Unit 1 :

Introduction to SQL

Que 1: Introduction SQL

- SQL (Structured Query Language) is used to perform operations on the records stored in the database, such as updating records, inserting records, deleting records, creating and modifying database tables, views, etc.
- SQL is not a database system, but it is a query language.
- SQL has two broad categories:
    - **DDL (Data Definition Language):**used for defining, managing, and manipulating the structure of a relational database. DDL statements are primarily concerned with the creation, modification, and deletion of database objects such as tables, indexes, views, and constraints. The data definition contains commands like create, drop, truncate, alter and rename .
    - **DML(Data Manipulation Language):** DML statements are used to perform operations like retrieving, inserting, updating, and deleting data from database tables. It contains Select, Insert, Update and Delete command

Que 2: Data types.

- Data types are used to represent the nature of the data that can be stored in the database table.
- Data types mainly classified into three categories for every database.
    - String Data types
    - Numeric Data types
    - Date and time Data types
- **MySQL String Data Types**

CHAR(Size)	It is used to specify a fixed length string that can contain numbers, letters, and special characters. Its size can be 0 to 255 characters. Default is 1.
VARCHAR(Size)	It is used to specify a variable length string that can contain

	numbers, letters, and special characters. Its size can be from 0 to 65535 characters.
BINARY(Size)	It is equal to CHAR() but stores binary byte strings. Its size parameter specifies the column length in the bytes. Default is 1.
TEXT(Size)	It holds a string that can contain a maximum length of 255 characters.
TINYTEXT	It holds a string with a maximum length of 255 characters.
MEDIUMTEXT	It holds a string with a maximum length of 16,777,215.
LONGTEXT	It holds a string with a maximum length of 4,294,967,295 characters.
ENUM(val1, val2, val3,...)	It is used when a string object having only one value, chosen from a list of possible values. It contains 65535 values in an ENUM list. If you insert a value that is not in the list, a blank value will be inserted.
BLOB(size)	It is used for BLOBs (Binary Large Objects). It can hold up to 65,535 bytes. stores any kind of binary data in random-access chunks, called sbspaces

MySQL Numeric Data Types

BIT(Size)	It is used for a bit-value type. The number of bits per value is specified in size. Its size can be 1 to 64. The default value is 1.
INT(size)	It is used for the integer value. Its signed range varies from -2147483648 to 2147483647 and unsigned range varies from 0 to 4294967295. The size parameter specifies the max display width that is 255.
INTEGER(size)	It is equal to INT(size).
FLOAT(size, d)	It is used to specify a floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal point is specified by **d** parameter.

FLOAT(p)	It is used to specify a floating point number. MySQL used p parameter to determine whether to use FLOAT or DOUBLE. If p is between 0 to24, the data type becomes FLOAT (). If p is from 25 to 53, the data type becomes DOUBLE().
DOUBLE(size, d)	It is a normal size floating point number. Its size parameter specifies the total number of digits. The number of digits after the decimal is specified by d parameter.
DECIMAL(size, d)	It is used to specify a fixed point number. Its size parameter specifies the total number of digits. The number of digits after the decimal parameter is specified by **d** parameter. The maximum value for the size is 65, and the default value is 10. The maximum value for **d** is 30, and the default value is 0.
DEC(size, d)	It is equal to DECIMAL(size, d).
BOOL	It is used to specify Boolean values true and false. Zero is considered as false, and nonzero values are considered as true.

MySQL Date and Time Data Types

DATE	It is used to specify date format YYYY-MM-DD. Its supported range is from '1000-01-01' to '9999-12-31'.
DATETIME(fsp)	It is used to specify date and time combination. Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1000-01-01 00:00:00' to 9999-12-31 23:59:59'.
TIMESTAMP(fsp)	It is used to specify the timestamp. Its value is stored as the number of seconds since the Unix epoch('1970-01-01 00:00:00' UTC). Its format is YYYY-MM-DD hh:mm:ss. Its supported range is from '1970-01-01 00:00:01' UTC to '2038-01-09 03:14:07' UTC.
TIME(fsp)	It is used to specify the time format. Its format is hh:mm:ss. Its supported range is from '-838:59:59' to '838:59:59'
YEAR	It is used to specify a year in four-digit format. Values allowed in four digit format from 1901 to 2155, and 0000.

Que 3: How to create Table

- CREATE TABLE statement is used to create table in a database.
- If you want to create a table, you should name the table and define its column and each column's data type.

  Syntax:
  ```
      create table "tablename"
      ("column1" "data type",
      "column2" "data type",
      "column3" "data type",
      ...
  "columnN" "data type");
  ```
  Example:
  ```
      create table student(name varchar(10),address varchar(50),mobn int(10));
  ```

Que 4: Explain Sql constraints .

- SQL constraints are used to specify rules for the data in a table.
- **Constraints can be column level** or **table level**. Column level constraints apply to a column, and table level constraints apply to the whole table.
- Constraints can be specified when the table is created with the CREATE TABLE statement, or after the table is created with the ALTER TABLE statement.

  Syntax
  ```
      CREATE TABLE table_name (
         column1 datatype constraint,
         column2 datatype constraint,
         column3 datatype constraint,
         ....
  );
  ```

- **The following constraints are commonly used in SQL:**

1. **<u>NOT NULL</u>**

- Ensures that a column cannot have a NULL value

  Example

  create table employee(ename varchar(10) not null,address varchar(10) not null, mnumber int not null);

2. **<u>UNIQUE</u>**

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

  Example

  create table emp2 (enamevarchar(12) not null unique , adress varchar(40) not null);

3. <u>PRIMARY KEY</u>

- The PRIMARY KEY constraint uniquely identifies each record in a table.
- Primary keys must contain UNIQUE values, and cannot contain NULL values.

  Example

  create table emp3(eid int not null,ename varchar(12) not null , adress varchar(40) not null,primary key (eid));

4. <u>FOREIGN KEY</u>

- A FOREIGN KEY is a field (or collection of fields) in one table, that refers to the PRIMARY KEY in another table.
- The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.

  Example

  CREATE TABLE teacher(tid int not null, tname varchar(10), PRIMARY KEY (tid));
  CREATE TABLE student (sid int NOT NULL, sname varchar(12) NOT NULL, tid int,
  PRIMARY KEY (sid),FOREIGN KEY (tid) REFERENCES teacher(tid));

5. <u>CHECK</u>–

- The CHECK constraint is used to limit the value range that can be placed in a column.
- If you define a CHECK constraint on a column it will allow only certain values for this column.
- If you define a CHECK constraint on a table it can limit the values in certain columns based on values in other columns in the row.

  Example

  CREATE TABLE student1 (sid int NOT NULL, sname varchar(12) NOT NULL,age int not null, check (age>=10));

6. **DEFAULT**–

- The DEFAULT constraint is used to set a default value for a column.
- The default value will be added to all new records, if no other value is specified.

  Example

  CREATE TABLE student (sid int not null,sname varchar(12), scity varchar(12) default "kalol");

7. **CREATE INDEX**

- The CREATE INDEX statement is used to create indexes in tables.
- Indexes are used to retrieve data from the database more quickly than otherwise. The users cannot see the indexes, they are just used to speed up searches/queries.

  Example

  create index index_sname on student(sname);

Que 4: How to add Table row ?

- SQL requires the use of the INSERT command to enter data into a table.

  Syntax
- It is possible to write the INSERT INTO statement in two ways:

1. Specify both the column names and the values to be inserted:

INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);

Example

insert into student (sid,sname)values(1,"monali");

sid	**sname**	
1	monali	

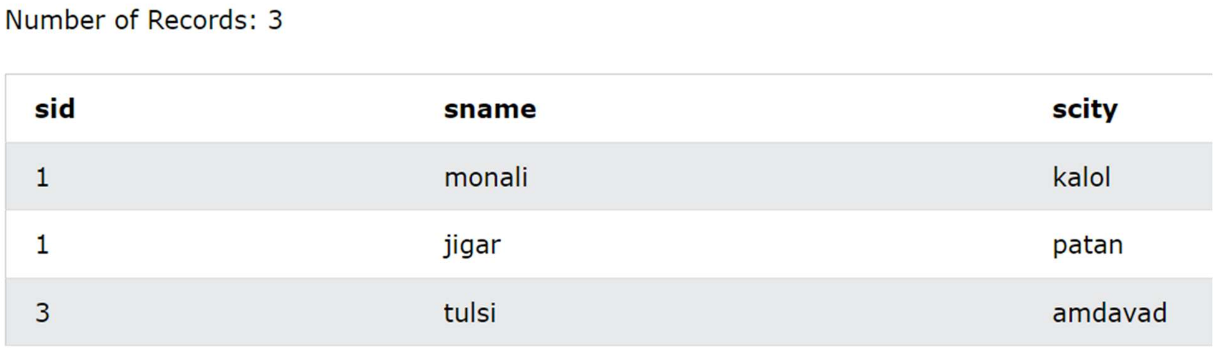
2. If you are adding values for all the columns of the table, you do not need to specify the column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO syntax would be as follows:

INSERT INTO table_name VALUES (value1, value2, value3, ...);

Example

insert into student values (3,"tulsi","amdavad");

Que



Number of Records: 3

sid	sname	scity
1	monali	kalol
1	jigar	patan
3	tulsi	amdavad

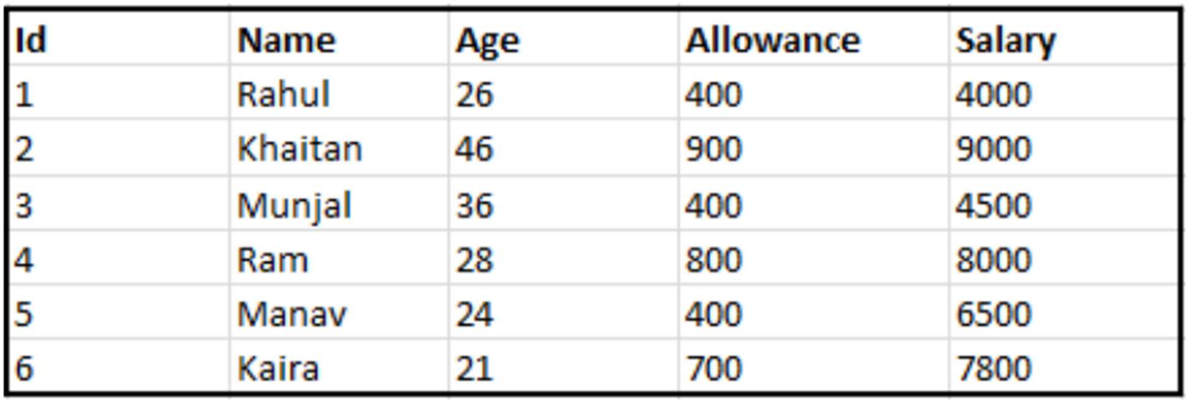
Que 5 :Saving table changes and restoring table data or commit or rollback command in sql

<u>Commit</u>

- If you want to save all the commands which are executed in a transaction, then just after completing the transaction, you have to execute the commit command.
- This command will save all the commands which are executed on a table. All these changes made to the table will be saved to the disk permanently.

Example: Consider the following STAFF table with records:

STAFF

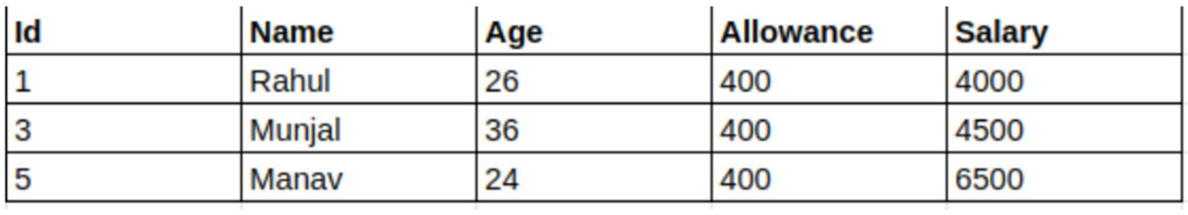


Id	Name	Age	Allowance	Salary
1	Rahul	26	400	4000
2	Khaitan	46	900	9000
3	Munjal	36	400	4500
4	Ram	28	800	8000
5	Manav	24	400	6500
6	Kaira	21	700	7800

sql> SELECT *FROM Staff WHERE Allowance = 400;

sql> COMMIT;

Output:



Id	Name	Age	Allowance	Salary
1	Rahul	26	400	4000
3	Munjal	36	400	4500
5	Manav	24	400	6500

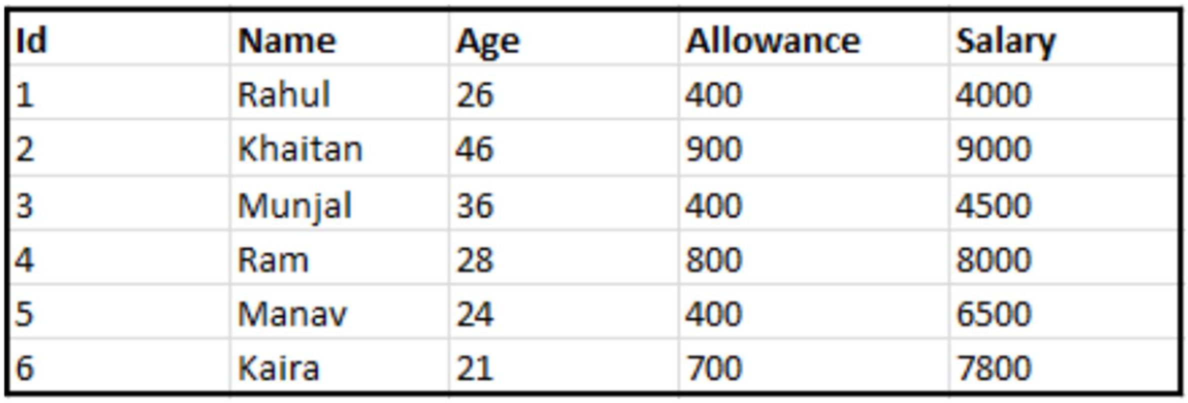
<u>rollback</u>

- The rollback command is used to get back to the previous permanent status of the table, which is saved by the commit command.
- Suppose, we have started editing a table and later thought that the changes that we have recently carried out on a table are not required. Then, in that case, we can roll back our transaction, which simply means to get back to the previous permanent status of the table, which is saved by the commit command.

Example:

sql> SELECT *FROM EMPLOYEES WHERE ALLOWANCE = 400;

sql> ROLLBACK;



Id	Name	Age	Allowance	Salary
1	Rahul	26	400	4000
2	Khaitan	46	900	9000
3	Munjal	36	400	4500
4	Ram	28	800	8000
5	Manav	24	400	6500
6	Kaira	21	700	7800

Difference between COMMIT and ROLLBACK

	COMMIT	**ROLLBACK**
1.	COMMIT permanently saves the	ROLLBACK undo the changes made by the

	COMMIT	**ROLLBACK**
	changes made by the current transaction.	current transaction.
2.	The transaction can not undo changes after COMMIT execution.	Transaction reaches its previous state after ROLLBACK.
3.	When the transaction is successful, COMMIT is applied.	When the transaction is aborted, incorrect execution, system failure ROLLBACK occurs.
4.	COMMIT statement permanently save the state, when all the statements are executed successfully without any error.	In ROLLBACK statement if any operations fail during the completion of a transaction, it cannot permanently save the change and we can undo them using this statement.
5.	**Syntax of COMMIT statement are:** COMMIT;	**Syntax of ROLLBACK statement are:** ROLLBACK;

Que 6: How to select data from table or Listing Table Rows.

- The SELECT Statement in SQL is used to retrieve or fetch data from a table

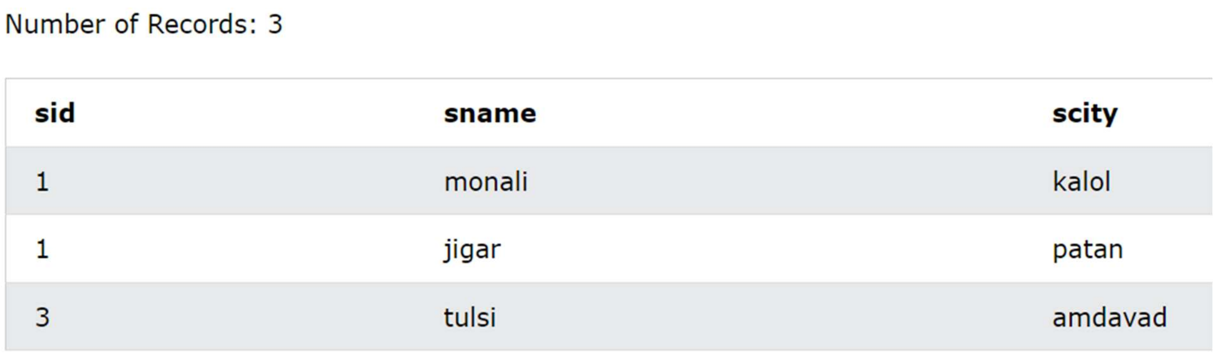
  Syntax:

  Select *from table;

  Example

  Select *from student;

  Output


Number of Records: 3

sid	sname	scity
1	monali	kalol
1	jigar	patan
3	tulsi	amdavad

Que 7:How to update rows in table?

- The UPDATE statement in SQL is used to update the data of an existing table in the database.
- We can update single columns as well as multiple columns using the UPDATE statement as per our requirement.

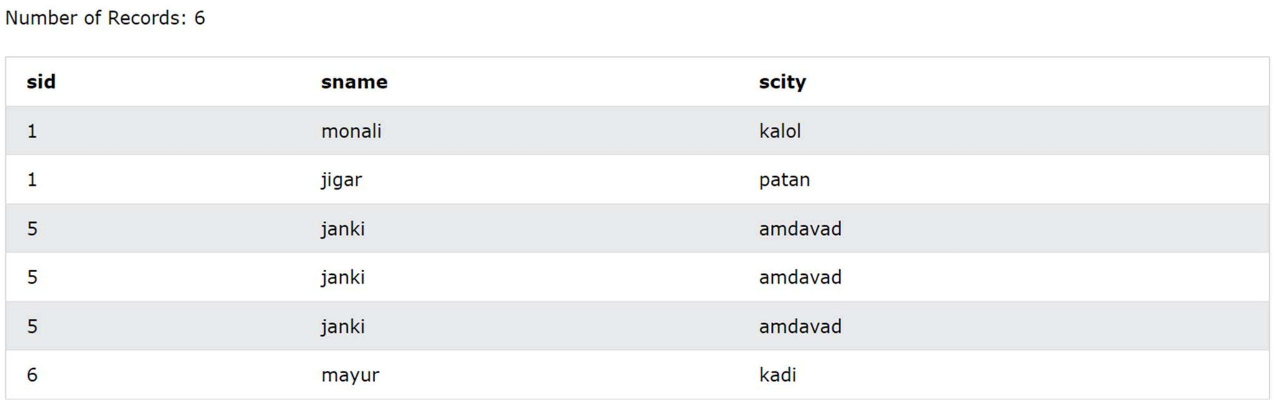
Syntax

UPDATE table_name SET column1 = value1, column2 = value2, ...WHERE condition;

Example

update student set sid=5,sname="janki" where scity="amdavad";

Output


Number of Records: 6

sid	sname	scity
1	monali	kalol
1	jigar	patan
5	janki	amdavad
5	janki	amdavad
5	janki	amdavad
6	mayur	kadi

Que 8:How to delete table row? Or how to delete records from table?

- The DELETE statement is used to delete existing records in a table.

Syntax

DELETE FROM *table_name* WHERE *condition*;

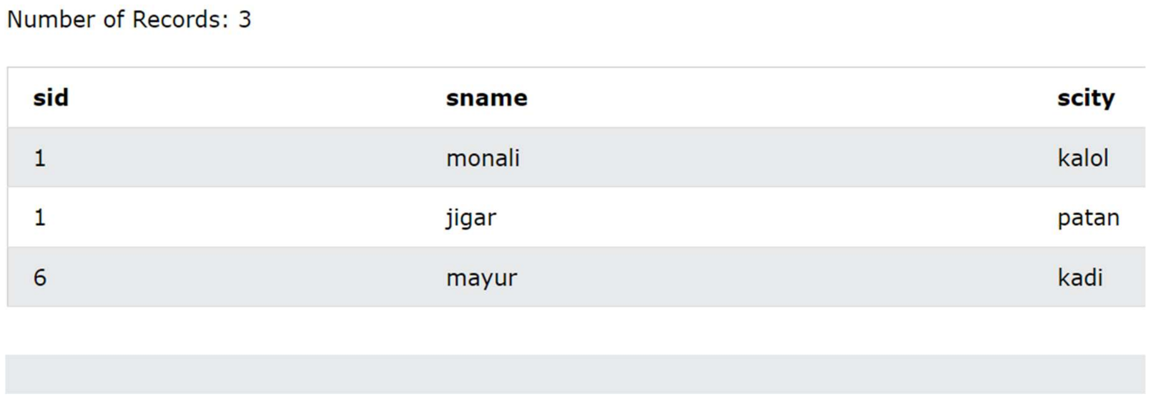
Example

delete from student where scity="amdavad";

Output

You have made changes to the database. Rows affected: 3

SELECT * FROM [student]



Number of Records: 3

sid	sname	scity
1	monali	kalol
1	jigar	patan
6	mayur	kadi

Que 9: What is What is SQL Operator? List out Types of operator .

Explain select query conditional operators or with conditional restriction.

1. **Conditional Operators**
2. **Arithmetic Operators**
3. **Logical Operators**
4. **Special Operators**

- The SQL reserved words and characters are called operators, which are used with a WHERE clause in a SQL query. In SQL, an operator can either be a unary or binary operator. The unary operator uses only one operand for performing the unary operation, whereas the binary operator uses two operands for performing the binary operation.You can select partial table contents by placing restrictions on the rows to be included in the output.
- This is done by using the WHERE clause to add conditional restrictions to the SELECT statement.
- Numerous conditional restrictions can be placed on the selected table contents. For example, the comparison
- operators shown in following Table can be used to restrict output.

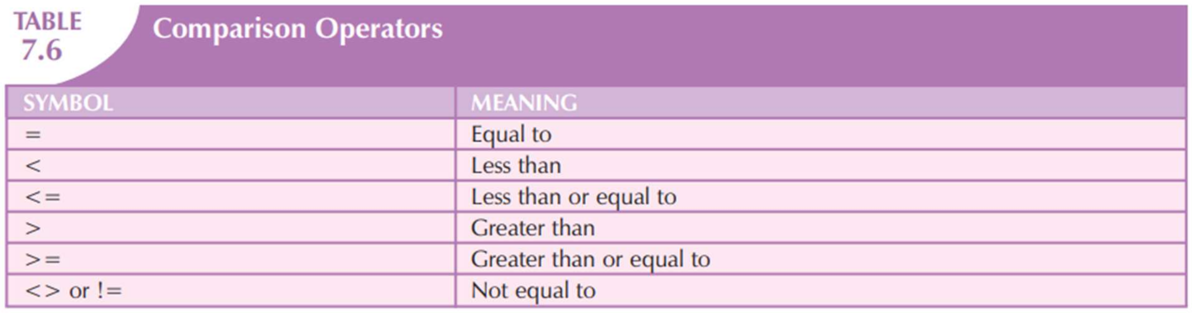


TABLE 7.6 Comparison Operators

SYMBOL	MEANING
=	Equal to
<	Less than
<=	Less than or equal to
>	Greater than
>=	Greater than or equal to
<> or !=	Not equal to

- The following syntax enables you to specify which rows to select:

  Syntax

  SELECT column1, column2, ...

  FROM table_name

  WHERE condition;

  Example:

SID	SNAME	SCITY
1	TULSI	AMDAVAD
2	JANVI	AMDAVAD
3	HIRAL	KALOL
4	MONALI	MEHSANA
5	SUCHITA	KADI

Select sname from student where SCITY="AMDAVAD";

Output:

SID	SNAME	SCITY
1	TULSI	AMDAVAD
2	JANVI	AMDAVAD

Que 10: Explain select query with Arithmetic Operators.

- We can use these operators with the SELECT statement in SQL.
- We can also use the WHERE clause in the SELECT statement for performing operations on particular rows.
- These types of operators are used between two numerical operands for performing addition, subtraction, multiplication, and division operations.
- The arithmetic operators shown in following Table

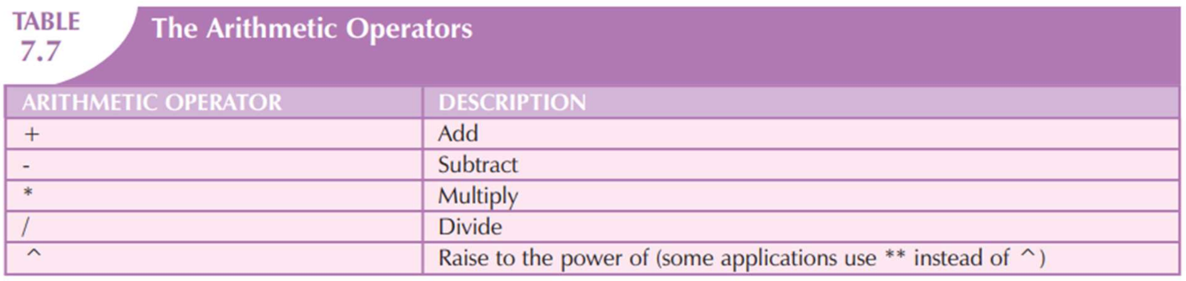


TABLE 7.7 The Arithmetic Operators

ARITHMETIC OPERATOR	DESCRIPTION
+	Add
-	Subtract
*	Multiply
/	Divide
^	Raise to the power of (some applications use ** instead of ^)

1. **<u>SQL Addition Operator (+)</u>**

- The SQL Addition Operator performs the addition on the numerical columns in the table.
- If you want to add the values of two numerical columns in the table, then you have to specify both columns as the first and second operand.
- You can also add the new integer value in the value of the integer column.

Syntax of SQL Addition Operator:

SELECT Column_Name_1 Addition_Operator Column_Name2 FROM Table_Name;

Addition Operator with WHERE Clause

The addition operator can also be used with the WHERE clause in the SQL SELECT query.

The syntax for using the WHERE clause with the addition operator is given below:

SELECT Column_Name_1 Addition_Operator Column_Name2 FROM Table_Name WHERE Condition

Example

Employee_Id	Emp_Name	Emp_City	Emp_Salary	Emp_bonus
101	Anuj	Ghaziabad	25000	2000
102	Tushar	Lucknow	29000	1000
103	Vivek	Kolkata	35000	2500
104	Shivam	Goa	22000	3000

SELECT Emp_Salary + Emp_Bonus AS Emp_Total_Salary FROM Employee;

Output:

Emp_Total_Salary
27000
30000
37500
25000

The following query performs the addition operation on the above Employee table with the WHERE clause:

SELECT Emp_Salary + Emp_Bonus AS Emp_Total_Salary FROM Employee WHERE Emp_Salary> 25000;

Emp_Total_Salary
30000
37500

2. **SQL Subtraction Operator (-)**

- The SQL Subtraction Operator performs the subtraction on the numerical columns in the table.
- If we want to subtract the values of one numerical column from the values of another numerical column, then we have to specify both columns as the first and second operand.
- We can also subtract the integer value from the values of the integer column.

Syntax of SQL Subtraction Operator:

SELECT Column_Name_1 Subtraction_Operator Column_Name2 FROM Table_Name;

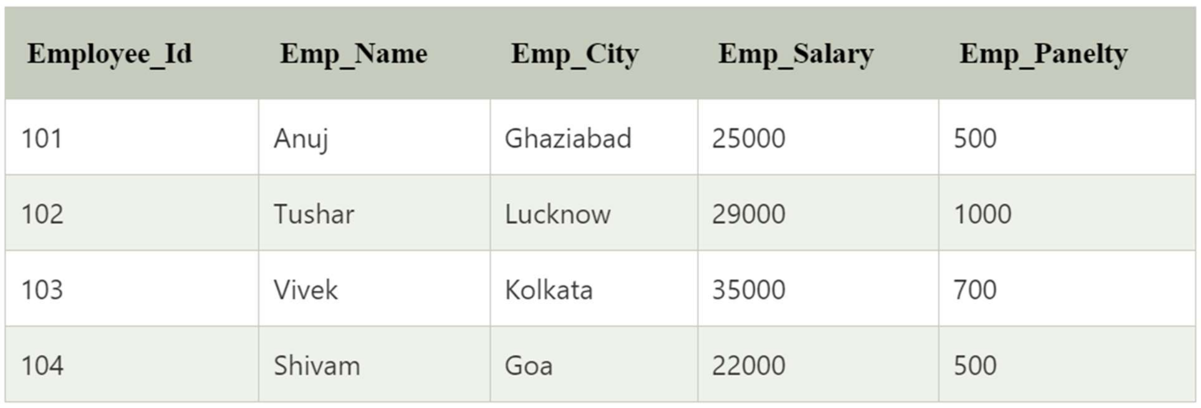
Subtraction Operator with WHERE Clause

- The subtraction operator can also be used with the WHERE clause in the SELECT query.

The syntax for using the WHERE clause with the subtraction operator is given below:

SELECT Column_Name_1 Subtraction_Operator Column_Name2 FROM Table_Name WHERE condition;

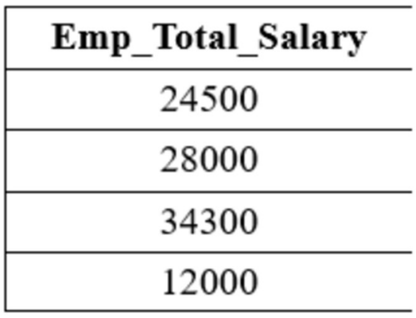
Example



Employee_Id	Emp_Name	Emp_City	Emp_Salary	Emp_Panelty
101	Anuj	Ghaziabad	25000	500
102	Tushar	Lucknow	29000	1000
103	Vivek	Kolkata	35000	700
104	Shivam	Goa	22000	500

SELECT Emp_Salary - Emp_Panelty AS Emp_Total_Salary FROM Employee;

Output:

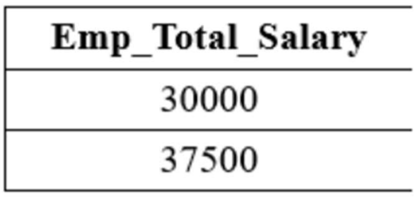


Emp_Total_Salary
24500
28000
34300
12000

The following query performs the subtraction operation on the above Employee table with the WHERE clause:

SELECT Emp_Panelty - Emp_Salary AS Emp_Total_Salary FROM Employee WHERE Employee_ID =104;

Output:



Emp_Total_Salary
30000
37500

3. <u>**SQL Multiplication Operator (*)**</u>

- The SQL Multiplication Operator performs the multiplication on the numerical columns in the table.
- If you want to multiply the values of two numerical columns, then you have to specify both columns as the first and second operand.
- You can also multiply the integer value with the values of an integer column.

Syntax of SQL Multiplication Operator:

SELECT Column_Name_1 Multiplication_Operator Column_Name2 FROM Table_Name;

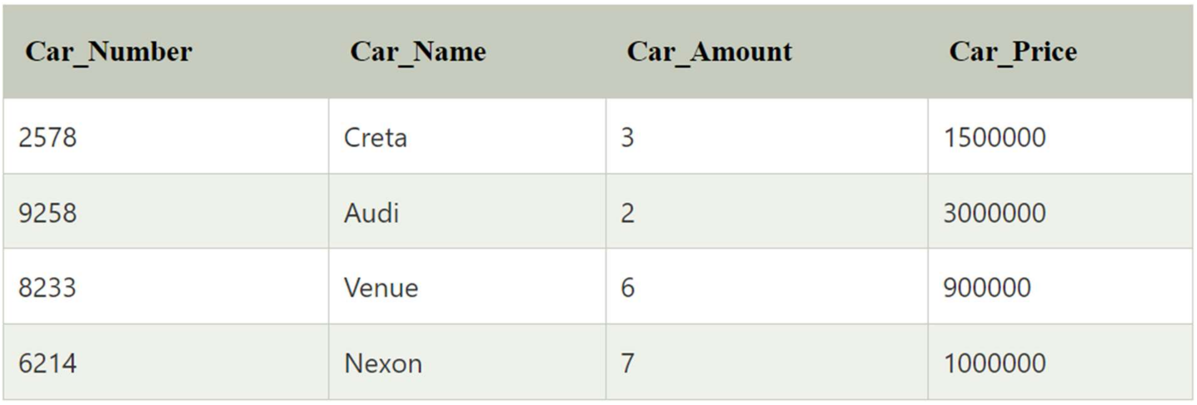
Multiplication Operator with WHERE Clause

- The multiplication operator (*) can also be used with the WHERE clause in the SELECT query.

The syntax for using the WHERE clause with the multiplication operator is given below:

SELECT Column_Name_1 Multilplication_Operator Column_Name2 FROM Table_Name WHERE Condition;

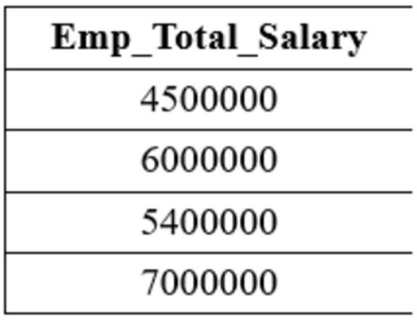
Example



Car_Number	Car_Name	Car_Amount	Car_Price
2578	Creta	3	1500000
9258	Audi	2	3000000
8233	Venue	6	900000
6214	Nexon	7	1000000

SELECT Car_Amount * Car_Price AS Car_Total_Price FROM Cars;

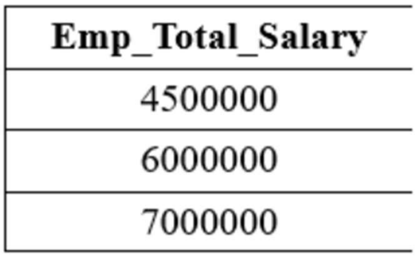
Output:



Emp_Total_Salary
4500000
6000000
5400000
7000000

SELECT Car_Amount * Car_Price AS Car_Total_Price FROM Cars WHERE Car_Price>= 1000000;

Output:



Emp_Total_Salary
4500000
6000000
7000000

4. **<u>SQL Division Operator (/)</u>**

- The SQL Division operator divides the numerical values of one column by the numerical values of another column.

Syntax of SQL Division Operator:

SELECT Column_Name_1 Division_Operator Column_Name2 FROM Table_Name;

Division Operator with WHERE Clause

The SQL division operator can also be used with the WHERE clause in the SELECT query.

The syntax for using the WHERE clause with the division operator is given below:

SELECT Column_Name_1 Division_Operator Column_Name2 FROM Table_Name WHERE Condition;

Example

Car_Number	Car_Name	Car_Amount	Car_Price
2578	Creta	3	1500000
9258	Audi	2	3000000
8233	Venue	6	900000
6214	Nexon	10	1000000

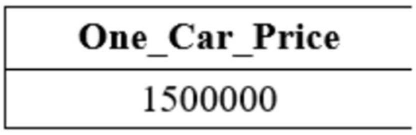
SELECT Car_Price / Car_Amount AS One_Car_Price FROM Cars;

Output:

One_Car_Price
500000
1500000
150000
100000

SELECT Car_Price / Car_Amount AS One_Car_Price FROM Cars WHERE Car_Number = 9258;

Output:



One_Car_Price
1500000

Que 10: Explain select query with Logical Operators.

The Logical Operators in SQL perform the Boolean operations, which give two results True and False. These operators provide True value if both operands match the logical condition.

Following are the various logical operators which are performed on the data stored in the SQL database tables:

1. SQL ALL operator
2. SQL AND operator
3. SQL OR operator
4. SQL NOT operator
5. SQL ANY operator

The ALL operator in SQL compares the specified value to all the values of a column from the sub-query in the SQL database.

This operator is always used with the following statement:

SELECT,

HAVING, and

WHERE.

1. **<u>Syntax of ALL operator:</u>**

SELECT column_Name1, ..., column_NameN FROM table_Name WHERE column Compari
son_operator

Example

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Gurgaon
202	Ankit	45000	Delhi
203	Bheem	30000	Jaipur
204	Ram	29000	Mumbai
205	Sumit	40000	Kolkata

SELECT Emp_Id, Emp_Name FROM Employee_details WHERE Emp_Salary **>** ALL (SELECT Emp_Salary FROM Employee_details WHERE Emp_City = Jaipur)

2. **SQL AND Operator**

The **AND operator** in SQL would show the record from the database table if all the conditions separated by the AND operator evaluated to True. It is also known as the conjunctive operator and is used with the WHERE clause.

Syntax of AND operator:

SELECT column1, ..., columnN FROM table_Name WHERE condition1 AND condition2 AND condition3AND .... AND conditionN;

Example

Emp Id	Emp Name	Emp Salary	Emp City
201	Abhay	25000	Delhi
202	Ankit	45000	Chandigarh
203	Bheem	30000	Delhi
204	Ram	25000	Delhi
205	Sumit	40000	Kolkata

Suppose, we want to access all the records of those employees from the Employee_details table whose salary is 25000 and the city is Delhi. For this, we have to write the following query in SQL:

SELECT * FROM Employee_details WHERE Emp_Salary = 25000 AND Emp_City = 'Delhi';

Here,SQL AND operator with WHERE clause shows the record of employees whose salary is 25000 and the city is Delhi.

3. SQL OR Operator

The OR operator in SQL shows the record from the table if any of the conditions separated by the OR operator evaluates to True. It is also known as the conjunctive operator and is used with the WHERE clause.

Syntax of OR operator:

SELECT column1, ..., columnN FROM table_Name WHERE condition1 OR condition2 OR condition3 OR....... OR conditionN;

Example

SELECT * FROM Employee_details WHERE Emp_Salary = 25000 OR Emp_City = 'Delhi';

4. SQL NOT Operator

The NOT operator in SQL shows the record from the table if the condition evaluates to false. It is always used with the WHERE clause.

Syntax of NOT operator:

SELECT column1, column2 ...., columnN FROM table_Name WHERE NOT condition;

Example

Suppose, we want to show all the information of those employees from the Employee_details table whose Cityis not Delhi and Chandigarh. For this, we have to write the following query in SQL:

SELECT * FROM Employee_details WHERE NOT Emp_City = 'Delhi' AND NOT Emp_City = 'Chandigarh';

5. SQL ANY Operator

The **ANY operator** in SQL shows the records when any of the values returned by the sub-query meet the condition.

The ANY logical operator must match at least one record in the inner query and must be preceded by any SQL comparison operator.

Syntax of ANY operator:

SELECT column1, column2 ...., columnN FROM table_Name WHERE column_name comp arison_operator ANY (SELECT column_name FROM table_name WHERE condition(s)) ;

Example

The following SQL statement lists the ProductName if it finds ANY records in the OrderDetails table has Quantity equal to 10 (this will return TRUE because the Quantity column has some values of 10):

SELECT ProductName FROM Products WHERE ProductID = ANY (SELECT ProductID FROM OrderDetails WHERE Quantity = 10);

Que 11:Explain Special Operators .

The different special operators in SQL are as follows

1. Between Operator
2. Is Null Operator
3. Like Operator
4. In Operator
5. Exists Operator

1. Between Operator

The **BETWEEN operator** in SQL shows the record within the range mentioned in the SQL query. This operator operates on the numbers, characters, and date/time operands.

If there is no value in the given range, then this operator shows NULL value.

Syntax of BETWEEN operator:

SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WHERE co lumn_name BETWEEN value1 and value2;

Example

SELECT * FROM Employee_details WHERE Emp_Salary BETWEEN 30000 AND 45000;

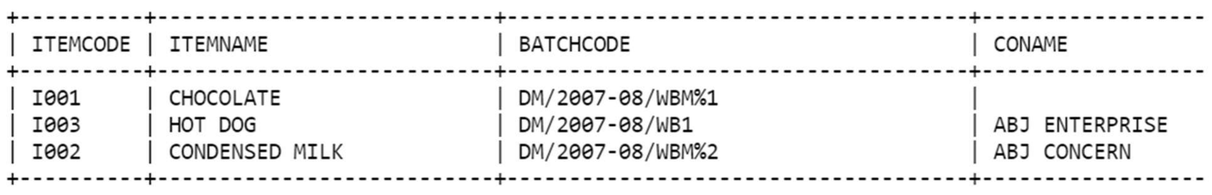
2. **<u>Is Null Operator</u>**

IS NULL operator is used to test for a NULL value.

Syntax

SELECT [column_name1,column_name2]FROM [table_name]WHERE [column_name] IS [NOT] NULL;

Example



```
+----------+-------------------------+----------------------------------+-----------------
| ITEMCODE | ITEMNAME                | BATCHCODE                        | CONAME
+----------+-------------------------+----------------------------------+-----------------
| I001     | CHOCOLATE               | DM/2007-08/WBM%1                 |
| I003     | HOT DOG                 | DM/2007-08/WB1                   | ABJ ENTERPRISE
| I002     | CONDENSED MILK          | DM/2007-08/WBM%2                 | ABJ CONCERN
+----------+-------------------------+----------------------------------+-----------------
```

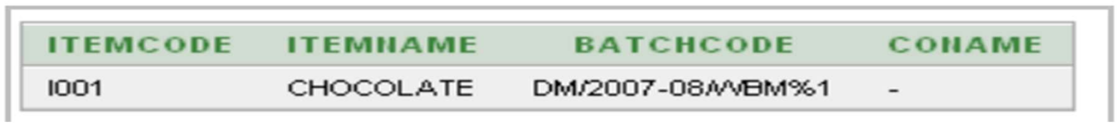
To get data of all columns from the 'listofitem' table with following condition -

1. coname column contain NULL value,

the following sql statement can be used :

SELECT * FROM listofitem WHERE coname IS NULL;

Output:



ITEMCODE	ITEMNAME	BATCHCODE	CONAME
I001	CHOCOLATE	DM/2007-08/WBM%1	-

3. **<u>Like Operator</u>**

Syntax of Like operator

SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WHERE colum n_name LIKE pattern;

Example

If we want to show all the information of those employees from the Employee_details whose name starts with ''s''. For this, we have to write the following query in SQL:

SELECT * FROM Employee_details WHERE Emp_Name LIKE 's%' ;

4. **<u>In Operator</u>**

The IN operator in SQL allows database users to specify two or more values in a WHERE clause. This logical operator minimizes the requirement of multiple OR conditions.

This operator makes the query easier to learn and understand. This operator returns those rows whose values match with any value of the given list.

Syntax of IN operator:

SELECT column_Name1, column_Name2 ...., column_NameN FROM table_Name WHERE column_name IN (list_of_values);

Example

Suppose, we want to show all the information of those employees from the **Employee_details** table whose **Employee Id** is 202, 204, and 205. For this, we have to write the following query in SQL:

SELECT * FROM Employee_details WHERE Emp_Id IN (202, 204, 205);

5. **<u>Exists Operator</u>**

The EXISTS operator is used to test for the existence of any record in a subquery.
The EXISTS operator returns TRUE if the subquery returns one or more records.

Syntax

SELECT column_name(s)FROM table_nameWHERE EXISTS(SELECT column_name FROM table_name WHERE condition);

Example

```
SELECT fname, lname FROM Customers WHERE EXISTS (SELECT * FROM Orders
WHERE Customers.customer_id = Orders.c_id);
```

Que 12:Explain alter command .

- The ALTER TABLE statement in Structured Query Language allows you to add, modify, and delete columns of an existing table.
- This statement also allows database users to add and remove various SQL constraints on the existing tables.Any user can also change the name of the table using this statement.

<u>ALTER TABLE ADD Column statement in SQL</u>

- In many situations, you may require to add the columns in the existing table. Instead of creating a whole table or database again you can easily add single and multiple columns using the ADD keyword.

Syntax

ALTER TABLE table_name ADD column_name column-definition;

Example

alter table student add scity varchar(12);

ALTER TABLE - DROP COLUMN

To delete a column in a table, use the following syntax (notice that some database systems don't allow deleting a column):

Syntax

ALTER TABLE table_nameDROP COLUMN column_name;

Example

ALTER TABLE student DROP COLUMN scity;

ALTER TABLE - RENAME COLUMN

To rename a column in a table, use the following syntax:

Syntax

ALTER TABLE table_name rename COLUMN old to new;

Example

alter table student rename column scity to city;

ALTER TABLE - ALTER/MODIFY DATATYPE

To change the data type of a column in a table, use the following syntax:

ALTER TABLE table_nameMODIFY COLUMN column_name datatype;

Example

ALTER TABLE student ALTER COLUMN city text;

ALTER TABLE Column's Data characteristics in SQL

If the column to be changed already contains data, you can make changes in the column's characteristics if thosechanges do not alter the data type.

For example, if you want to increase the width of the P_PRICE column to 9 digits, use the command:

Example

ALTER TABLE PRODUCTMODIFY (P_PRICE DECIMAL(9,2));

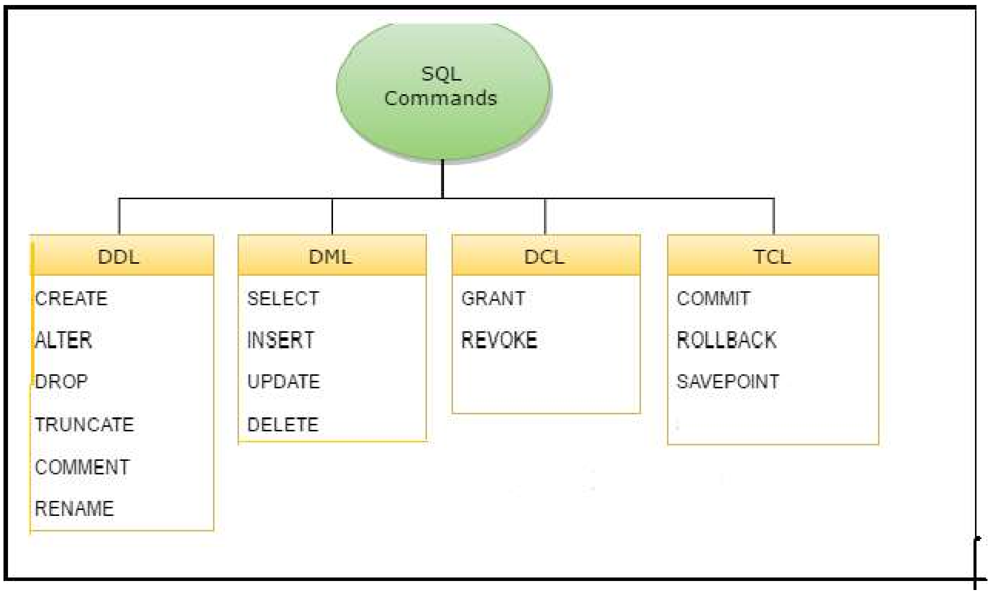
Que 12: How to delete Table from Database.

Using drop keyword we can delete the table from database.

Example:

Drop table student;

Que 13: Explain SQL Command



DDL (Data Definition Language) :

CREATE: This command is used to create the database or its objects (like table, index, function, views, store procedure, and triggers).

- If you want to create a table, you should name the table and define its column and each column's data type.

  Syntax:

  ```
  create table "tablename"
  ("column1" "data type",
  "column2" "data type",
  "column3" "data type",
  ...
  "columnN" "data type");
  ```

  Example:

  ```
  create table student(name varchar(10),address varchar(50),mobn int(10));
  ```

DROP: This command is used to delete objects from the database.

Syntax

DROP object object_name ;

Example

DROP TABLE table_name;

ALTER: This is used to alter the structure of the database. (Syntax and Examples and already given in above question no.12)

TRUNCATE: This is used to remove all records from a table, including all spaces allocated for the records are removed. The major difference between TRUNCATE and DROP is that truncate is used to delete the data inside the table not the whole table.

Syntax

TRUNCATE TABLE table_name;

Example:

To truncate the Student_details table from the student_data database.

Query:

TRUNCATE TABLE Student_details;

COMMENT: This is used to add comments to the data dictionary.

- Comments can be written in the following three formats:

<u>Single-line comments</u>

Comments starting and ending in a single line are considered single-line comments. A line starting with '–' is a comment and will not be executed.

Example

```
SELECT * FROM customers;
-- This is a comment that explains
```

the purpose of the query.

Multi-line comments

Comments starting in one line and ending in different lines are considered as multi-line comments.

A line starting with '/*' is considered as starting point of the comment and is terminated when '*/' is encountered.

Example

```
/*
This is a multi-line comment that explains
the purpose of the query below.
The query selects all the orders from the orders
table that were placed in the year 2022.
*/
SELECT * FROM orders WHERE YEAR(order_date) = 2022;
```

In-line comments

In-line comments are an extension of multi-line comments, comments can be stated in between the statements and are enclosed in between '/*' and '*/'.

Example

```
SELECT * FROM /* Customers; */
```

RENAME: Sometimes we may want to rename our table to give it a more relevant name. For this purpose, we can use ALTER TABLE to rename the name of the table.

DQL (Data Query Language)

- **DQL** statements are used for performing queries on the data within schema objects.
- The purpose of the DQL Command is to get some schema relation based on the query passed to it.
- We can define DQL as follows it is a component of SQL statement that allows getting data from the database and imposing order upon it. It includes the SELECT statement.

  List of DQL:

  SELECT: It is used to retrieve data from the database.

DML(Data Manipulation Language)

- The SQL commands that deal with the manipulation of data present in the database belong to DML or Data Manipulation Language and this includes most of the SQL statements.

- It is the component of the SQL statement that controls access to data and to the database. Basically, DCL statements are grouped with DML statements.

List of DML commands:

1) Insert Command

• The INSERT INTO statement is used to insert new records in a table.

It is possible to write the INSERT INTO statement in two ways:

1) Specify both the column names and the values to be inserted:

Syntax:

INSERT INTO table_name (column1, column2, column3, ...) VALUES (value1, value2, value3, ...);

Example: INSERT INTO Person (id,name,age) VALUES (1,'mira',30);

2) If you are adding values for all the columns of the table, you do not need to specifythe column names in the SQL query. However, make sure the order of the values is in the same order as the columns in the table. Here, the INSERT INTO

Syntax:

INSERT INTO table_name VALUES (value1, value2, value3, ...);

Example:

INSERT INTO Person VALUES (1,'dixita',30);

2) Update Command

• The UPDATE statement is used to modify the existing records in a table.

Syntax:

UPDATE table_name SET column1 = value1, column2 = value2 WHERE condition;

Example:

UPDATE Customers SET ContactName = 'Alfred' WHERE CustomerID = 1;

3) Delete Command

• The DELETE statement is used to delete existing records in a table.

Syntax:

DELETE FROM table_name WHERE condition;

Example:

DELETE FROM Customers WHERE CustomerID = 1;

4) Select Command

• The SELECT statement is used to select data from a database.

Syntax:

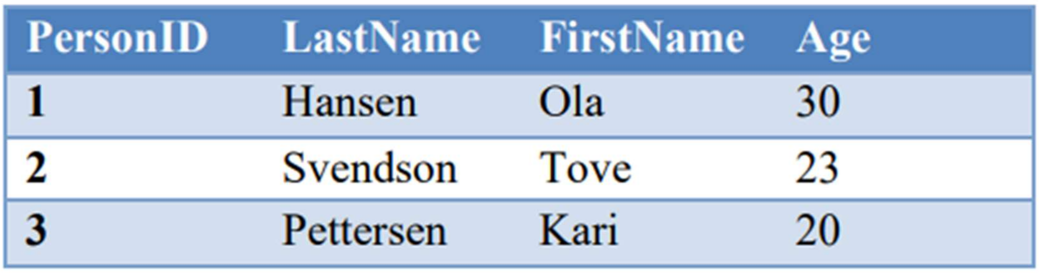
SELECT column1, column2, FROM table_name;

Here, column1, column2, are the field names of the table you want to select data from. If you want to select all the fields available in the table, use the following syntax:

SELECT * FROM table_name;

Example:

SELECT * FROM Person;



PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

1. SELECT with DISTINCT Clause

• The SELECT DISTINCT statement is used to return only distinct (different) values.

Syntax:

SELECT DISTINCT column1, column2,FROM table_name;

Example:

SELECT DISTINCT Country FROM Customers;

2. SELECT with WHERE Clause

• The WHERE clause is used to filter records.

• The WHERE clause is used to extract only those records that fulfill a specified condition.

Syntax:

SELECT column1, column2, FROM table_name WHERE condition;

Example:

SELECT * FROM Customers WHERE Country='Mexico';

3. SELECT with ORDER BY Keyword

• The ORDER BY keyword is used to sort the result-set in ascending or descending order.

• The ORDER BY keyword sorts the records in ascending order by default. To sort the records in descending order, use the DESC keyword.

Syntax:

SELECT column1, column2, FROM table_name ORDER BY column1, column2, ASC|DESC;

Example:

SELECT * FROM Customers ORDER BY Country DESC;

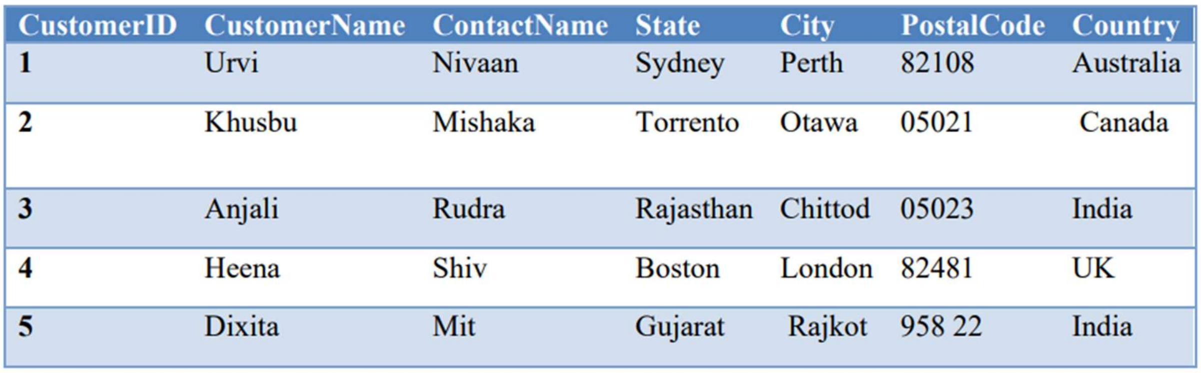
4. SELECT with Group by Clause

• The GROUP BY statement groups rows that have the same values into summary rows like "find the number of customers in each country".

• The GROUP BY statement is often used with aggregate functions (COUNT (), MAX (), MIN (), SUM (), AVG ()) to group the result-set by one or more columns.

Syntax:

SELECT column_name(s) FROM table_name WHERE condition GROUP BY column_name(s) ORDER BY column_name(s);



CustomerID	CustomerName	ContactName	State	City	PostalCode	Country
1	Urvi	Nivaan	Sydney	Perth	82108	Australia
2	Khusbu	Mishaka	Torrento	Otawa	05021	Canada
3	Anjali	Rudra	Rajasthan	Chittod	05023	India
4	Heena	Shiv	Boston	London	82481	UK
5	Dixita	Mit	Gujarat	Rajkot	958 22	India

Example:

SELECT COUNT (CustomerID), Country FROM Customers GROUP BY Country;

Output:

CustomerID	Country
1	Australia
1	Canada
2	India
1	UK

5. SELECT with Having Clause

• The HAVING clause was added to SQL because the WHERE keyword cannot be used with aggregate functions**.**

Syntax:

SELECT expression1, expression2,expression_n,aggregate_function (aggregate_expression) FROM tables WHERE conditions GROUP BY expression1, expression2, ... expression_n HAVING having_condition; ORDER BY column_name(s);

• **expression1, expression2, expression_n:** It specifies the expressions that are not encapsulated within aggregate function. These expressions must be included in GROUP BY clause.

• **aggregate_function:** It specifies the aggregate functions i.e. SUM, COUNT, MIN, MAX or AVG functions.

• **aggregate_expression:** It specifies the column or expression on that the aggregate function is based on.

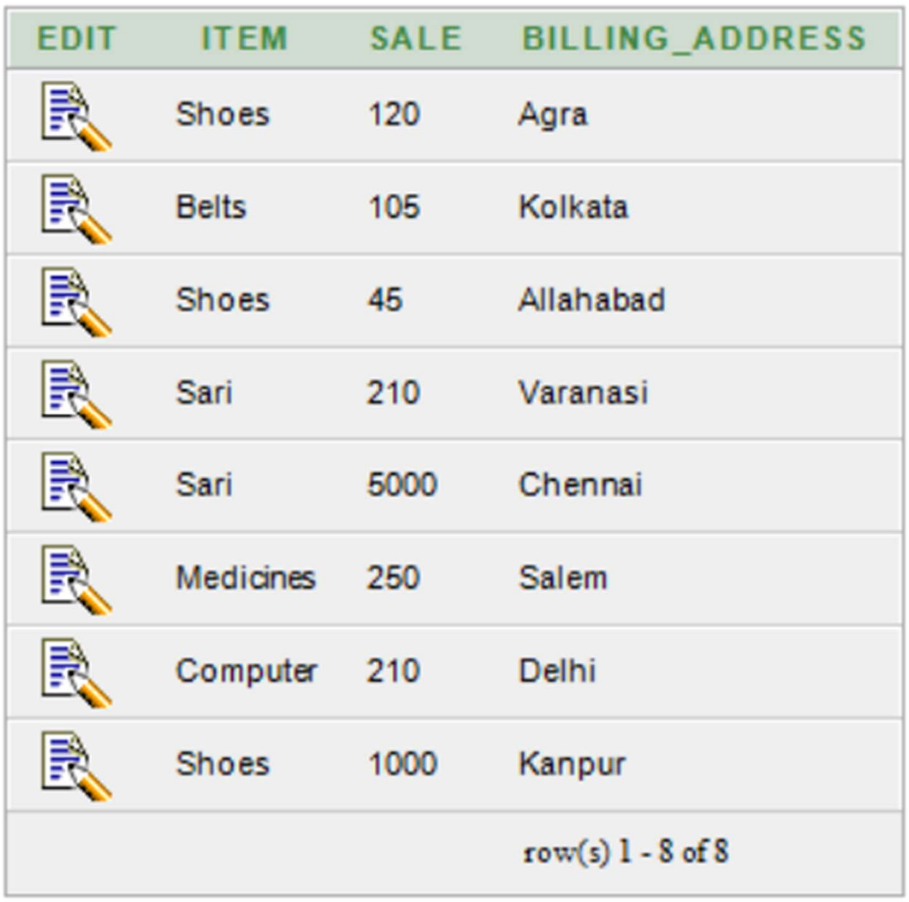
• **tables:** It specifies the table from where you want to retrieve records.

• **conditions:** It specifies the conditions that must be fulfilled for the record to be selected.

• **having_conditions:** It specifies the conditions that are applied only to the aggregated results to restrict the groups of returned rows.

Example

Table Name: sales_department



EDIT	ITEM	SALE	BILLING_ADDRESS
✎	Shoes	120	Agra
✎	Belts	105	Kolkata
✎	Shoes	45	Allahabad
✎	Sari	210	Varanasi
✎	Sari	5000	Chennai
✎	Medicines	250	Salem
✎	Computer	210	Delhi
✎	Shoes	1000	Kanpur
			row(s) 1 - 8 of 8

Example:

SELECT item, SUM(sale) FROM salesdepartment GROUP BY item HAVING SUM(sale) < 1000;

Output:

DCL (Data Control Language)

- DCL includes commands such as GRANT and REVOKE which mainly deal with the rights, permissions, and other controls of the database system.
  List of DCL commands:
- **GRANT:** This command gives users access privileges to the database.
  Syntax:
  GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

  REVOKE: This command withdraws the user's access privileges given by using the GRANT command.
  Syntax:

REVOKE SELECT, UPDATE ON MY_TABLE FROM USER1, USER2;

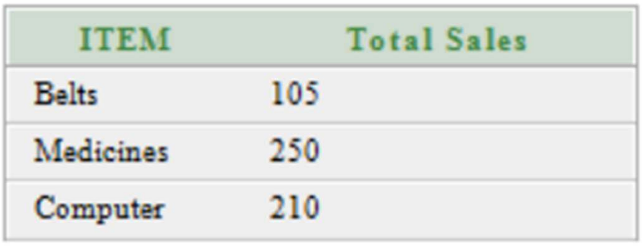
TCL (Transaction Control Language)

- Transactions group a set of tasks into a single execution unit. Each transaction begins with a specific task and ends when all the tasks in the group are successfully completed. If any of the tasks fail, the transaction fails. Therefore, a transaction has only two results: success or failure. You can explore more about transactions here. Hence, the following TCL commands are used to control the execution of a transaction:
  BEGIN: Opens a Transaction.
  COMMIT: Commits a Transaction.
  Syntax:
  COMMIT;

  ROLLBACK: Rollbacks a transaction in case of any error occurs.
  Syntax:
  ROLLBACK;

  SAVEPOINT: Sets a save point within a transaction.

  SAVEPOINT SAVEPOINT_NAME;



ITEM	Total Sales
Belts	105
Medicines	250
Computer	210