Unit IV

CHAPTER 12
STRUCTURED QUERY LANGUAGE (SQL)

OPE46

Learning Objectives

After studying this lesson, students will be able to:

- The processing skills of SQL.
- The components of SQL.
- To create a table by specifying the fields and records.
- To apply various manipulations like inserting, updating and deleting records in a table.
- To learn about various constraints and to apply it on tables.
- To generate queries in the table by applying various clauses.
- To modify the structure of an existing table.
- The commands to delete records, table and revoke the commands.

12.1 Introduction to SQL

The Structured Query Language (SQL) is a standard programming language to access and manipulate databases. SQL allows the user to create, retrieve, alter, and transfer information among databases. It is a language designed for managing and accessing data in a Relational Data Base Management System (RDBMS).

There are many versions of SQL. The original version was developed at IBM's Research centre and originally called as Sequel in early 1970's. Later the language was changed to SQL. In 1986, ANSI (American National Standard Institute) published an SQL standard that was updated again in 1992, the latest SQL was released in 2008 and named as SQL 2008.

Note

Latest SQL standard as of now is SQL 2008, released in 2008.

12.2 Role of SQL in RDBMS

RDBMS stands for Relational DataBase Management System. Oracle, MySQL, MS SQL Server, IBM DB2 and Microsoft Access are RDBMS packages. SQL is a language used to access data in such databases.

In general, Database is a collection of tables that store sets of data that can be queried for use in other applications. A database management system supports the development, administration and use of database platforms. RDBMS is a type of DBMS with a row-based table structure that connects related data elements and includes functions related to Create, Read, Update and Delete operations, collectively known as CRUD.

The data in RDBMS, is stored in database objects, called Tables. A table is a collection of related data entries and it consist of rows and columns.

A field is a column in a table that is designed to maintain specific related information about every record in the table. It is a vertical entity that contains all information associated with a specific field in a table. The fields in a student table may be of the type AdmnNo, StudName, StudAge, StudClass, Place etc.

A Record is a row, which is a collection of related fields or columns that exist in a table. A record is a horizontal entity in a table which represents the details of a particular student in a student table.

12.3 Processing Skills of SQL

The various processing skills of SQL are :

1. **Data Definition Language (DDL) :** The SQL DDL provides commands for defining relation schemas (structure), deleting relations, creating indexes and modifying relation schemas.
2. **Data Manipulation Language (DML)** : The SQL DML includes commands to insert, delete, and modify tuples in the database.
3. **Embedded Data Manipulation Language :** The embedded form of SQL is used in high level programming languages.
4. **View Definition :** The SQL also includes commands for defining views of tables.
5. **Authorization :** The SQL includes commands for access rights to relations and views of tables.
6. **Integrity :** The SQL provides forms for integrity checking using condition.
7. **Transaction control :** The SQL includes commands for file transactions and control over transaction processing.

DO YOU KNOW?

SQL-Structured Query Language is a language used for accessing databases while **MySQL** is a database management system, like SQL Server, Oracle, Informix, Postgres, etc. **MySQL** is a **RDBMS**.

Refer installing MYSQL in Annexure -1

12.4 Creating Database

1. To create a database, type the following command in the prompt:

   CREATE DATABASE database_name;

   For example to create a database to store the tables:

   CREATE DATABASE stud;

2. To work with the database, type the following command.

   USE DATABASE;

   For example to use the stud database created, give the command

   USE stud;

DO YOU KNOW?

WAMP stands for **"Windows, Apache, MySQL** and **PHP". WAMP** is a variation of LAMP for windows systems and is installed as a software bundle (Apache, MySQL and PHP). It is often used for web development and internal testing, but may also be used to serve live websites.

12.5 Components of SQL

SQL commands are divided into five categories:

- DML - Data Manipulation Language
- DDL - Data Definition Language
- DCL - Data Control Language
- TCL - Transaction Control Language
- DQL - Data Query Language

12.5.1 DATA DEFINITION LANGUAGE

The Data Definition Language (DDL) consist of SQL statements used to define the database structure or schema. It simply deals with descriptions of the database schema and is used to create and modify the structure of database objects in databases.

The DDL provides a set of definitions to specify the storage structure and access methods used by the database system.

A DDL performs the following functions :

1. It should identify the type of data division such as data item, segment, record and database file.
2. It gives a unique name to each data item type, record type, file type and data base.
3. It should specify the proper data type.
4. It should define the size of the data item.
5. It may define the range of values that a data item may use.
6. It may specify privacy locks for preventing unauthorized data entry.

SQL commands which comes under Data Definition Language are:

Create	To create tables in the database.
Alter	Alters the structure of the database.
Drop	Delete tables from database.
Truncate	Remove all records from a table, also release the space occupied by those records.

12.5.2 DATA MANIPULATION LANGUAGE

A Data Manipulation Language (DML) is a computer programming language used for adding (inserting), removing (deleting), and modifying (updating) data in a database. In SQL, the data manipulation language comprises the SQL-data change statements, which modify stored data but not the schema of the database table.

After the database schema has been specified and the database has been created, the data can be manipulated using a set of procedures which are expressed by DML.

By Data Manipulation we mean,

- Insertion of new information into the database
- Retrieval of information stored in a database.
- Deletion of information from the database.
- Modification of data stored in the database.

The DML is basically of two types:

Procedural DML – Requires a user to specify what data is needed and how to get it.

Non-Procedural DML - Requires a user to specify what data is needed without specifying how to get it.

SQL commands which comes under Data Manipulation Language are :

Insert	Inserts data into a table
Update	Updates the existing data within a table.
Delete	Deletes all records from a table, but not the space occupied by them.

12.5.3 DATA CONTROL LANGUAGE

A **Data Control Language (DCL)** is a programming language used to control the access of data stored in a database. It is used for controlling privileges in the database (Authorization). The privileges are required for performing all the database operations such as creating sequences, views of tables etc.

SQL commands which come under Data Control Language are:

Grant	Grants permission to one or more users to perform specific tasks.
Revoke	Withdraws the access permission given by the GRANT statement.

12.5.4 TRANSACTIONAL CONTROL LANGUAGE

Transactional control language (TCL) commands are used to manage transactions in the database. These are used to manage the changes made to the data in a table by DML statements.

SQL command which come under Transfer Control Language are:

Commit	Saves any transaction into the database permanently.
Roll back	Restores the database to last commit state.
Save point	Temporarily save a transaction so that you can rollback.

12.5.5 DATA QUERY LANGUAGE

The Data Query Language consist of commands used to query or retrieve data from a database. One such SQL command in Data Query Language is

Select	It displays the records from the table.

12.6 Data Types

The data in a database is stored based on the kind of value stored in it. This is identified as the data type of the data or by assigning each field a data type. All the values in a given field must be of same type.

The ANSI SQL standard recognizes only Text and Number data type, while some commercial programs use other datatypes like Date and Time etc. The ANSI data types are listed below in Table 12.1

Data Type	Description
char (Character)	Fixed width string value. Values of this type is enclosed in single quotes. For ex. Anu's will be written as 'Anu' 's'.
varchar	Variable width character string. This is similar to char except the size of the data entry vary considerably.
dec (Decimal)	It represents a fractional number such as 15.12, 0.123 etc. Here the size argument consist of two parts : precision and scale. The precision indicates how many digits the number may have and the scale indicates the maximum number of digits to the right of the decimal point. The size (5, 2) indicates precision as 5 and scale as 2. The scale cannot exceed the precision.
numeric	It is same as decimal except that the maximum number of digits may not exceed the precision argument.
int (Integer)	It represents a number without a decimal point. Here the size argument is not used.
smallint	It is same as integer but the default size may be smaller than Integer.
float	It represents a floating point number in base 10 exponential notation and may define a precision up to a maximum of 64.
real	It is same as float, except the size argument is not used and may define a precision up to a maximum of 64.
double	Same as real except the precision may exceed 64.

Table 12.1

12.7 SQL Commands and their Functions

Tables are the only way to store data, therefore all the information has to be arranged in the form of tables. The SQL provides a predetermined set of commands to work on databases.

Keywords	They have a special meaning in SQL. They are understood as instructions.
Commands	They are instructions given by the user to the database also known as statements.
Clauses	They begin with a keyword and consist of keyword and argument.

Arguments	They are the values given to make the clause complete.

12.7.1 DDL Commands

CREATE TABLE Command

You can create a table by using the **CREATE TABLE** command. When a table is created, its columns are named, data types and sizes are to be specified. Each table must have at least one column. The syntax of **CREATE TABLE** command is :

```
CREATE TABLE <table-name>
(<column name><data type>[<size>]
(<column name><data type>[<size>]......
);
```

Now let us use the above syntax to store some information about the students of a class in a database, for this you first need to create table. To create a student table, let us take some information related to students like admission number which we can use it in short form as (admno), name of student (name), gender, age etc. of the student. Let us create a table having the field names Admno, Name, Gender, Age and Place.

The SQL command will be as follows:

```
CREATE TABLE Student

    (Admno    integer,
       Name  char(20),
       Gender  char(1),
       Age      integer,
       Place char(10),
       );
```

The above one is a simple table structure without any restrictions. You can also set constraints to limit the type of data that can go into the fields of the table. Constraints are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be either on a column level or a table level.

Note

Constraint is a condition applicable on a field or set of fields.

Column constraint:	Column constraint apply only to individual column.
Table constraint :	Table constraint apply to a group of one or more columns.

The syntax for a table created with constraint is given as below:

```
CREATE TABLE <table-name>
(<column name><data type>[<size>]<column constraint>,
(<column name><data type>[<size>]<column constraint>……
<table constraint>(<column name>,[<column name>….]).….
);
```

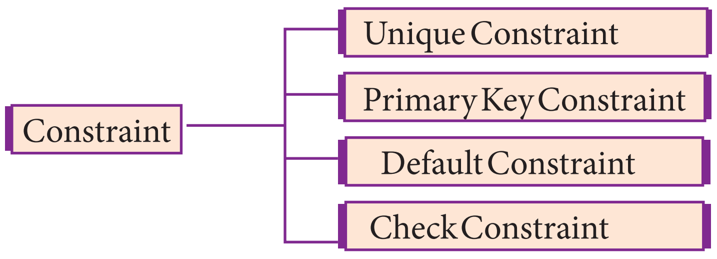
Following is an example for student table with **"NOT NULL"** column constraint. This constraint enforces a field to always contain a value.

```
CREATE TABLE Student
(
Admno integer   NOT NULL PRIMARY KEY, → Primary Key constraint
Name char(20)  NOT NULL,
Gender char(1),
Age integer,
Place char(10),
);
```

The above command creates a table "student" in which the field Admno of integer type is defined NOT NULL, Name of char type is defined as NOT NULL which means these two fields must have values. The fields Gender, Age and Place do not have any constraints.

12.7.2 Type of Constraints

Constraints ensure database integrity, therefore known as database integrity constraints. The different types of constraints are :



12.7.2.1 Unique Constraint

This constraint ensures that no two rows have the same value in the specified columns. For example **UNIQUE** constraint applied on Admno of student table ensures that no two students have the same admission number and the constraint can be used as:

```
CREATE TABLE Student
(
Admno integer NOT NULL UNIQUE, → Unique constraint
Name char (20) NOT NULL,
Gender char (1),
Age integer,
Place char (10),
);
```

The **UNIQUE** constraint can be applied only to fields that have also been declared as **NOT NULL**.

When two constraints are applied on a single field, it is known as multiple constraints. In the above Multiple constraints **NOT NULL** and **UNIQUE** are applied on a single field Admno, the constraints are separated by a space and at the end of the field definition a comma(,) is added. By adding these two constraints the field Admno must take some value ie. will not be **NULL** and should not be duplicated.

12.7.2.2 Primary Key Constraint

This constraint declares a field as a Primary key which helps to uniquely identify a record. It is similar to unique constraint except that only one field of a table can be set as primary key. The primary key does not allow **NULL** values.

Example showing Primary Key Constraint in the student table:

```
CREATE TABLE Student
(
Admno integer   PRIMARY KEY, → Primary Key constraint
Name char(20)  NOT NULL,
Gender char(1),
Age integer,
Place char(10),
);
```

In the above example the Admno field has been set as primary key and therefore will help us to uniquely identify a record, it is also set **NOT NULL**, therefore this field value cannot be empty.

12.7.2.3 DEFAULT Constraint

The **DEFAULT** constraint is used to assign a default value for the field. When no value is given for the specified field having **DEFAULT** constraint, automatically the default value will be assigned to the field.

Example showing **DEFAULT** Constraint in the student table:

```
CREATE TABLE Student
(
Admno integer    PRIMARY KEY,
Name char(20)NOT NULL,
Gender char(1),
Age integer  DEFAULT 17, → Default Constraint
Place char(10),
);
```

In the above example the "Age" field is assigned a default value of 17, therefore when no value is entered in age by the user, it automatically assigns 17 to Age.

12.7.2.4 Check Constraint

This constraint helps to set a limit value placed for a field. When we define a check constraint on a single column, it allows only the restricted values on that field. Example showing check constraint in the student table:

```
CREATE TABLE Student
(
Admno integer   PRIMARY KEY
Name char(20)NOT NULL,
Gender char(1),
Age integer CHECK (Age <=19), → Check Constraint
Place char(10),
);
```

In the above example the check constraint is set to Age field where the value of Age must be less than or equal to 19.

Note

The check constraint may use relational and logical operators for condition.

12.7.2.5 TABLE CONSTRAINT

When the constraint is applied to a group of fields of the table, it is known as Table constraint. The table constraint is normally given at the end of the table definition. Let us take a new table namely Student1 with the following fields Admno, Firstname, Lastname, Gender, Age, Place:

```
CREATE TABLE Student 1
(
Admno integer        NOT NULL,
Firstname char(20),
Lastname char(20),
Gender char(1),
Age integer,
Place char(10),
PRIMARY KEY (Firstname, Lastname) → Table constraint
);
```

In the above example, the two fields, Firstname and Lastname are defined as Primary key which is a Table constraint.

12.7.3 DML COMMANDS

Once the schema or structure of the table is created, values can be added to the table. The DML commands consist of inserting, deleting and updating rows into the table.

12.7.3.1 INSERT command

The **INSERT** command helps to add new data to the database or add new records to the table. The command is used as follows:

```
INSERT INTO <table-name> [column-list] VALUES (values);
```

INSERT INTO Student (Admno, Name, Gender, Age, Place)

VALUES (100,' Ashish',' M', 17,' Chennai');

INSERT INTO Student (Admno, Name, Gender, Age, Place)

VALUES (101, 'Adarsh', 'M', 18, 'Delhi');

Two new records are added to the table as shown below:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi

The order of values must match the order of columns in the **CREATE TABLE** command. Specifying the column names is optional if data is to be added for all columns. The command to add values into the student table can also be used in the following way:

INSERT INTO Student VALUES (102, 'Akshith', 'M', '17,' 'Bangalore');

102	Akshith	M	17	Bangalore

The above command inserts the record into the student table.

To add data to only some columns in a record by specifying the column name and their data, it can be done by:

INSERT INTO Student(Admno, Name, Place) VALUES (103, 'Ayush', 'Delhi');

103	Ayush	M	18	Delhi

The above command adds the following record with default values of 'M' for Gender and Age as 18.

INSERT INTO Student (Admno, Name, Place) VALUES (104, 'Abinandh', 'Chennai');

104	Abinandh	M	18	Chennai

The student table will have the following data:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	17	Bangalore
103	Ayush	M	18	Delhi
104	Abinandh	M	18	Chennai

The fields that are not given in the INSERT command will take default values, if it is defined for them, otherwise NULL value will be stored.

DO YOU KNOW?

In the INSERT command the fields that are omitted will have either default value defined or NULL value.

12.7.3.2 DELETE COMMAND

The **DELETE** command permanently removes one or more records from the table. It removes the entire row, not individual fields of the row, so no field argument is needed. The **DELETE** command is used as follows :

DELETE FROM table-name WHERE condition;

For example to delete the record whose admission number is 104 the command is given as follows:

DELETE FROM Student WHERE Admno=104;

104	Abinandh	M	18	Chennai

The following record is deleted from the Student table.

To delete all the rows of the table, the command is used as :

DELETE FROM Student;

The table will be empty now and could be destroyed using the DROP command (Discussed in section 12.7.4.3).

12.7.3.3 UPDATE COMMAND

The **UPDATE** command updates some or all data values in a database. It can update one or more records in a table. The **UPDATE** command specifies the rows to be changed using the **WHERE** clause and the new data using the **SET** keyword. The command is used as follows:

UPDATE <table-name> SET column-name = value, column-name = value,… WHERE condition;

For example to update the following fields:

UPDATE Student SET Age = 20 WHERE Place = 'Bangalore';

The above command will change the age to 20 for those students whose place is "Bangalore".

The table will be as updated as below:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	20	Bangalore
103	Ayush	M	18	Delhi

To update multiple fields, multiple field assignment can be specified with the **SET** clause separated by comma. For example to update multiple fields in the Student table, the command is given as:

UPDATE Student SET Age=18, Place = 'Chennai' WHERE Admno = 102;

102	Akshith	M	18	Chennai

The above command modifies the record in the following way.

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	18	Chennai
103	Ayush	M	18	Delhi

12.7.4 Some Additional DDL Commands:

12.7.4.1 ALTER COMMAND

The **ALTER** command is used to alter the table structure like adding a column, renaming the existing column, change the data type of any column or size of the column or delete the column from the table. It is used in the following way :

ALTER TABLE <table-name> ADD <column-name><data type><size>;

To add a new column "Address" of type 'char' to the Student table, the command is used as

ALTER TABLE Student ADD Address char;

To modify existing column of table, the **ALTER TABLE** command can be used with **MODIFY** clause like wise:

> *ALTER TABLE <table-name> MODIFY<column-name><data type><size>;*

ALTER TABLE Student MODIFY Address char (25);

The above command will modify the address column of the Student table to now hold 25 characters.

The **ALTER** command can be used to rename an existing column in the following way :

> *ALTER TABLE <table-name> CHANGE old-column-name new-column-name new column definition;*

For example to rename the column Address to City, the command is used as :

ALTER TABLE Student CHANGE Address City char(20);

The **ALTER** command can also be used to remove a column or all columns, for example to remove a particular column, the **DROP COLUMN** is used with the **ALTER TABLE** to remove a particular field. The command can be used as:

> *ALTER TABLE <table-name> DROP COLUMN <column-name>;*

To remove the column City from the Student table, the command is used as :

ALTER TABLE Student DROP COLUMN City;

12.7.4.2 TRUNCATE command

The **TRUNCATE** command is used to delete all the rows from the table, the structure remains and the space is freed from the table. The syntax for **TRUNCATE** command is:

> *TRUNCATE TABLE table-name;*

For example to delete all the records of the student table and delete the table the SQL statement is given as follows:

TRUNCATE TABLE Student;

The table Student is removed and the space is freed.

12.7.4.3 DROP TABLE command

The **DROP TABLE** command is used to remove a table from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back, so be careful while using **DROP TABLE** command.

To delete all rows, the command is given as :

DELETE FROM Student;

Once all the rows are deleted, the table can be deleted by **DROP TABLE** command in the following way:

```
DROP TABLE table-name;
```

For example to delete the Student table:

DROP TABLE Student;

DELETE, TRUNCATE AND DROP statement:

DELETE	The **DELETE** command deletes only the rows from the table based on the condition given in the where clause or deletes all the rows from the table if no condition is specified. But it does not free the space containing the table.
TRUNCATE	The **TRUNCATE** command is used to delete all the rows, the structure remains in the table and free the space containing the table.
DROP	The **DROP** command is used to remove an object from the database. If you drop a table, all the rows in the table is deleted and the table structure is removed from the database. Once a table is dropped we cannot get it back.

12.7.5 DQL COMMAND– SELECT command

One of the most important tasks when working with SQL is to generate Queries and retrieve data. A Query is a command given to get a desired result from the database table. The **SELECT** command is used to query or retrieve data from a table in the database. It is used to retrieve a subset of records from one or more tables. The **SELECT** command can be used in various forms:

Syntax of **SELECT** command :

```
SELECT <column-list>FROM<table-name>;
```

- Table-name is the name of the table from which the information is retrieved.
- Column-list includes one or more columns from which data is retrieved.

For example to view only admission number and name of students from the Student table the command is given as follows:

If the Student table has the following data:

Admno	Name	Gender	Age	Place
100	Ashish	M	17	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	17	Bangalore
103	Ayush	M	18	Delhi
104	Abinandh	M	18	Chennai
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
107	Hema	F	17	Chennai

SELECT Admno, Name FROM Student;

The above **SELECT** command will display the following data:

Admno	Name
100	Ashish
101	Adarsh
102	Akshith
103	Ayush
104	Abinandh
105	Revathi
106	Devika
107	Hema

To view all the fields and rows of the table the **SELECT** command can be given as

SELECT * FROM STUDENT;

12.7.5.1 DISTINCT Keyword

The **DISTINCT** keyword is used along with the **SELECT** command to eliminate duplicate rows in the table. This helps to eliminate redundant data. For Example:

SELECT DISTINCT Place FROM Student;

Will display the following data as follows :

Place
Chennai
Bangalore
Delhi

In the above output you can see, there would be no duplicate rows in the place field. When the keyword **DISTINCT** is used, only one **NULL** value is returned, even if more **NULL** values occur.

12.7.5.2 ALL Keyword

The **ALL** keyword retains duplicate rows. It will display every row of the table without considering duplicate entries.

SELECT ALL Place FROM Student;

The above command will display all values of place field from every row of the table without eliminating the duplicate entries.

Place
Chennai
Delhi
Bangalore
Delhi
Chennai
Chennai
Bangalore
Chennai

The **WHERE** clause in the **SELECT** command specifies the criteria for getting the desired result. The general form of **SELECT** command with **WHERE** Clause is:

SELECT <column-name>[,<column-name>,….] FROM <table-name>WHERE condition>;

For example to display the students admission number and name of only those students who belong to Chennai, the **SELECT** command is used in the following way :

SELECT Admno, Name, Place FROM Student WHERE Place = ʻChennaiʼ;

Admno	Name	Place
100	Ashish	Chennai
104	Abinandh	Chennai
105	Revathi	Chennai
107	Hema	Chennai

SELECT Admno, Name, Age FROM Student WHERE Age >= 18;

Admno	Name	Age
101	Adarsh	18
103	Ayush	18
104	Abinandh	18
105	Revathi	19
106	Devika	19

The relational operators like =, <, <=, >, >=, <> can be used to compare two values in the **SELECT** command used with **WHERE** clause. The logical operaors **OR, AND** and **NOT** can also be used to connect search conditions in the **WHERE** clause. For example :

SELECT Admno, Name, Age, Place FROM Student WHERE (Age>=18 AND Place = 'Delhi');

Admno	Name	Age	Place
101	Adarsh	18	Delhi
103	Ayush	18	Delhi

The **SELECT** command can also be used in the following ways:

SELECT Admno, Name, Age, Place FROM Student WHERE (Age>=18 OR Place ='Delhi');

SELECT Admno, Name, Place FROM Student WHERE (NOT Place ='Delhi');

12.7.5.3 BETWEEN and NOT BETWEEN Keywords

The **BETWEEN** keyword defines a range of values the record must fall into to make the condition true. The range may include an upper value and a lower value between which the criteria must fall into.

SELECT Admno, Name, Age, Gender FROM Student WHERE Age BETWEEN 18 AND 19;

Admno	Name	Age	Gender
101	Adarsh	18	M
103	Ayush	18	M
104	Abinandh	18	M
105	Revathi	19	F
106	Devika	19	F

The **NOT BETWEEN** is reverse of the **BETWEEN** operator where the records not satisfying the condition are displayed.

SELECT Admno, Name, Age FROM Student WHERE Age NOT BETWEEN 18 AND 19;

Admno	Name	Age
100	Ashish	17
102	Akshith	17
107	Hema	17

12.7.5.4 IN Keyword

The **IN** keyword is used to specify a list of values which must be matched with the record values. In other words it is used to compare a column with more than one value. It is similar to an **OR** condition.

For example :

SELECT Admno, Name, Place FROM Student WHERE Place IN ('Chennai', 'Delhi');

Admno	Name	Place
100	Ashish	Chennai
101	Adarsh	Delhi
103	Ayush	Delhi
104	Abinandh	Chennai
105	Revathi	Chennai
107	Hema	Chennai

The **NOT IN** keyword displays only those records that do not match in the list.

For example:

SELECT Admno, Name, Place FROM Student WHERE Place NOT IN ('Chennai', 'Delhi');

will display students only from places other than "Chennai" and "Delhi".

Admno	Name	Place
102	Akshith	Bangalore
106	Devika	Bangalore

NULL Value :

The **NULL** value in a field can be searched in a table using the **IS NULL** in the **WHERE** clause. For example to list all the students whose Age contains no value, the command is used as:

SELECT * FROM Student WHERE Age IS NULL;

Note

Non NULL values in a table can be listed using IS NOT NULL.

12.7.5.5 ORDER BY clause

The **ORDER BY** clause in SQL is used to sort the data in either ascending or descending based on one or more columns.

1. By default **ORDER BY** sorts the data in ascending order.
2. We can use the keyword **DESC** to sort the data in descending order and the keyword **ASC** to sort in ascending order.

The ORDER BY clause is used as :

SELECT <column-name>[,<column-name>,....] FROM <table-name>ORDER BY <column1>,<column2>,...ASC| DESC ;

For example :

To display the students in alphabetical order of their names, the command is used as

SELECT * FROM Student ORDER BY Name;

The above student table is arranged as follows :

Admno	Name	Gender	Age	Place
104	Abinandh	M	18	Chennai
101	Adarsh	M	18	Delhi
102	Akshith	M	17	Bangalore
100	Ashish	M	17	Chennai
103	Ayush	M	18	Delhi
106	Devika	F	19	Bangalore
107	Hema	F	17	Chennai
105	Revathi	F	19	Chennai

Note

The ORDER BY clause does not affect the original table.

12.7.5.6 WHERE clause

The **WHERE** clause is used to filter the records. It helps to extract only those records which satisfy a given condition. For example in the student table, to display the list of students of age18 and above in alphabetical order of their names, the command is given as below:

SELECT * FROM Student WHERE Age>=18 ORDER BY Name;

Admno	Name	Gender	Age	Place
104	Abinandh	M	18	Chennai
101	Adarsh	M	18	Delhi
103	Ayush	M	18	Delhi
106	Devika	F	19	Bangalore
105	Revathi	F	19	Chennai

To display the list of students in the descending order of names of those students of age 18 and above the command is given as :

SELECT * FROM Student WHERE Age>=18 ORDER BY Name DESC;

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai

Note
Sorting can be done on multiple fields.

12.7.5.7 GROUP BY clause

The **GROUP BY** clause is used with the **SELECT** statement to group the students on rows or columns having identical values or divide the table in to groups. For example to know the number of male students or female students of a class, the **GROUP BY** clause may be used. It is mostly used in conjunction with aggregate functions to produce summary reports from the database.

The syntax for the **GROUP BY** clause is

SELECT <column-names> FROM <table-name> GROUP BY <column-name>HAVING condition];

To apply the above command on the student table :

SELECT Gender FROM Student GROUP BY Gender;

The following command will give the below given result:

Gender
M
F

The point to be noted is that only two results have been returned. This is because we only have two gender types 'Male' and 'Female'. The GROUP BY clause grouped all the 'M' students together and returned only a single row for it. It did the same with the 'F' students.

For example to count the number of male and female students in the student table, the following command is given :

SELECT Gender, count(*) FROM Student GROUP BY Gender;

Gender	count(*)
M	5
F	3

Note

The * is used with the COUNT to include the NULL values.

The GROUP BY applies the aggregate functions independently to a series of groups that are defined by having a field value in common. The output of the above SELECT statement gives a count of the number of Male and Female students.

12.7.5.8 HAVING clause

The **HAVING** clause can be used along with GROUP BY clause in the SELECT statement to place condition on groups and can include aggregate functions on them. For example to count the number of Male and Female students having age greater thanor equal to 18.

SELECT Gender , FROM Student GROUP BY Gender HAVING count(*)> = 18;

Gender	count(*)
M	2
F	2

The above output shows the gender which has the count greater than 20.

12.7.6 TCL commands

12.7.6.1 COMMIT command

The **COMMIT** command is used to permanently save any transaction to the database. When any DML commands like **INSERT, UPDATE, DELETE** commands are used, the changes made by these commands are not permanent. It is marked permanent only after the **COMMIT** command is given from the SQL prompt. Once the **COMMIT** command is given, the changes made cannot be rolled back. The **COMMIT** command is used as

COMMIT;

12.7.6.2 ROLLBACK command

The **ROLLBACK** command restores the database to the last commited state. It is used with **SAVEPOINT** command to jump to a particular savepoint location. The syntax for the **ROLLBACK** command is :

ROLL BACK TO save point name;

12.7.6.3 SAVEPOINT command

The **SAVEPOINT** command is used to temporarily save a transaction so that you can rollback to the point whenever required. The different states of our table can be saved at anytime using different names and the rollback to that state can be done using the **ROLLBACK** command.

```
SAVEPOINT savepoint_name;
```

Example showing **COMMIT, SAVEPOINT** and **ROLLBACK** in the student table having the following data:

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai

```
INSERT INTO Student  VALUES (107, 'Beena', 'F', 20 , 'Cochin');
COMMIT;
```

Admno	Name	Gender	Age	Place
105	Revathi	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai
107	Beena	F	20	Cochin

```
UPDATE Student SET Name = ‘Mini’ WHERE  Admno=105;
SAVEPOINT A;
```

Admno	Name	Gender	Age	Place
105	Mini	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai
107	Beena	F	20	Cochin

INSERT INTO Student VALUES(108, 'Jisha', 'F', 19, 'Delhi');

SAVEPOINT B;

Admno	Name	Gender	Age	Place
105	Mini	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	vAbinandh	M	18	Chennai
107	Beena	F	20	Cochin
108	Jisha	F	19	Delhi

ROLLBACK TO A;

Admno	Name	Gender	Age	Place
105	Mini	F	19	Chennai
106	Devika	F	19	Bangalore
103	Ayush	M	18	Delhi
101	Adarsh	M	18	Delhi
104	Abinandh	M	18	Chennai
107	Beena	F	20	Cochin

Points to remember:

- ***SQL*** is a language that helps to create and operate relational databases.
- ***MySQL*** is a database management system.
- The various components of ***SQL*** are Data Definition Language (***DDL***), Data Manipulation Language (***DML***), Data Query Language (***DQL***), Transactional Control Language (***TCL***), Data Control Language (***DCL***).
- The ***DDL*** provides statements for creation and deletion of tables.
- The ***DML*** provides statements to insert, update and delete data of a table.
- The ***DCL*** provides authorization commands to access data.
- The ***TCL*** commands are used to manage transactions in a database.
- The ***DQL*** commands help to generate queries in a database.
- The ***CREATE TABLE*** command creates a new table.

Hands on Experience

1. Create a query of the student table in the following order of fields name, age, place and admno.
2. Create a query to display the student table with students of age more than 18 with unique city.

Hands on Experience

1. Create a employee table with the following fields employee number, employee name, designation, date of joining and basic pay.
2. In the above table set the employee number as primary key and check for NULL values in any field.
3. Prepare a list of all employees who are Managers.

2UG94

Evaluation

Part - I

I Choose the best answer 1X 5 = 5

1. Which commands provide definitions for creating table structure, deleting relations, and modifying relation schemas.

   a. DDL b. DML

   c. DCL d. DQL

2. Which command lets to change the structure of the table?

   a. SELECT b. ORDER BY

   c. MODIFY d. ALTER

3. The command to delete a table including the structure is

   A) DROP B) DELETE

   C) DELETE ALL D) ALTER TABLE

4. Queries can be generated using

   a. SELECT b. ORDER BY

   c. MODIFY d. ALTER

5. The clause used to sort data in a database

   a. SORT BY b. ORDER BY

   c. GROUP BY d. SELECT

Part -II

II Write Short Answers 2 x 5 = 10

1. Write a query that selects all students whose age is less than 18 in order wise.
2. Differentiate Unique and Primary Key constraint.
3. Write the difference between table constraint and column constraint?
4. Which component of SQL lets insert values in tables and which lets to create a table?
5. What is the difference between SQL and MySQL?

Part -III

III Explain 3 x 5 = 15

1. What is a constraint? Write short note on Primary key constraint.
2. Write a SQL statement to modify the student table structure by adding a new field.
3. Write any three DDL commands.
4. Write the use of Savepoint command with an example.
5. Write a SQL statement using DISTINCT keyword.

Part -IV

IV Explain in Detail 5 x 5 = 25

1. Write the different types of constraints and their functions.
2. Consider the following employee table. Write SQL commands for the qtns.(i) to (v).

EMP CODE	NAME	DESIG	PAY	ALLO WANCE
S1001	Hariharan	Supervisor	29000	12000
P1002	Shaji	Operator	10000	5500
P1003	Prasad	Operator	12000	6500
C1004	Manjima	Clerk	8000	4500
M1005	Ratheesh	Mechanic	20000	7000

(i) To display the details of all employees in descending order of pay.

(ii) To display all employees whose allowance is between 5000 and 7000.

(iii) To remove the employees who are mechanic.

(iv) To add a new row.

(v) To display the details of all employees who are operators.

3. What are the components of SQL? Write the commands in each.
4. Construct the following SQL statements in the student table-
   (i) SELECT statement using GROUP BY clause.
   (ii) SELECT statement using ORDER BY clause.
5. Write a SQL statement to create a table for employee having any five fields and create a table constraint for the employee table.