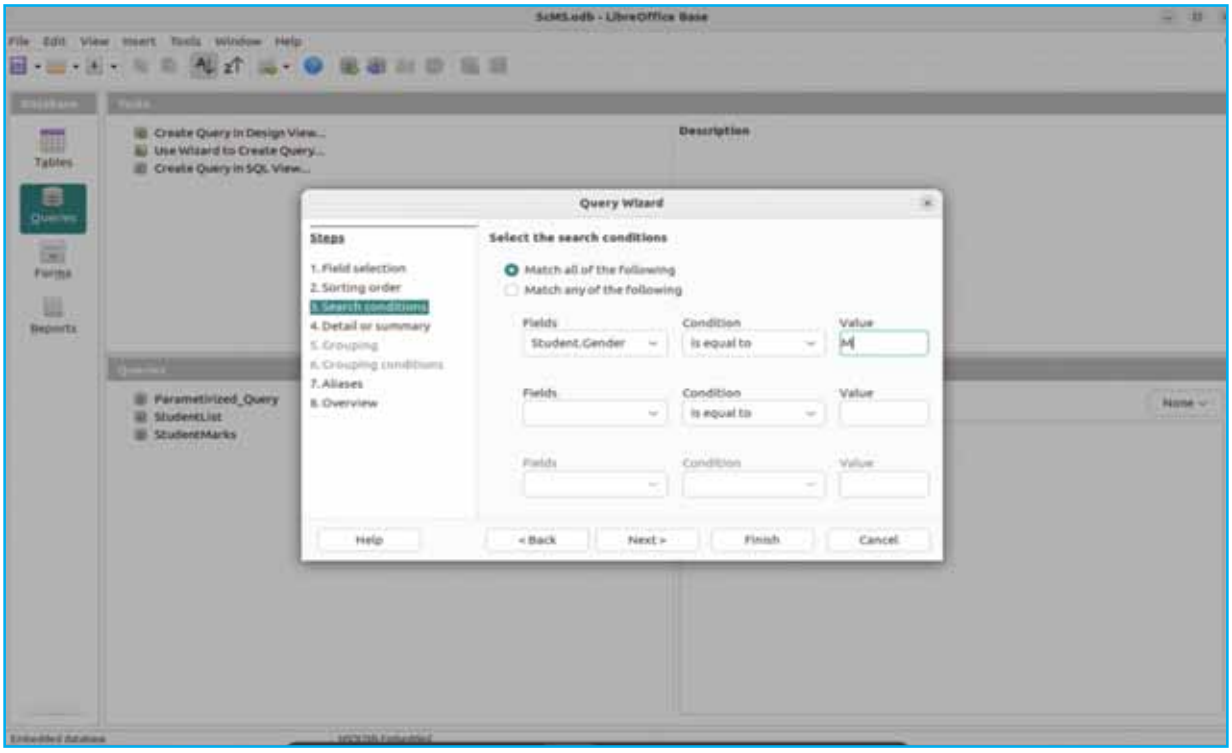

Figure 3.5 : Applying a Search Condition on the Field

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.

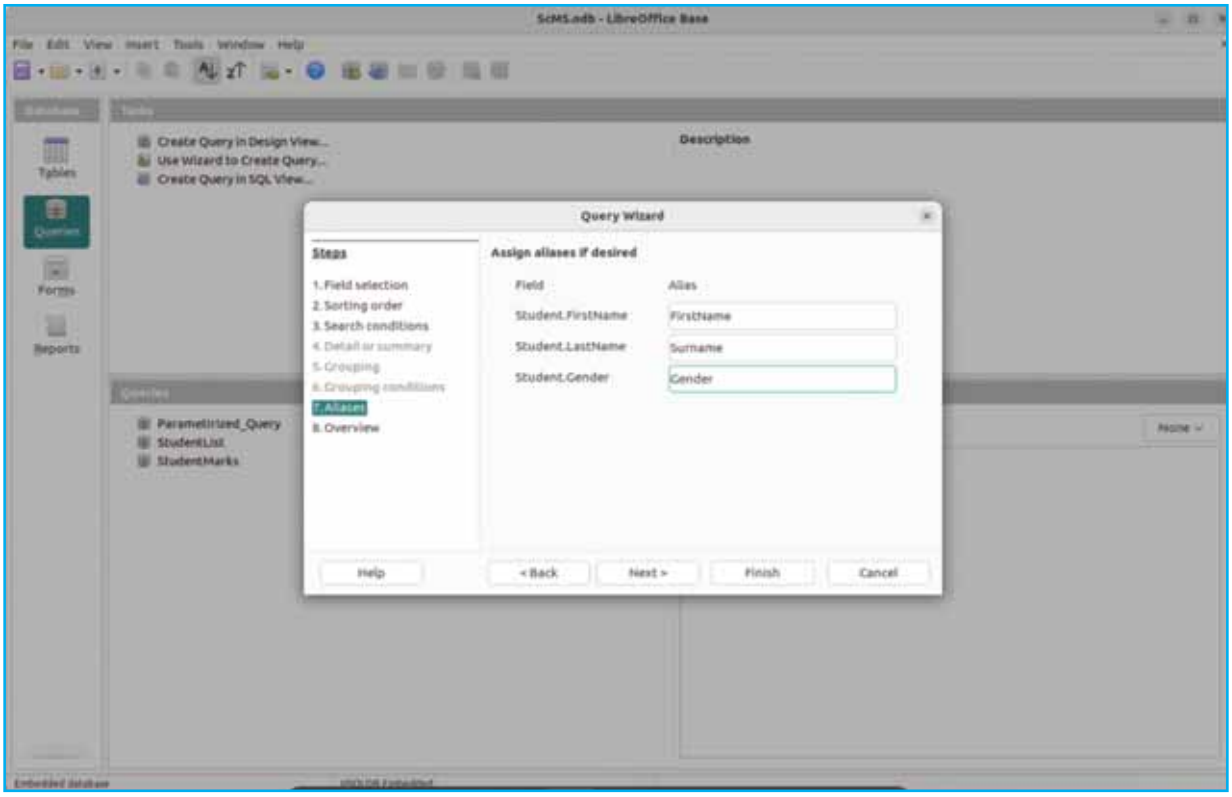

Figure 3.6 : Adding Aliases

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.

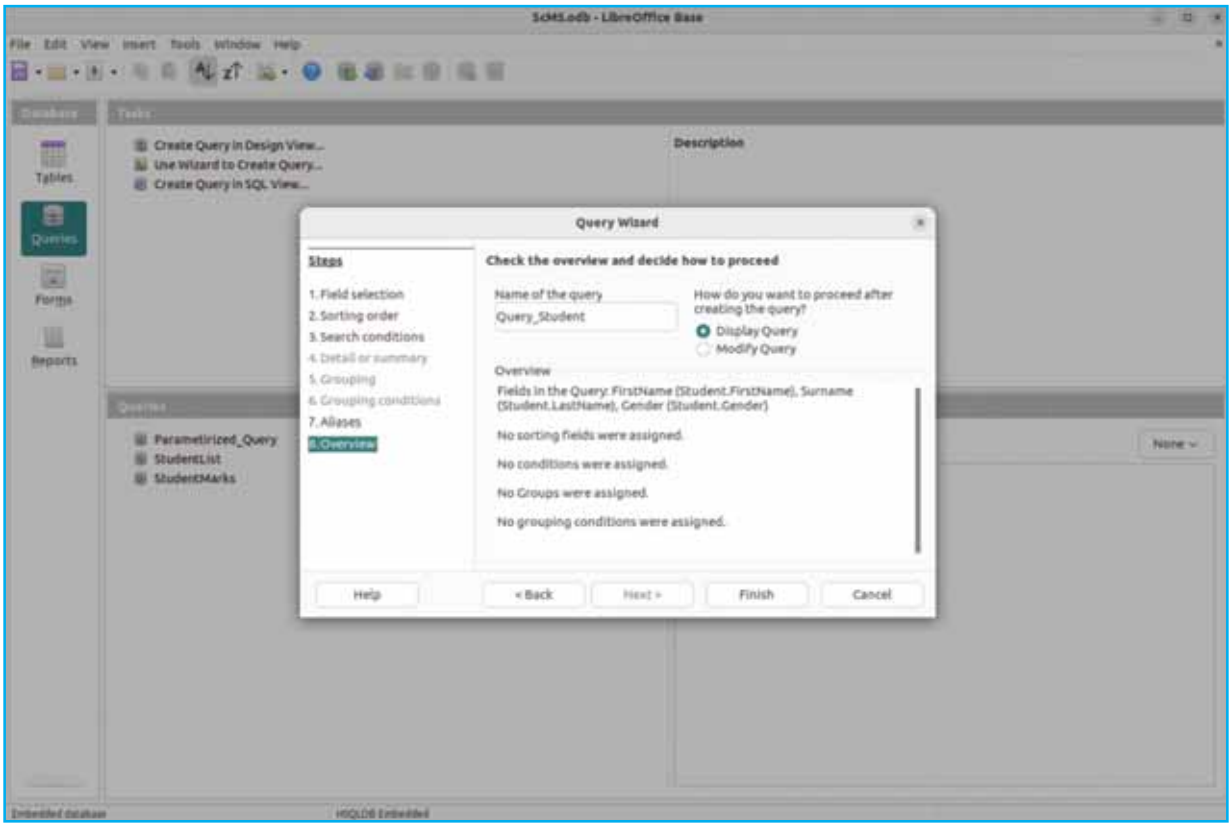

Figure 3.7 : Overview of Created Query

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.)

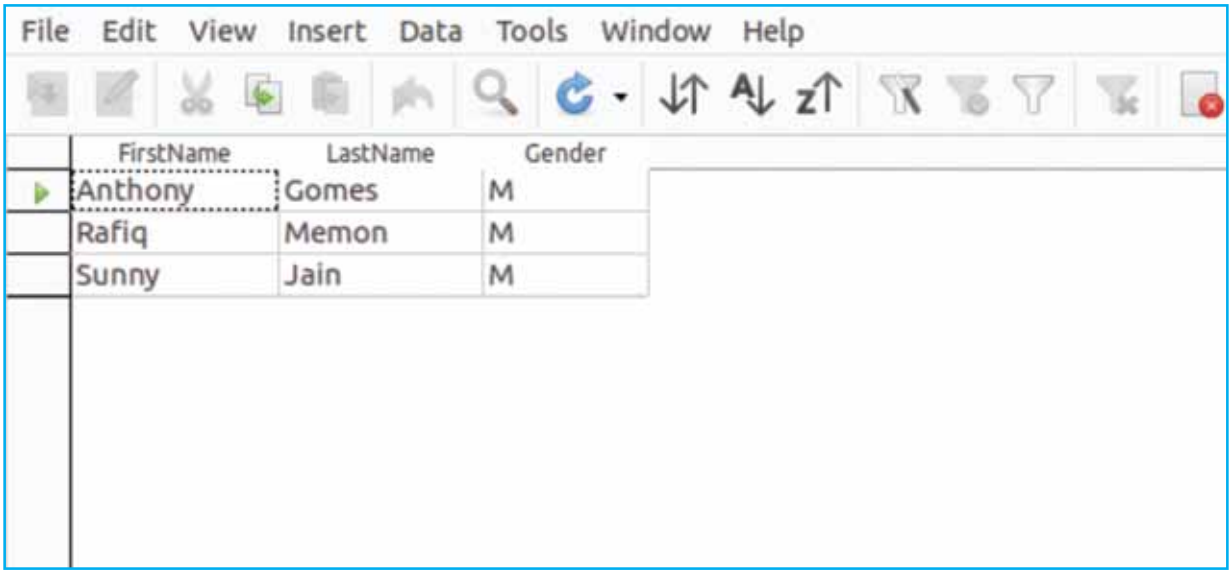

Figure 3.8 : Result of Query

Creating Query Using Design View

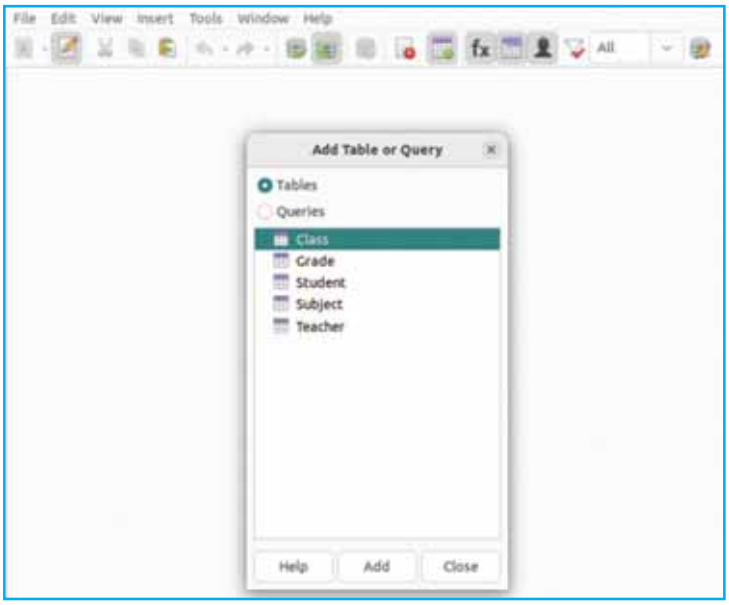

Figure 3.9 : Add Query or Table Dialog box

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.

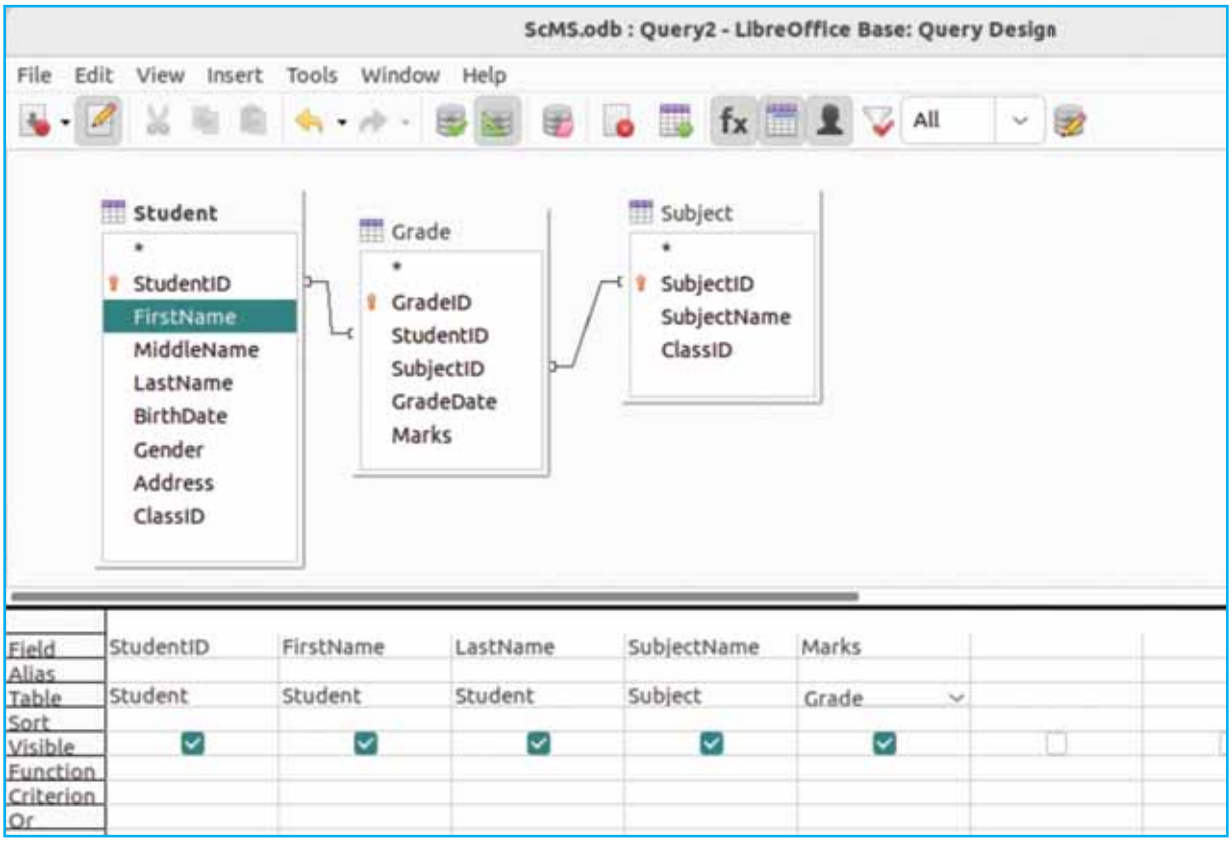

Figure 3.10 : Selection of Fields

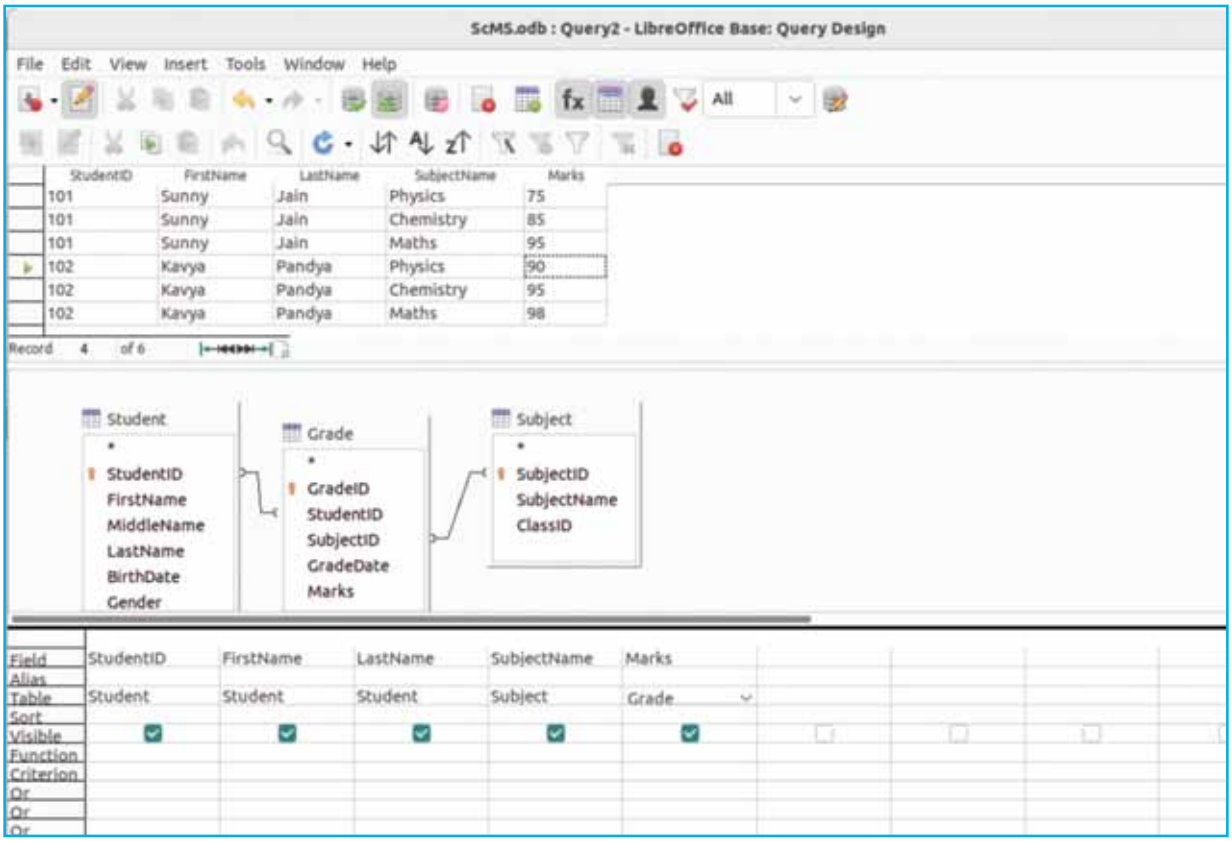

Figure 3.11 : Query Output

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.

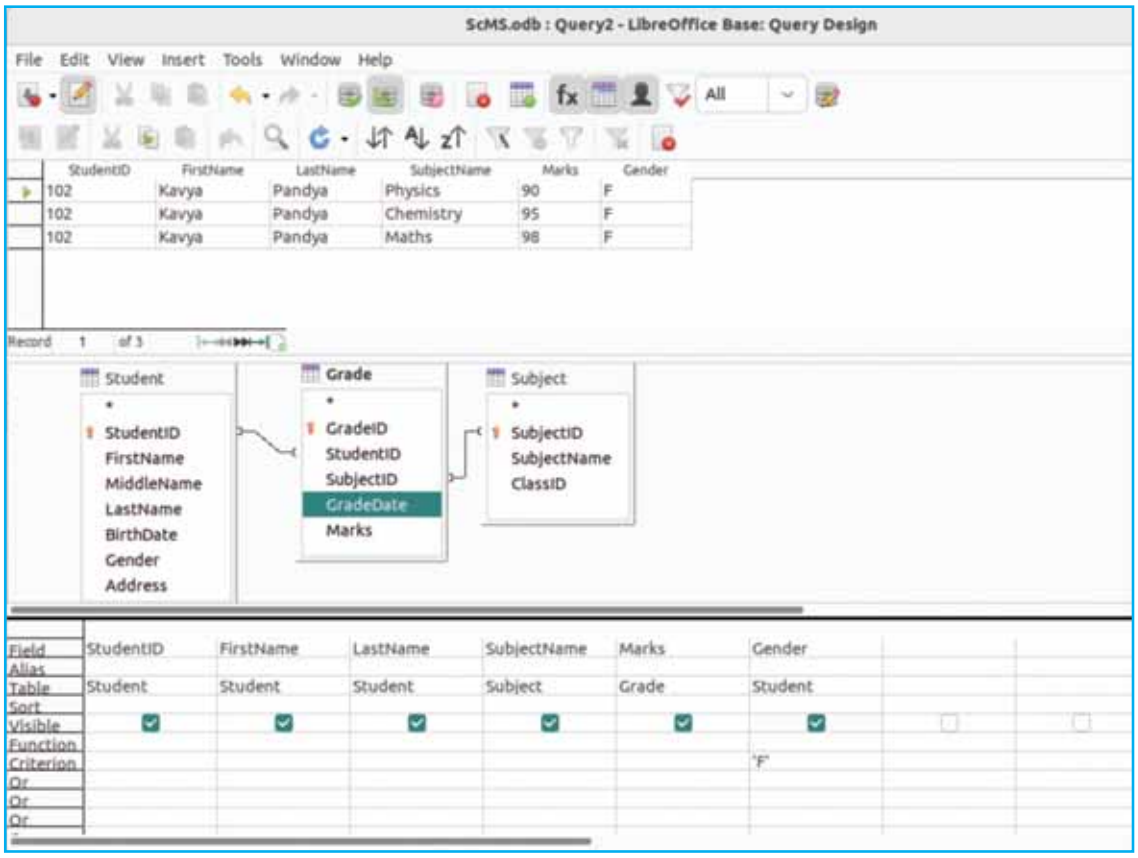

Figure 3.12 : Setting Criteria

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

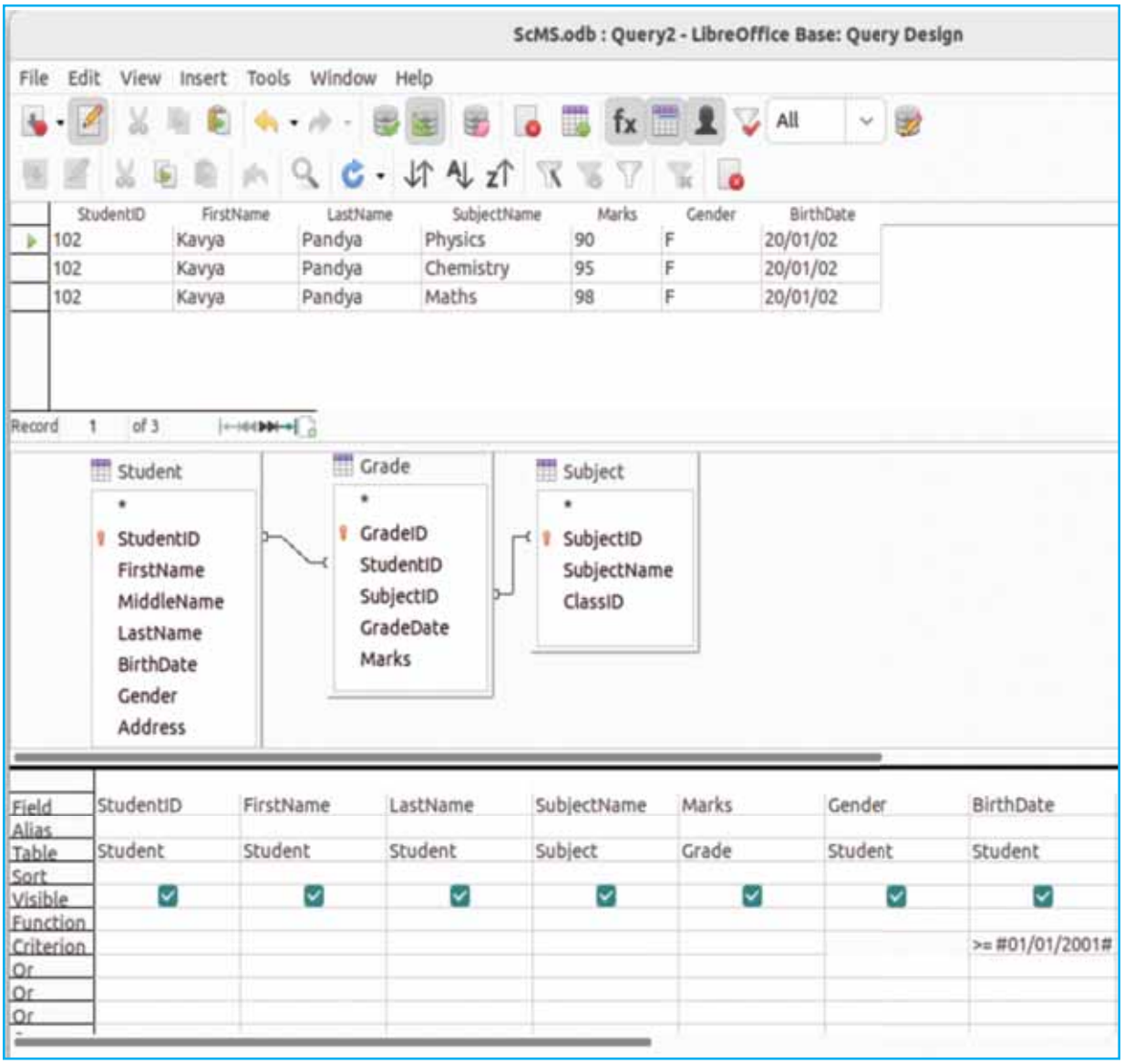

Figure 3.13 : Applying Criteria in Date Field

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.

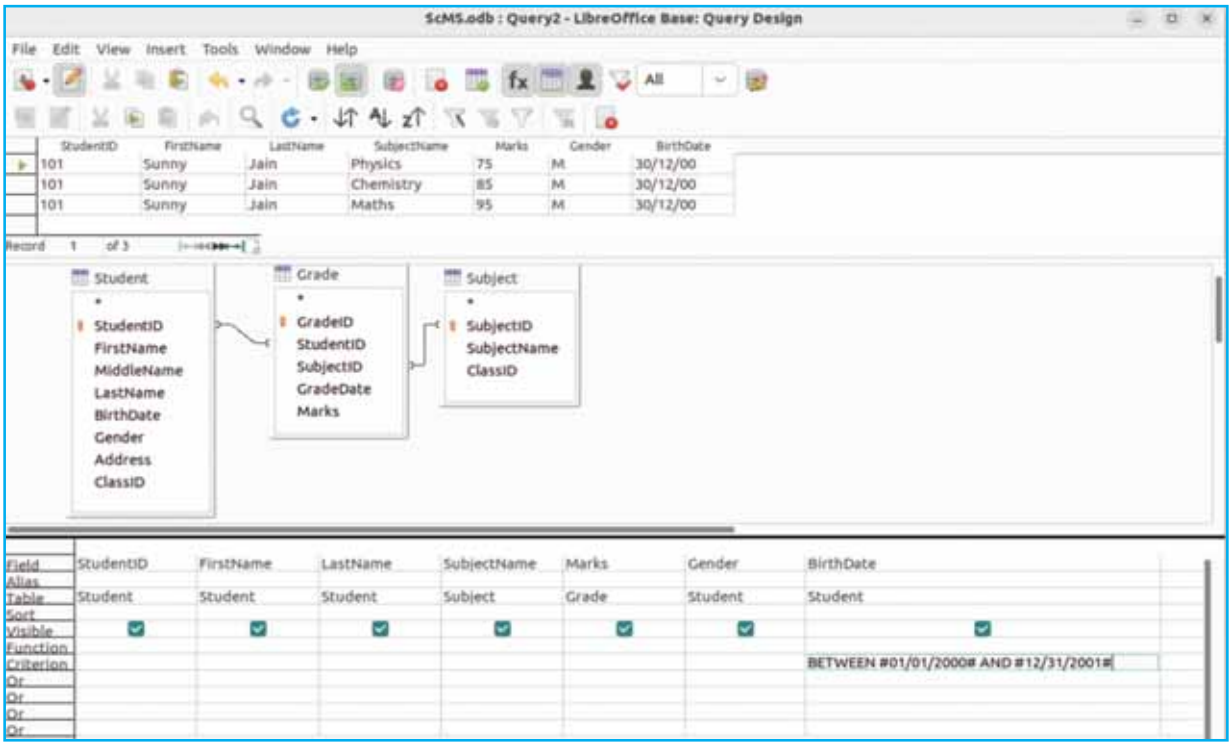

Figure 3.14 : Using the Between Operator

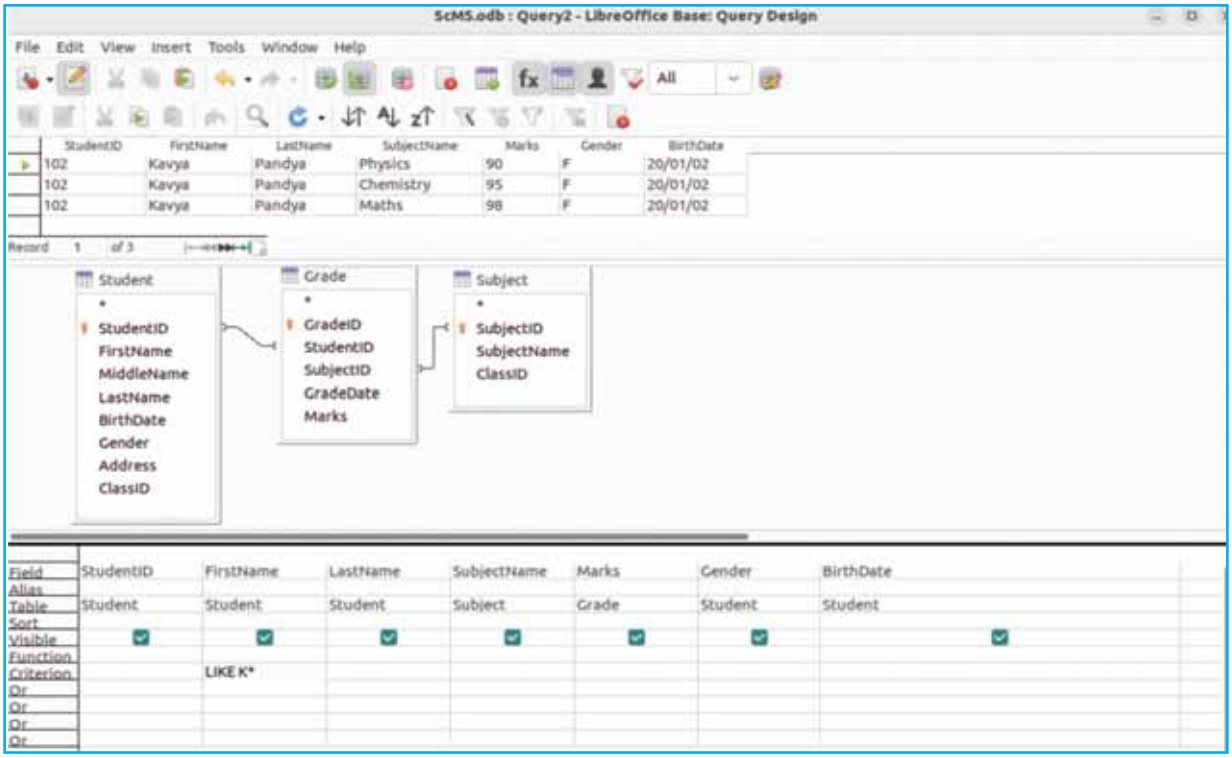

Figure 3.15 : Wild Cards

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.

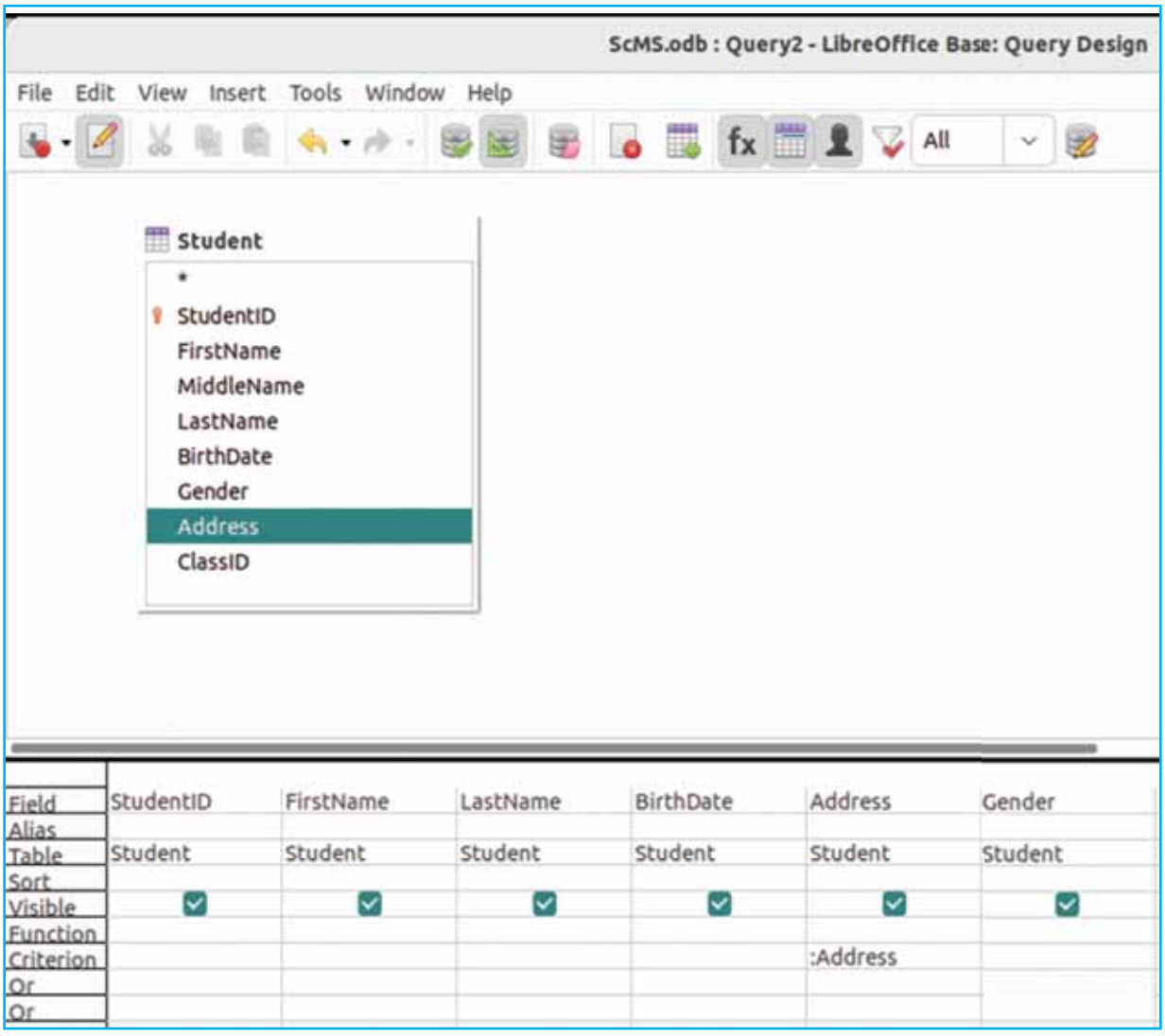

Figure 3.16 : Parameterized Query

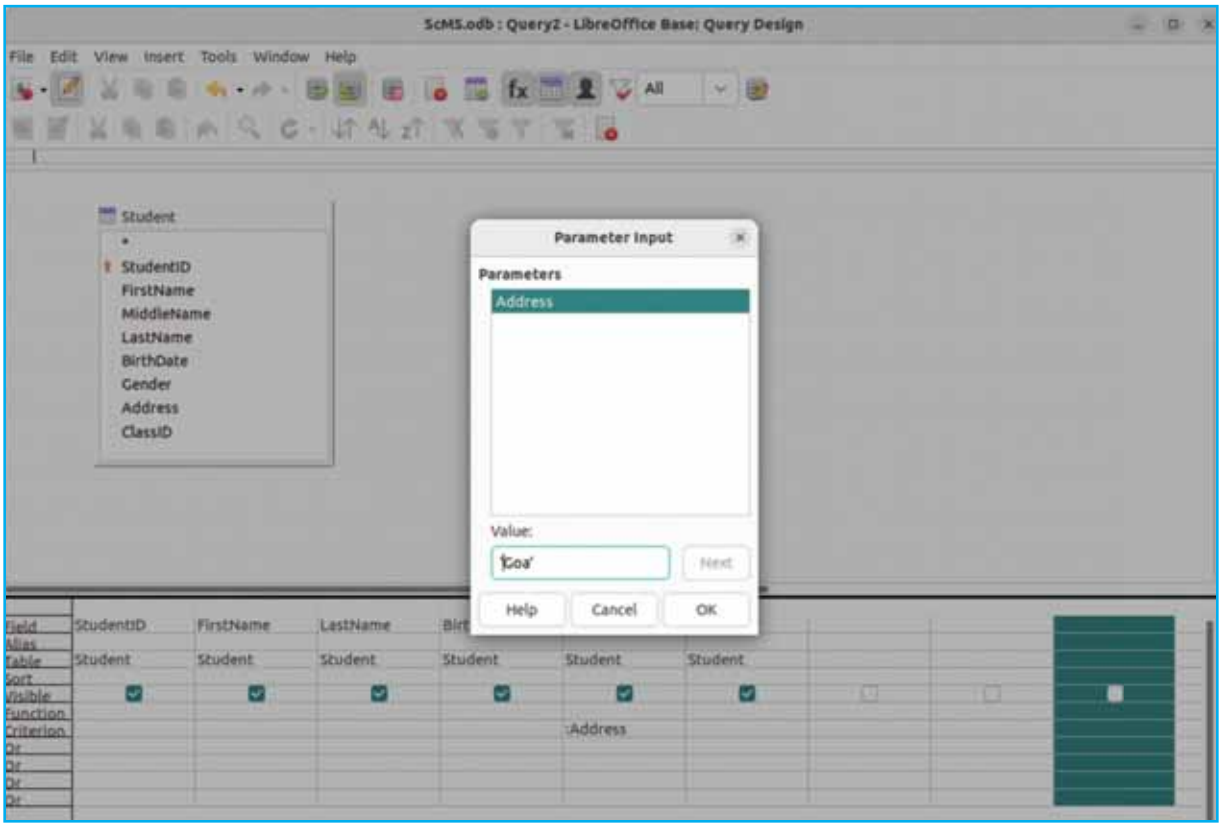

Figure 3.17 : Parameter Value

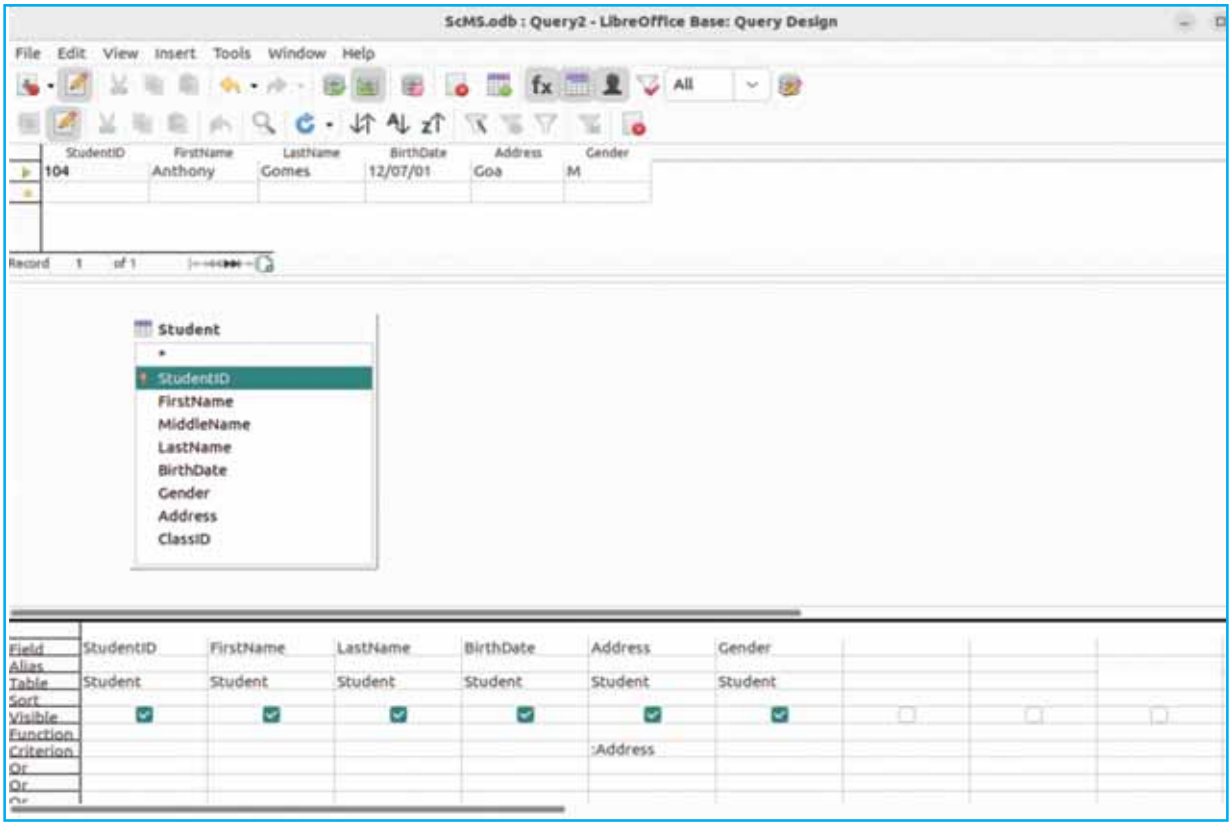

Figure 3.18 : Output

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.