DDL(Data Definition Language)

CREATE: Create command is used to create the database objects such as tables, indexes, views, procedures etc..

To create a database,

Syntax:

```
CREATE DATABASE database_name;
```

To access a particular database

Syntax:

```
USE database_name;
```

To check the databases that are present in MySQL

```
SHOW DATABASES;
```

Creation of table

```
Syntax: CREATE TABLE table_name
        (
          column_name_1 DATATYPE [CONSTRAINT] NULL/NOT
NULL,
          column_name_2 DATATYPE [CONSTRAINT] NULL/NOT
NULL,
          :
          column_name_n DATATYPE [CONSTRAINT] NULL/NOT
NULL
        );

CREATE TABLE ACCOUNTS
(
 ACC_NO BIGINT PRIMARY KEY,
 NAME VARCHAR(20) NOT NULL,
 PHONE BIGINT UNIQUE NOT NULL,
 BALANCE DECIMAL(12,2) DEFAULT 0.00 NOT NULL
```

```
);
```

To check the tables that are available in the database

```
SHOW TABLES;
```

To view the table structure

```
DESC table_name;
```

BRANCH

```
CREATE TABLE BRANCH
(
BID INT PRIMARY KEY AUTO_INCREMENT,
BNAME VARCHAR(20) NOT NULL,
PINCODE INT UNIQUE
);
```

LOCATION

```
CREATE TABLE LOCATION
(
PINCODE INT PRIMARY KEY,
AREA VARCHAR(20) NOT NULL,
CITY VARCHAR(20) NOT NULL
);
```

ALTER:

This command is used to modify the structure of the table.

1. To add a column

```
ALTER TABLE table_name
```

```
ADD column_name DATATYPE CONSTRAINT;

-- To add email column in accounts table

ALTER TABLE ACCOUNTS
ADD EMAIL VARCHAR(20) UNIQUE;

-- To add last name just after name column in accounts

ALTER TABLE ACCOUNTS
ADD L_NAME VARCHAR(10) AFTER NAME;

-- To add the IFSC Code in the branch table just after
the BID.

ALTER TABLE BRANCH
ADD IFSC_Code VARCHAR(15) AFTER BID;

-- To add the State column in the Location table.
```

```
ALTER TABLE LOCATION
ADD STATE VARCHAR(20);

2. To remove a column
_____________________________

ALTER TABLE table_name
DROP COLUMN column_name;

-- To remove the branch name from branch table

ALTER TABLE BRANCH
DROP COLUMN BNAME;

> Remove the Mail ID column from accounts table

ALTER TABLE ACCOUNTS
DROP COLUMN EMAIL;
```

3. To modify the datatypes

```
ALTER TABLE table_name
MODIFY column_name new_datatype CONSTRAINT;

> To modify the Last_name datatype from varchar to char.
ALTER TABLE ACCOUNTS
MODIFY L_NAME CHAR(10);

-- To modify the PINCODE from int to bigint
ALTER TABLE BRANCH
MODIFY PINCODE BIGINT;
```

4. To convert the column to NULL/NOT NULL

```
ALTER TABLE table_name
MODIFY column_name existing_DATATYPE NULL/NOT NULL;

-- To convert the l_name from null to not null
```

```
ALTER TABLE ACCOUNTS
MODIFY NAME VARCHAR(20) NOT NULL;

> To convert the CITY column to NULL
ALTER TABLE LOCATION
MODIFY CITY VARCHAR(20) NULL;

5. To rename a table
_____________________________

ALTER TABLE table_name
RENAME new_table_name;

> Give another name to Accounts
ALTER TABLE ACCOUNTS
RENAME BANK_ACC;

> Give a new name to branch table
ALTER TABLE BRANCH
RENAME BANK_BRANCHES;
```

6. To rename a column

```
ALTER TABLE table_name
CHANGE old_column_name new_column_name
existing_datatype;
```

> To rename the name to f_name in bank_acc table.

```
ALTER TABLE BANK_ACC
CHANGE NAME FNAME VARCHAR(20);
```

> To rename any column in Branch table.

```
ALTER TABLE BANK_BRANCHES
CHANGE IFSC_CODE IFSC VARCHAR(15);
```

7. To add a constraint for a particular column

```
ALTER TABLE table_name
ADD CONSTRAINT PRIMARY KEY(column_name);
```

```
ADD CONSTRAINT UNIQUE(column_name);
ADD CONSTRAINT CHECK(condition);
ADD CONSTRAINT FOEREIGN KEY(column_name) REFERENCES
parent_table(column_name);

> ALTER TABLE BANK_ACC
  ADD BID INT;

ALTER TABLE BANK_ACC
ADD CONSTRAINT FOREIGN KEY(BID) REFERENCES
BANK_BRANCHES(BID);

> To make PINCODE column as a foreign key in branch
which ust take the reference from location table.

ALTER TABLE BANK_BRANCHES
MODIFY PINCODE INT,
ADD CONSTRAINT FOREIGN KEY(PINCODE) REFERENCES
LOCATION(PINCODE);
```

9. To drop a constraint

```
ALTER TABLE table_name
DROP CONSTRAINT constraint_name;

ALTER TABLE BANK_ACC
DROP CONSTRAINT bank_acc_ibfk_1;

SELECT CONSTRAINT_NAME
FROM TABLE_CONSTRAINTS
WHERE TABLE_NAME='BANK_ACC';
```

10. To drop a Primary Key

```
ALTER TABLE table_name
DROP PRIMARY KEY;
```

> To remove Primary Key from Accounts table

```
ALTER TABLE BANK_ACC
DROP PRIMARY KEY;
```

11. To drop UNIQUE Constraint

```
ALTER TBALE table_name
DROP INDEX column_name;
```

DROP :

This command is used to delete a particular table from the database.
Syntax: DROP TABLE table_name;

```
DROP TABLE BRANCH;
```

DML(Data Manipulation Language)

INSERT : This command is used to add a new ecord inside the table.

Syntax1:

```
INSERT INTO
table_name(col1,col2,..,coln)VALUES(v1,v2,..,vn),(v1,v2,
..,vn)....
```

Synta2:

```
INSERT INTO table_name
VALUES(v1,v2,..,vn),(v1,v2,..,vn)....
```

```
INSERT INTO
LOCATION(PINCODE,AREA,CITY,STATE)VALUES(560040,'Vijayana
gar','Bengaluru','Karnataka');
```

```
ALTER TABLE LOCATION
MODIFY STATE VARCHAR(20) DEFAULT 'Karnataka' NOT NULL;

INSERT INTO
LOCATION(PINCODE,AREA,CITY)VALUES(560001,'MGRoad','Benga
luru');

INSERT INTO LOCATION(pincode,area,city)VALUES
(560023,'MagadiRoad','Bengaluru');
```

UPDATE

This command is used to modify the recods preent in the table.

Syntax:

```
UPDATE table_name
SET col1=val1[,col2=val2,..,coln=valn]
```

```
[WHERE filter_condition];

> To modify some records in location table
UPDATE LOCATION
SET AREA='MAHATMA_GANDHI_ROAD'
WHERE PINCODE=560001;

> To modify multiple columns,
UPDATE LOCATION
SET PINCODE=560072,AREA='Nagarabhavi'
WHERE PINCODE=560023;

UPDATE LOCATION
SET CITY='MUMBAI';

DELETE:
_______
This command is used to delete the particular record
from the table.
```

Syntax:

```
DELETE FROM table_name
[WHERE filter_condition];

DELETE FROM LOCATION;
```

1. WAQTD THE EMP FNAME,LAST NAME FROM EMP TABLE.

```
SELECT FNAME,LNAME
FROM EMP;
```

2. WAQTD THE EMP FIRST NAME,SALARY, DOB FROM THE EMP TABLE.

```
SELECT FNAME,SAL,JOB,DOB
FROM EMP;
```

3. WAQTD THE EMP FNAME,SAL ALONG WITH ANNUAL SALARY.

```
SELECT FNAME,SAL,SAL*12
FROM EMP;
```

4. WAQTD THE EMP FNAME,SAL,ANNUAL SALARY WITH 50000RS BONUS.

```
SELECT FNAME,SAL,(SAL*12)+50000
FROM EMP;
```

4. WAQTD THE DETAILS OF EMP ALONG WITH ANNUAL SALARY.

```
SELECT *,SAL*12
FROM EMP;
```

5. WAQTD THE EMP FIRST NAME, SALARY WITH 5000 HIKE.

```
SELECT FNAME,SAL+5000
FROM EMP;
```

6. WAQTD THE EMP FIRST NAME, SALARY WITH 5% HIKE.

```
SELECT FNAME,SAL*(1+0.05)
FROM EMP;
```

or

```
SELECT FNAME,SAL+(SAL*5/100)
FROM EMP;
```

7. WAQTD THE EMP FIRST NAME,SAL WITH 10% HIKE.

```
SELECT FNAME,SAL*(1+0.1)
FROM EMP;
```

or

```
SELECT FNAME,SAL+(SAL*10/100)
FROM EMP;
```

8. WAQTD THE EMP FIRST NAME,LAST NAME, SAL WITH 5% HIKE AND ANNUAL SALARY WITH 10 % HIKE.

```
SELECT FNAME,LNAME,SAL+(SAL*5/100),(SAL+(SAL*10/100))*12
FROM EMP;
```

or

```
SELECT FNAME,LNAME,SAL*(1+0.05),(SAL*12)*(1+0.1)
FROM EMP;
```

9. WAQTD THE EMP FIRST NAME,LAST NAME,SAL WITH 5% HIKE AND ANNUAL SALARY WITH 3% DEDUCTION.

```
SELECT FNAME,LNAME,SAL*(1+0.05),(SAL*12)*(1-0.03)
FROM EMP;
```

or

```
SELECT FNAME,LNAME,SAL+(SAL*5/100),(SAL-(SAL*3/100))*12
FROM EMP;
```

Projection

It is used to retrieve the data from the tbake by using the column names.

Syntax:

```
SELECT column_name
FROM table_name;
```

FROM clause:

> It is used to put the table under execution. ie, it checks for the availability of the table in the database and puts it under the execution.
> It is the first executable clause.

SELECT clause:

> It is used to display the records.

10. WAQTD THE EMP FIRST NAME, SALARY WITH 3% HIKE AND ANNUAL SALARY WITH 5% DEDUCTION.

```
SELECT FNAME,SAL+(SAL*3/100),(SAL-(SAL*5/100))*12
FROM EMP;
```

or

```
SELECT FNAME,SAL*(1+0.03) sal_3hike,SAL*12*(1-0.05) "5%
annual deduction"
FROM EMP;
```

ALIAS

> It is a alternative name given to the columns or the tables.
> We can pass alias name either by using AS keyword or "double quotes".
> With or without using AS keyword, we can pass alias name.
> If we want to use spaces or special characters, then we can use double quotes.

```
Syntax: SELECT column_name_1 AS alias_1,
                column_name_2 alias_2,
                column_name_3 "alias 3"
          FROM table_name;
```

11. WAQTD THE EMP FNAME,LAST NAEM AND JOB AS DESIGNATION, DOB AS BIRTH DATE.

```
SELECT FNAME,LNAME,JOB DESIGNATION,DOB BIRTHDATE
FROM EMP;
```

12. WAQTD THE DIFFERENT JOB ROLES AVAILABLE IN THE COMPANY.

```
SELECT DISTINCT JOB
FROM EMP;
```

DISTINCT

> It is used to avoid the duplicate records from the resultant table.
> Whenever we pass multiple columns inside DISTINCT clause, it checks for the combination.
> Either * or DISTINCT must be the very first arguement in SELECT clause.

Syntax:

```
SELECT DISTINCT column_name
```

```
FROM table_name;
```

13. WAQTD THE DIFFERENT DEPTS AVAILABLE IN THE COMPANY.

```
>> UPDATE EMP SET DNO=114 WHERE EID=1602;

SELECT DISTINCT DNO
FROM EMP;
```

14. WAQTD THE DETAILS OF EMP WHOSE FNAME IS AMAN.

```
SELECT *
FROM EMP
WHERE FNAME='AMAN';
```

15. WAQTD THE EMP FNAME, LAST NAME AND JOB ROLE IF THE EMP IS WORKING AS SALESMAN.

```
SELECT FNAME,LNAME,JOB
```

```
FROM EMP
WHERE JOB='SALESMAN';
```

16. WAQTD THE DETAILS OF EMP WHO ARE GETITNG SALARY MORE THAN 35000.

```
SELECT *
FROM EMP
WHERE SAL>35000;
```

17. WAQTD THE EMP FIRST NAME,LNAME AND DOB IF THE EMP WAS BORN AFTER THE YEAR 1995.

```
SELECT FNAME,LNAME,DOB
FROM EMP
WHERE DOB>'1995-12-31';
```

18. WAQTD THE EMP FIRST NAME,LNAME AND DOJ IF THE EMP WAS JOINED BEFORE THE YEAR 2020.

```
SELECT FNAME,LNAME,DOJ
FROM EMP
WHERE DOJ<'2020-01-01';
```

19. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY MORE THAN 32000 AND LESS THAN 50000.

OPERATORS

1. Arithmetic Operators(+,-,*,/,%)
2. Relational Operators(<,>,< =,> =,=,! =)
3. Logical Operators (AND, OR, NOT)
4. Special Operators (IN, NOT IN, IS, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE)
5. Sub Query operators (ALL,ANY,EXISTS, NOT EXISTS)

```
SELECT *
FROM EMP
```

```
WHERE SAL>32000 AND SAL<50000;
```

AND : The output will be true if all the conditions are true.

A	B	output
0	0	0
0	1	0
1	0	0
1	1	1

OR : The output will be true if any one of the condition is true.

A	B	output
0	0	0
0	1	1

```
1  0    1
1  1    1
```

20. WAQTD THE DETAILS OF EMP WHO ARE WORKING AS SALESMAN, MANAGER, DEVELOPER.

```
SELECT *
FROM EMP
WHERE JOB='SALESMAN' OR JOB='MANAGER' OR
JOB='DEVELOPER';
```

21. WAQTD THE EMP FNAME,LNAME AND DOB IF THE EMP WAS BORN DURING THE YEAR 1995.

```
SELECT FNAME,LNAME,DOB
FROM EMP
WHERE DOB≥'1995-01-01' AND DOB≤'1995-12-31';
```

22. WAQTD THE EMP FNAME,LNAME,DOB,JOB AND SALARY IF THE

EMP IS WORKING AS SALESMAN, MANAGER OR DISPATCHER AND GETTING SALARY MORE THAN 32000.

```
SELECT FNAME,LNAME,DOB,JOB,SAL
FROM EMP
WHERE (JOB='SALESMAN' OR JOB='MANAGER' OR
JOB='DISPATCHER') AND SAL>32000;
```

23. WAQTD THE DETAILS OF EMP WHO ARE WORKING AS ACCOUNTANT, DISPATCHER OR SALESMAN IN DEPT 110 OR 111 OR 112.

```
SELECT *
FROM EMP
WHERE (JOB='ACCOUNTANT' OR JOB='DISPATCHER' OR
JOB='SALESMAN') AND
       (DNO=110 OR DNO=111 OR DNO=112);

SELECT *
```

```
FROM EMP
WHERE JOB IN ('ACCOUNTANT','DISPATCHER','SALESMAN') AND
DNO IN (110,111,112);
```

IN/NOT IN OPERATOR

> It is a multivalued operator which takes single value at LHS and multiple values at RHS.
> Syntax: column_name IN/NOT IN(v1,v2,v3,..,vn);

> IN operator works in the form of OR operator.
> NOT IN operator works in the form of AND operator.

24. WAQTD THE DETAILS OF EMP ALONG WITH ANNUAL SALARY IF THE EMP IS GETTING ANNUAL SALARY MORE THAN 600000 AND WORKING AS SALESMAN, ACCOUNTANT OR MANAGER BUT NOT AS CEO.

```
SELECT *,SAL*12 ANNUAL_SAL
```

```
FROM EMP
WHERE SAL*12>600000 AND JOB IN
('SALESMAN','ACCOUNTANT','MANAGER') AND JOB≠'CEO';
```

25. WAQTD THE EMP FIRST NAME,LAST NAME AND JOB IF THE EMP IS NOT WORKING AS SALESMAN OR MANAGER OR DEVELOPER (Dont use special operator)

```
SELECT FNAME,LNAME,JOB
FROM EMP
WHERE JOB NOT IN ('SALESMAN','MANAGER','DEVELOPER');
```

26. WAQTD THE FNAME,LNAME,DOJ ALONG WITH SALARY IF THE LAST NAME MUST BE SHETTY, RAI OR RAJ BUT SAL LESS THAN 60000.

```
SELECT FNAME,LNAME,SAL,DOJ
FROM EMP
WHERE LNAME IN ('SHETTY','RAI','RAJ') AND SAL<60000;
```

27. WAQTD THE DETAILS OF EMP WHO ARE GETTING SLAARY MORE THAN OR EQUAL TO 32000 AND LESS THAN OR EQUAL TO 50000.

```
SELECT *
FROM EMP
WHERE SAL≥32000 AND SAL≤50000;
```

```
SELECT *
FROM EMP
WHERE SAL BETWEEN 32000 AND 50000;
```

BETWEEN/NOT BETWEEN OPERATOR

Whenever we come across range of values, we go for BETWEEN operator.

Syntax:

```
column_name BETWEEN/NOT BETWEEN lower_range AND
```

higher_range;

> = < =

>BETWEEN operator includes the range values.

28. WAQTD THE DETAILS OF EMP WHO WERE BORN DURING THE YEAR 1999.

```
SELECT *
FROM EMP
WHERE DOB BETWEEN '1999-01-01' AND '1999-12-31';
```

29. WAQTD THE DETAILS OF EMP WHO ARE GETTING SALARY MORE THAN 30000 AND LESS THAN 50000.

```
SELECT *
FROM EMP
WHERE SAL>30000 AND SAL<50000;
```

30. WAQTD THE DETAILS OF EMP WHOA RE GETTING THE SALARY IN THE RANGE OF 32000 TO 60000 BUT NOT HIRED IN THE YEAR 2022.

```
SELECT *
FROM EMP
WHERE (SAL BETWEEN 32000 AND 60000) AND
(DOJ NOT BETWEEN '2022-01-01' AND '2022-12-31');
```

31. WAQTD THE FNAME,LNAME AND CID IF THE EMP IS ONE OF THE CUSTOMER OF THE COMPANY.

```
SELECT FNAME,LNAME,CID
FROM EMP
WHERE CID IS NOT NULL;
```

IS Operator

It is used to check the given column whether the records

are NULL or NOT NULL.

Syntax
column_name IS NULL/NOT NULL;

32. WAQTD THE DETAILS OF EMP WHO ARE GETTING COMM.

```
SELECT *
FROM EMP
WHERE COMM IS NOT NULL;
```

33. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN OR MANAGER OR DEVELOPER OR DISPATCHER BUT NOT IN DEPT 113 AND GETTING SALARY IN THE RANGE OF 30000 TO 60000 BUT NOT HIRED IN THE YEAR 2016 AND ALSO HE MUST GET SOME COMM.

```
SELECT *
FROM EMP
```

```
WHERE JOB IN
('SALESMAN','MANAGER','DEVELOPER','DISPATCHER') AND
DNO≠113
AND SAL BETWEEN 30000 AND 60000 AND
DOJ NOT BETWEEN '2016-01-01' AND '2016-12-31' AND
COMM IS NOT NULL;
```

34. WAQTD THE DETAILS OF EMP WHO DOESNOT HAVE ANY REPORTING MANAGER.

```
SELECT *
FROM EMP
WHERE MGR IS NULL;
```

35. WAQTD THE FNAME AND LNAME OF THE EMP WHOSE FNAME STARTS WITH S.

```
SELECT FNAME,LNAME
FROM EMP
```

```
WHERE FNAME LIKE 'S%';
```

36. WAQTD THE DETAILS OF EMP WHOSE LAST NAME ENDS WITH I.

```
SELECT *
FROM EMP
WHERE LNAME LIKE '%I';
```

37. WAQTD THE FNAME,LAST NAME AND JOB IF THE EMP JOB ROLE CONTAINS MAN IN IT.

```
SELECT FNAME,LNAME,JOB
FROM EMP
WHERE JOB LIKE '%MAN%';
```

38. WAQTD THE EMP FIRST NAME, LAST NAME IF THE EMP FIRST NAME LAST 2ND SECOND CHARACTER IS A.

```
SELECT FNAME,LNAME
FROM EMP
WHERE FNAME LIKE '%A_';
```

39. WAQTD THE DETAILS OF EMP IF THE EMP FNAME STARTS WITH S OR A.

```
SELECT *
FROM EMP
WHERE FNAME LIKE 'S%' OR FNAME LIKE 'A%';
```

40. WAQTD THE FNAME,LNAME, DOB IF THE EMP WAS BORN DURING THE YEAR 1995 USING LIKE OPERATOR.

```
SELECT FNAME,LNAME,DOB
FROM EMP
WHERE DOB LIKE '1995%';
```

41. WAQTD THE DETAILS OF EMP WHO WERE HIRED IN THE MONTH

OF JAN, FEB OR MARCH.

```
SELECT *
FROM EMP
WHERE DOJ LIKE '%-01-%' OR DOJ LIKE '%-02-%' OR DOJ LIKE
'%-03-%';
```

42. WAQTD THE FNAME,LNAME OF THE EMPS IF THE EMP FNAME STARTS WITH S OR A AND FNAME ENDS WITH I OR N.

```
SELECT FNAME,LNAME
FROM EMP
WHERE (FNAME LIKE 'S%' OR FNAME LIKE 'A%') AND
(FNAME LIKE '%I' OR FNAME LIKE '%N');
```

43. WAQTD THE DETAILS OF EMPS WHOSE FNAME STARTS WITH VOWELS.

```
SELECT *
```

```
FROM EMP
WHERE FNAME LIKE 'A%' OR
      FNAME LIKE 'E%' OR
      FNAME LIKE 'I%' OR
      FNAME LIKE 'O%' OR
      FNAME LIKE 'U%';
```

44. WAQTD THE DETAILS OF EMP WHOSE LAST NAME MUST NOT START WITH R.

```
SELECT *
FROM EMP
WHERE LNAME NOT LIKE 'R%';
```

45. WAQTD THE DETAILS OF EMPS WHOSE LAST NAME MUST NOT END WITH VOWELS.

```
SELECT *
FROM EMP
```

```
WHERE LNAME NOT LIKE '%A' AND
      LNAME NOT LIKE '%E' AND
      LNAME NOT LIKE '%I' AND
      LNAME NOT LIKE '%O' AND
      LNAME NOT LIKE '%U';
```

46. WAQTD THE FNAME AND LNAME IF THE EMP FIRST NAME STARTS WITH A OR S BUT FNAME MUST NOT END WITH N OR I.

```
SELECT FNAME,LNAME
FROM EMP
WHERE (FNAME LIKE 'A%' OR FNAME LIKE 'S%') AND
      (FNAME NOT LIKE '%I' AND FNAME NOT LIKE '%N');
```

47. WAQTD THE DETAILS OF EMP WHO ARE WORKING AS SALESMAN, MANAGER OR DEVELOPER AND THEY WERE HIRED IN THE YEAR 2016, 2017 OR 2022.

```
SELECT *
```

```
FROM EMP
WHERE JOB IN ('SALESMAN','MANAGER','DEVELOPER') AND
      (DOJ LIKE '2016%' OR DOJ LIKE '2017%' OR DOJ LIKE
'2022%');
```

CREATE A TABLE CALLED STUD

48. WAQTD THE STUD NAME IF THE NAME CONTAINS ATLEAST 1 % IN IT.

```
SELECT NAME
FROM STUD
WHERE NAME LIKE '%\%%';
```

49. WAQTD THE STUD NAME IF THE NAME CONTAINS ATLEAST 1 _ IN IT.

```
SELECT NAME
FROM STUD
```

```
WHERE NAME LIKE '%\_%';
```

50. WAQTD THE STUD NAME WHICH CONTAINS ATLEAST 2 % IN IT.

```
'%A%A%'

'%\%%\%%'

SELECT NAME
FROM STUD
WHERE NAME LIKE '%\%%\%%';
```

FUNCTIONS
==========

It is a block of code which is used to perform a specific task.

Functions has been configured as

→ Inbuilt functions → User defined functions
  > Aggregate functions
  > Character functions
  > Number functions
  > Date functions

Aggregate functions

It takes n number of input and generates a single output.

Types of Aggregate Functions

MAX(): This function is used to obtain the maximum value from the given column.
Syntax: MAX(column_name);

MIN(): This function is used to obtain the minimum value from the given column.

Syntax: MIN(column_name);

SUM(): This function is used to obtain the total value from the given column.
Syntax: SUM(column_name);

AVG(): This function is used to obtain the average value from the given column.
Syntax: AVG(column_name);

COUNT(): This function is used to obtain the number of rows from the given column.
Syntax: COUNT(*/column_name);

Note: Only COUNT(*) takes * as a argument.

Characteristics of Aggregate functions

> Aggregate functions execute group by group.

> We cannot pass any other columns along with aggregate functions.
> We cannot nest aggregate functions.
> We cannot pass multiple columns along with aggregate functions.
> We cannot pass aggregate functions inside WHERE clause.
GROUP BY> We can pass group by expression along with aggragate functions
> Aggregate functions ignore null values.

51. WAQTD THE MAXIMUM SAL AND MINIMUM SAL FROM EMP TABLE.

```
SELECT MAX(SAL),MIN(SAL)
FROM EMP;
```

52. WAQTD THE TOTAL SALARY GIVEN TO ALL THE SALESMAN.

```
SELECT SUM(SAL)
FROM EMP
WHERE JOB='SALESMAN';
```

53. WAQTD THE AVG SALARY GIVEN TO EMPS WHOSE NAME STARTS WITH A OR K.

```
SELECT AVG(SAL)
FROM EMP
WHERE FNAME LIKE 'A%' OR FNAME LIKE 'K%';
```

54. WAQTD THE TOTAL SALARY, MAX SAL, MIN SALARY AND AVG SALARY IN THE EMP TABLE.

```
SELECT SUM(SAL),MAX(SAL),MIN(SAL),AVG(SAL)
FROM EMP;
```

55. WAQTD THE DOJ OF THE FIRST EMP IN THE COMPANY.

```
SELECT MIN(DOJ)
FROM EMP;
```

56. WAQTD THE NUMBER OF EMPS WORKING IN THE COMPANY.

```
SELECT COUNT(*)
FROM EMP;
```

57. WAQTD THE NUMBER OF DIFFERENT JOB ROLES AVAILABLE IN THE COMPANY.

```
SELECT COUNT(DISTINCT JOB)
FROM EMP;
```

58. WAQTD THE DIFFERENCE BETWEEN MAXIMUM SALARY AND MINIMUM SALARY.

```
SELECT MAX(SAL)-MIN(SAL)
FROM EMP;
```

59. WAQTD THE NUMBER OF EMPS WORKING IN DEPT 113.

```
SELECT COUNT(*)
FROM EMP
WHERE DNO=113;
```

60. WAQTD THE NUMBER OF EMPS WORKING IN DEPT 112 AND 111.

```
SELECT COUNT(*)
FROM EMP
WHERE DNO IN (112,111);
```

61. WAQTD THE NUMBER OF EMPS WORKING IN EACH DEPT.

```
SELECT COUNT(*),DNO
FROM EMP
GROUP BY DNO;
```

GROUP BY clause

GROUP BY clause is used to group the records.

Syntax:

```
SELECT aggregate_functions/group_by_expression
FROM table_name
[WHERE filter_condition]
GROUP BY column_name;
         group_by_expression
```

Characteristics of GROUP BY clause

> It executes row by row.
> GROUP BY clause executes after FROM clause if there is no WHERE clause.
> GROUP BY clause con`verts the row records into group

records.
> We can pass multiple columns inside GROUP BY clause.
> After the execution of GROUP BY clause, the other clauses execute group by group.
> We can display only Aggregate functions and group by expression.

Note: The column which we pass as a argument inside GROUP BY is known as group_by_expression.

62. WAQTD THE MAXIMUM SALARY AND MINIMUM SALARY GIVEN IN EACH DEPT.

```
SELECT MAX(SAL),MIN(SAL),DNO
FROM EMP
GROUP BY DNO;
```

63. WAQTD THE NUMBER OF EMPS WORKING IN EACH JOB ROLE.

```
SELECT COUNT(*),JOB
FROM EMP
GROUP BY JOB;
```

64. WAQTD THE MAXIMUM AND MINIMUM GIVEN TO ALL THE SALESMAN OR DISPATCHER OR ACCOUNTANT IN EVERY DEPT.

```
SELECT MAX(SAL),MIN(SAL),DNO
FROM EMP
WHERE JOB IN ('SALESMAN','DISPATCHER','ACCOUNTANT')
GROUP BY DNO;
```

65. WAQTD THE DOJ OF FIRST EMP AND LAST EMP IN EVERY DEPT.

```
SELECT MIN(DOJ) first_joiner, MAX(DOJ) recent_joiner,DNO
FROM EMP
GROUP BY DNO;
```

66. WAQTD THE NUMBER OF EMPS WHO ARE GETTING SALARY MORE THAN 35000 AND SALARY LESS THAN 100000 IN EACH DEPT.

```
SELECT COUNT(*),DNO
FROM EMP
WHERE SAL>35000 AND SAL<100000
GROUP BY DNO;
```

HAVING CLAUSE
==========

HAVING CLAUSE is used to filter the group records.

Syntax:

```
SELECT aggregate_functions/group_by_expression
FROM table_name
[WHERE filter_condition]
GROUP BY column_name
HAVING filter_group_condition;
```

Characteristics of HAVING clause

> HAVING clause executes group by group.
> It is used to filter the group conditions.
> It executes after the GROUP BY clause.
> We can pass multiple group conditions inside HAVING clause.

67. WAQTD THE NUMBER OF EMPS WHO ARE GETTING SALARY MORE THAN OR EQUAL TO 30000 AND MAX SALARY LESS THAN OR EQUAL TO 150000 IN EACH DEPT.

```
SELECT COUNT(*),DNO
FROM EMP
WHERE SAL≥30000
GROUP BY DNO
HAVING MAX(SAL)≤150000;
```

68. WAQTD THE MAXIMUM SALARY AND THE MINIMUM SALARY GIVEN TO ALL THE EMPS IN EACH DEPT IF THE DEPT AVG SALARY IS LESS THAN 70000.

```
SELECT MAX(SAL),MIN(SAL),DNO
FROM EMP
GROUP BY DNO
HAVING AVG(SAL)<70000;
```

69. WAQTD THE AVG SALARY AND TOTAL SALARY GIVEN IN EACH JOB ROLE IF THE JOB ROLE CONTAINS MORE THAN 2 WORKERS.

```
SELECT AVG(SAL),SUM(SAL),JOB
FROM EMP
GROUP BY JOB
HAVING COUNT(*)>2;
```

70. WAQTD THE MAX SALARY GIVEN TO EACH EMP IF THE DEPT CONTAINS ATLEAST 4 EMPS IN THE DEPT.

```
SELECT MAX(SAL),DNO
FROM EMP
GROUP BY DNO
HAVING COUNT(*)≥4;
```

71. WAQTD THE MAXIMUM DOB AND MINIMUM DOB IF THE EMP WAS BORN DURING THE YEAR 1990 TO 1999 IN EACH DEPT.

```
SELECT MAX(DOB),MIN(DOB),DNO
FROM EMP
WHERE DOB BETWEEN '1990-01-01' AND '1999-12-31'
GROUP BY DNO;
```

72. WAQTD THE NUMBER OF EMPS WORKING IN EACH DEPT IF DEPT S MINIMUM SALARY MORE THAN 32000 AND MAXIMUM SALAARY LESS THAN 150000.

```
SELECT COUNT(*),DNO
```

```
FROM EMP
GROUP BY DNO
HAVING MIN(SAL)>32000 AND MAX(SAL)<150000;
```

73. WAQTD THE NUMBER OF EMPS WHO ARE GETTING SAME SALARY.

```
SELECT COUNT(*),SAL
FROM EMP
GROUP BY SAL
HAVING COUNT(*)≥2;
```

74. WAQTD THE NUMBER OF EMPS WHO ARE GETTING SAME SALARY AND WORKING IN SAME DEPT.

```
SELECT COUNT(*),SAL,DNO
FROM EMP
GROUP BY SAL,DNO
HAVING COUNT(*)>1;
```

75. WAQTD THE NUMBER OF EMPS REPORTING TO EACH MANAGER.

```
SELECT COUNT(*),MGR
FROM EMP
WHERE MGR IS NOT NULL
GROUP BY MGR;
```

76. WAQTD THE NUMBER OF EMPS WHO HAVE SAME GENDER AND WORKING IN SAME DEPT.

```
SELECT COUNT(*),GENDER,DNO
FROM EMP
GROUP BY GENDER,DNO
HAVING COUNT(*)>1;
```

77. WAQTD THE DETAILS OF EMPS ACCORDING TO THEIR SALARY FROM MAXIMUM TO MINIMUM.

```
SELECT *
FROM EMP
ORDER BY SAL DESC;
```

ORDER BY clause

It is used to arrange the records either in ascending or descending order based on a specific column.

Syntax:

```
SELECT column_name
FROM table_name
ORDER BY column_name ASC/DESC;
```

Characteristics of ORDER BY clause

> It is the last executable clause.
> It executes after the SELECT clause.

> By default, all the tables by default will be arranged based on Primary Key.
> We can pass multiple columns inside ORDER BY clause.

78. WAQTD THE EMP FNAME, LNAME IF THE EMP IS WORKING AS SALESMAN OR DEVELOPER OR TESTER OR DISPATCHER. ARRANGE THE RTECORDS BASED ON THEIR FNAME IN ALPHABETICAL ORDER.

```
SELECT FNAME,LNAME
FROM EMP
WHERE JOB IN
('SALESMAN','DEVELOPER','DISPATCHER','TESTER')
ORDER BY FNAME ASC;
```

79. WAQTD THE NUMBER OF EMPS WHO ARE GETTING SAME SALARY AND WORKING IN SAME JOB ROLE IF THE EMP SALARY IS MORE THAN 30000. ARRANGE THE RECORDS BASED ON THEIR SALARY.

```
SELECT COUNT(*),JOB,SAL
```

```
FROM EMP
WHERE SAL>30000
GROUP BY JOB,SAL
HAVING COUNT(*)>1
ORDER BY SAL;
```

LIMIT AND OFFSET

LIMIT: It is used to return the specified number of rows.

Syntax: LIMIT no_of_rows_to_return;

80. WAQTD THE FIRST 3 RECORDS FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 3;
```

81. WAQTD THE FIRST 5 RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 5;
```

82. WAQTD THE FIRST RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 1;
```

83. WAQTD THE SECOND RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 1 OFFSET 1;
```

84. WAQTD THE 5TH RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 1 OFFSET 4;
```

85.WAQTD THE 7TH RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 1 OFFSET 6;
```

86. WAQTD THE 6TH AND 7TH RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 2 OFFSET 5;
```

87. WAQTD THE THE FIRST 5 MAXIMUM SALARY HOLDERS.

```
SELECT *
FROM EMP
ORDER BY SAL DESC
LIMIT 5;
```

88. WAQTD THE 6TH MAX SALARY.(DISPLAY ONLY SALARY)

```
SELECT DISTINCT SAL
FROM EMP
ORDER BY SAL DESC
LIMIT 1 OFFSET 5;
```

89. WAQTD THE 7TH MAX SALARY.

```
SELECT DISTINCT SAL
FROM EMP
ORDER BY SAL DESC
LIMIT 1 OFFSET 6;
```

90. WAQTD THE FIRST 5 EMPS WHO WERE HIRED IN THE COMPANY.

```
SELECT *
FROM EMP
ORDER BY DOJ
LIMIT 5;
```

91. WAQTD THE LAST 5 RECORDS.

```
SELECT *
FROM EMP
ORDER BY EID DESC
LIMIT 5;
```

92. WAQTD THE LAST 3RD RECORD FROM EMP TABLE.

```
SELECT *
```

```
FROM EMP
ORDER BY EID DESC
LIMIT 1 OFFSET 2;

93. WAQTD THE 5TH MINIMUM SALARY.

SELECT DISTINCT SAL
FROM EMP
ORDER  BY SAL ASC
LIMIT 1 OFFSET 4;

Character functions
====================
1. UPPER(): This function is used to convert the string
values to UPPER case.
Syntax: SELECT UPPER('pentagon');

2. LOWER(): This function is used to convert the string
values to lower case.
```

Syntax: SELECT LOWER('PENTAGON');

3. LENGTH(): This function is used to obtain the length of the given string.
Syntax: SELECT LENGTH('PENTAGON');

4. REVERSE(): This function is used to reverse the given string.
Syntax: SELECT REVERSE('pentagon');

5. CONCAT(): This function is used to combine the given strings.
Syntax: CONCAT('str1','str2',...,'str n');

94. WAQTD THE FULLNAME OF AN EMP AND ARRANGE THE NAMES ACCORDING TO ALPHABETICAL ORDER.

SIDDARTH PATIL

```
SELECT CONCAT(FNAME,' ',LNAME) FULL_NAME
FROM EMP
ORDER BY FULL_NAME ASC;
```

6. ***SUBSTR(): This function is use to obtain the part of a string.
Syntax: SUBSTR('original_string',Position[,Length]);

95. WAQTD THE DETAILS OF EMP WHOSE FNAME STARTS WITH S.

```
SELECT *
FROM EMP
WHERE SUBSTR(FNAME,1,1)='S';
```

96. WAQTD THE FNAME,LNAME FROM EMP WHOSE FNAME STARTS WITH S OR A.

```
SELECT FNAME,LNAME
FROM EMP
```

```
WHERE SUBSTR(FNAME,1,1)IN('S','A');
```

97. WAQTD THE FNAME,LNAME FROM EMP WHOSE FNAME ENDS WITH A.

```
SELECT FNAME,LNAME
FROM EMP
WHERE SUBSTR(FNAME,-1,1)='A';
```

98. WAQTD THE FNAME AND LNAME IF THE FNAME STARTS WITH VOWELS OR LAST NAME ENDS WITH VOWELS.

```
SELECT FNAME,LNAME
FROM EMP
WHERE SUBSTR(FNAME,1,1)IN('A','E','I','O','U') OR
SUBSTR(LNAME,-1,1)IN('A','E','I','O','U');
```

99. WAQTD THE DETAILS OF EMP WHO WERE HIRED IN THE MONTH

OF AUG SEPT OR OCT.

```
SELECT *
FROM EMP
WHERE SUBSTR(DOJ,6,2)IN(08,09,10);
```

100. WAQTD THE FIRST HALF OF THE FNAME.

```
SELECT  FROM EMP;

SELECT SUBSTR(FNAME,1,LENGTH(FNAME)/2)
FROM EMP;
```

101. WAQTD THE SECONS HALF OF THE FNAME.

```
SELECT SUBSTR(FNAME,LENGTH(FNAME)/2+1)
FROM EMP;
```

102. WAQTD THE FIRST HALF OF FNAME IN LOWER CASE AND

SECOND HALF IN UPPER REVERSE CASE.

SIDDARTH → siddHTRA

```
SELECT CONCAT(LOWER(SUBSTR(FNAME,1,LENGTH(FNAME)/2)),
REVERSE(UPPER(SUBSTR(FNAME,LENGTH(FNAME)/2+1))))
fullEMAN
FROM EMP;
```

REPLACE():
==========

It is used to replace the substring with a new string from the original string.
Syntax: REPLACE('original_string','substr','new-str');

```
SELECT REPLACE('PENTAGON','PEN','G');

SELECT REPLACE('SANJAY','SAN','007');
```

```
SELECT REPLACE('SANJAY','A','');

input 1    input 2  output
=============================
MALAYALAM    A        4
ASHWINI      I        2

SELECT
LENGTH('MALAYALAM')-LENGTH(REPLACE('MALAYALAM','A',''));
```

103. WAQTD THE DETAILS OF EMP IF THE EMP NAME CONTAINS ATLEAST 1 E WITHOUT USING LIKE OPERATOR.

```
SELECT *
FROM EMP
WHERE LENGTH(FNAME)-LENGTH(REPLACE(FNAME,'E',''))≥1;
```

104. WAQTD THE EMP FNAME,LNAME IF THE EMP FNAME CONTAINS ATLEAST 2 A IN IT.

```
SELECT FNAME,LNAME
FROM EMP
WHERE LENGTH(FNAME)-LENGTH(REPLACE(FNAME,'A',''))≥2;
```

105. WAQTD THE EMP FNAME IF THE EMP FNAME EXACTLY ONE A IN IT.

```
SELECT FNAME,LNAME
FROM EMP
WHERE LENGTH(FNAME)-LENGTH(REPLACE(FNAME,'A',''))=1;
```

NUMBER FUNCTIONS
==========

ABS(): This function is used to convert the negative numbers into positive numbers.
Syntax: ABS(m);
Ex: SELECT ABS(-20);

MOD(); This function is used to obtain the remainder of the given two numbers.
Syntax:MOD(m,n);
Ex: SELECT MOD(4,2);

ROUND(): This function is used to round off the given floating values.
Syntax:ROUND(m,n);
Ex: SELECT ROUND(4.79);

CEIL(): This function is used to obtain the next integer value from the given decimal number.

FLOOR(): This function is used to obtain the current integer value from the given decimal number.

POW(): This function is used to obtain the power of the given number. It takes the integer value and the degree value as a arguement.

Ex: SELECT POW(3,4);

DATE FUNCTIONS

SYSDATE(): It is used to obtain the current date and time.

```
SELECT SYSDATE();
```

CURDATE(): It is used to obtain the current date.

```
SELECT CURDATE();
```

DATEDIFF(): It is used to obtain the differences between two date values.

```
SELECT DATEDIFF(CURDATE(),'2024-05-07');
```

DATE_ADD(): This function is used to add the time intervals.

```
SELECT DATE_ADD(SYSDATE(),INTERVAL 5 SECOND);
```

DATE_SUB(): This function is used to subtract the time intervals.

```
SELECT DATE_SUB(SYSDATE(),INTERVAL 2 HOUR);
```

DATE_FORMAT(): This function is used to convert the date values into individual characters based on Format Model.

Syntax: DATE_FORMAT(date_value,'format_model');

106. WAQTD THE DETAILS OF EMP WHO WERE HIRED IN THE YEAR 2021 USING DATE_FORMAT().

```
SELECT *
FROM EMP
WHERE DATE_FORMAT(DOJ,'%Y')=2021;
```

107. WAQTD THE EMP FNAME, DOB IF THE EMPS WERE BORN IN THE MONTH OF MAY, JUN ,JUL OR AUG.

```
SELECT FNAME,DOB
```

```
FROM EMP
WHERE DATE_FORMAT(DOB,'%m')IN(05,06,07,08);
```

108. WAQTD THE DETAILS OF EMP WHO WERE HIRED DURING EVEN DATES.

```
SELECT *
FROM EMP
WHERE MOD(DATE_FORMAT(DOJ,'%d'),2)=0;
```

109. WAQTD THE DETAILS OF EMP WHO WERE HIRED ON SATURDAY AND SUNDAY.

```
SELECT *
FROM EMP
WHERE DATE_FORMAT(DOJ,'%a')IN('SAT','SUN');
```

110. WAQTD THE FNAME,DOB IN US FORMAT.

```
MM-DD-YYYY
SELECT FNAME,DATE_FORMAT(DOB,'%m/%d/%Y') US_FORMAT
FROM EMP;
```

111. WAQTD THE DETAILS OF EMP WHO WERE HIRED DURING LEAP YEAR.

```
SELECT *
FROM EMP
WHERE (MOD(DATE_FORMAT(DOJ,'%Y'),4)=0 OR
MOD(DATE_FORMAT(DOJ,'%Y'),400)=0) AND
      MOD(DATE_FORMAT(DOJ,'%Y'),100)≠0;
```

112. WAQTD THE NUMBER OF EMPS HIRED DURING EVERY YEAR. ARRANGE THE RECORDS BASED ON NUMBER OF HIRINGS FROM MAX TO MIN.

```
SELECT COUNT(*) "No. of emp hired",
```

```
DATE_FORMAT(DOJ,'%Y') Year
FROM EMP
GROUP BY DATE_FORMAT(DOJ,'%Y')
ORDER BY COUNT(*) DESC;

SELECT DATE_FORMAT(DOJ,'%Y')
FROM EMP;
```

113. WAQTD THE FNAME,LNAME,JOB,DOJ AND NO. OF YEARS OF EXPERIENCE, THE EMP HAS.

```
SELECT
FNAME,LNAME,JOB,DOJ,DATE_FORMAT(CURDATE(),'%Y')-DATE_FOR
MAT(DOJ,'%Y') Experience
FROM EMP;
```

114. WAQTD THE NUMBER OF EMPS HIRED IN EACH MONTH.

```
SELECT COUNT(*),DATE_FORMAT(DOJ,'%M') Months
```

```
FROM EMP
GROUP BY DATE_FORMAT(DOJ,'%M'),DATE_FORMAT(DOJ,'%m')
ORDER BY DATE_FORMAT(DOJ,'%m');
```

SUB QUERY

A query inside another query is known as SUb query.

Working Principle of Sub query

> Sub query consisting of Inner query and outer query.
> Always Inner query will execute first and generates the output.
> The output of the inner query will be given as input to the outer query.
> The outer query collects the input and executes completely to generate the final result.
> Here, the outer query depends upon the inner query.

Rules to write Sub query

1. WHERE SAL>(SELECT FNAME - wrong
   decimal ≠ varchar
   WHERE SAL>(SELECT COMM - right
   decimal = decimal

In subquery, the datatypes must match between inner query and outer query.

2. WHERE SAL=(SELECT SAL,COMM - wrong
   WHERE SAL=(SELECT SAL - right

In subquery, we can pass only single column inside the Inner query.

When or where we can go for sub query

CASE 1: Whenever we come across unknown values, we go for sub query.

CASE 2:

115. WAQTD THE DETAILS OF EMP WHO ARE GETTING SAL MORE THAN KIRAN'S SALARY.

```
SELECT * FROM EMP WHERE SAL>
(SELECT SAL FROM EMP WHERE FNAME='KIRAN');
```

116. WAQTD THE FNAME,LNAME,SAL WHO ARE WORKING IN SAME JOB ROLE AS KARAN'S JOB ROLE.

```
SELECT FNAME,LNAME,SAL,JOB FROM EMP WHERE JOB=
(SELECT JOB FROM EMP WHERE FNAME='KARAN');
```

117. WAQTD THE DETAILS OF EMP WHO ARE WORKING AS SALESMAN AND GETTING SALARY MORE THAN DHARANI.

```
SELECT * FROM EMP WHERE JOB='SALESMAN' AND SAL>
(SELECT SAL FROM EMP WHERE FNAME='DHARANI');
```

118. WAQTD THE EMP FNAME,SAL,DNO IF THE EMP IS WORKING IN SAME DEPT AS MURALI'S DEPT AND GETTING SALARY MORE THAN FARIYA.

```
SELECT FNAME,SAL,DNO FROM EMP WHERE DNO=
(SELECT DNO FROM EMP WHERE FNAME='MURALI')
AND SAL>
(SELECT SAL FROM EMP WHERE FNAME='FARIYA');
```

119. WAQTD THE EMP FNAME,SAL IF THE EMP IS GETTING SALARY MORE THAN DHARANI AND LESS THAN SAMEER.

```
SELECT FNAME,SAL FROM EMP WHERE SAL>
(SELECT SAL FROM EMP WHERE FNAME='DHARANI')
AND SAL<
(SELECT SAL FROM EMP WHERE FNAME='SAMEER');
```

120. WAQTD THE DETAILS OF EMP WHO WAS HIRED AFTER 2

YEARS OF FIRST EMP OF THE COMPANY.

```
SELECT * FROM EMP WHERE DOJ>
(SELECT DATE_ADD(MIN(DOJ),INTERVAL 2 YEAR) FROM EMP);
```

121. WAQTD THE FNAME, DOB IF THE EMP IS 2 YEARS OR MORE YOUNGER THAN RAHUL.

```
SELECT FNAME,DOB FROM EMP WHERE DOB>
(SELECT DATE_ADD(DOB,INTERVAL 2 YEAR) FROM EMP WHERE
FNAME='RAHUL');
```

122. WAQTD THE DETAILS OF EMP IF THE EMP WAS HIRED 2 YEARS BEFORE KIRAN'S DOJ.

```
SELECT * FROM EMP WHERE DOJ<
(SELECT DATE_SUB(DOJ,INTERVAL 2 YEAR) FROM EMP WHERE
FNAME='KIRAN');
```

123. WAQTD THE DEPT NAME OF SIDDARTH.

```
SELECT DNAME FROM DEPT WHERE DNO=
(SELECT DNO FROM EMP WHERE FNAME='SIDDARTH');
```

124. WAQTD THE DETAILS OF CUSTOMER WHO LIVES IN VIJAYANAGAR.

```
SELECT * FROM CUSTOMER WHERE LID=
(SELECT LID FROM LOCATION WHERE LOCATION='VIJAYANAGAR');
```

125. WAQTD THE DETAILS OF EMP WHO WORKS IN MARKETING DEPT.

```
SELECT * FROM EMP WHERE DNO=
```

126. WAQTD THE DETAILS OF EMP WHO WORKS IN MARKETING OR

TECHNICAL DEPT.

```
SELECT * FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE DNAME IN
('MARKETING','TECHNICAL'));
```

Types of Sub query

Single Row Sub query: When the inner query returns one row, then it is known as Single row sub query.
We can use normal operators(=,≠,≥,≤,<,>) or Special operators(IN,NOT IN,ALL,ANY).

Multi Row sub query: When the inner query returns more than one row, then it is known as Multi row sub query.
We can only use special operators(IN,NOT IN,ALL,ANY).

127. WAQTD THE DETAILS OF EMP WHO ARE GETITNG SALARY MORE THAN PRIYA AND WORKING IN TECHNICAL DEPT.

```
SELECT * FROM EMP WHERE SAL>
(SELECT SAL FROM EMP WHERE FNAME='PRIYA')
AND DNO IN
(SELECT DNO FROM DEPT WHERE DNAME='TECHNICAL');
```

128. WAQTD THE DEPT NAME OF THE EMP WHO ARE WORKING IN SAME JOB ROLE AS AMAN'S JOB ROLE.

```
SELECT DNAME FROM DEPT WHERE DNO IN
(SELECT DNO FROM EMP WHERE JOB IN
(SELECT JOB FROM EMP WHERE FNAME='AMAN'));
```

129. WAQTD THE DETAILS OF EMP WHO ARE WORKING IN TECHNICAL DEPT.

```
SELECT * FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE DNAME='TECHNICAL');
```

```
ALTER TABLE ORDERS
ADD CID INT;

UPDATE ORDERS SET CID=505 WHERE ORDER_ID=1111;
UPDATE ORDERS SET CID=507 WHERE ORDER_ID=1127;
UPDATE ORDERS SET CID=501 WHERE ORDER_ID=1201;
UPDATE ORDERS SET CID=511 WHERE ORDER_ID=1773;
```

130. WAQTD THE CUSTOMER FIRST_NAME,LAST_NAME IF THE CUSTOMER LIVES IN MUMBAI CITY.

```
SELECT FIRST_NAME,LAST_NAME FROM CUSTOMER WHERE LID IN
(SELECT LID FROM LOCATION WHERE CITY='MUMBAI');
```

131. WAQTD THE CUSTOMER ANME WHO HAS ORDERED ANY PRODUCTS.

```
SELECT FIRST_NAME,LAST_NAME FROM CUSTOMER WHERE CID IN
(SELECT CID FROM ORDERS);
```

132. WAQTD THE LOCATION OF MARKETING DEPT.

```
SELECT LOCATION FROM LOCATION WHERE LID IN
(SELECT LID FROM DEPT WHERE DNAME='MARKETING');
```

133. WAQTD THE WORKING LOCATION OF KIRAN.

```
SELECT LOCATION FROM LOCATION WHERE LID IN
(SELECT LID FROM DEPT WHERE DNO IN
(SELECT DNO FROM EMP WHERE FNAME='KIRAN'));
```

134. WAQTD THE CUSTOMER FIRST NAME WHO HAS ORDERED THE WALLET(PNAME).

```
SELECT FIRST_NAME FROM CUSTOMER WHERE CID IN
(SELECT CID FROM ORDERS WHERE PRODUCT_ID IN
(SELECT PRODUCT_ID FROM PRODUCT WHERE PNAME='WALLET'));
```

135. WAQTD THE DETAILS OF EMP WHO HAS DELIVERED THE PRODUCT.

```
SELECT * FROM EMP WHERE EID IN
(SELECT EID FROM ORDERS);
```

136. WAQTD THE DETAILS OF EMP WHO HAS DELIVERED THE PRODUCT TO MUMBAI CITY.

```
SELECT * FROM EMP WHERE EID IN
(SELECT EID FROM ORDERS WHERE CID IN
(SELECT CID FROM CUSTOMER WHERE LID IN
(SELECT LID FROM LOCATION WHERE CITY='MUMBAI')));
```

137. WAQTD THE LOCATION, CITY, STATE AS ADDRESS WHERE THE WALLET HAS BEEN DELIVERED.

```
SELECT CONCAT(LOCATION,', ',CITY,', ',STATE) ADDRESS
FROM LOCATION WHERE LID IN
```

```
(SELECT LID FROM CUSTOMER WHERE CID IN
(SELECT CID FROM ORDERS WHERE PRODUCT_ID IN
(SELECT PRODUCT_ID FROM PRODUCT WHERE PNAME='WALLET')));
```

138. WAQTD THE DETAILS OF EMP WHO WORKS IN MYSORE CITY.

```
SELECT * FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE LID IN
(SELECT LID FROM LOCATION WHERE CITY='MYSORE'));
```

139. WAQTD THE NUMBER OF EMPS WORKING IN VIJAYANAGAR.

```
SELECT COUNT(*) FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE LID IN
(SELECT LID FROM LOCATION WHERE
LOCATION='VIJAYANAGAR'));
```

140. WAQTD THE DETAILS OF EMP WHO ARE GETTING 5TH MAX SALARY.

```
SELECT * FROM EMP WHERE SAL=
(SELECT DISTINCT SAL
FROM EMP
ORDER BY SAL DESC
LIMIT 1 OFFSET 4);
```

141. WAQTD THE DETAILS OF EMP WHO ARE GETTING THE TOP 5 SALARY.

```
SELECT * FROM EMP WHERE SAL IN
(SELECT SAL FROM
(SELECT DISTINCT SAL FROM EMP ORDER BY SAL DESC LIMIT 5)
RES_TABLE);
```

142. WAQTD THE DETAILS OF EMP WHO ARE GETTING TOP 3 MINIMUM SALARY.

```
SELECT * FROM EMP WHERE SAL IN
(SELECT SAL FROM
(SELECT DISTINCT SAL FROM EMP ORDER BY SAL ASC LIMIT
3)RES_TABLE)
ORDER BY SAL ASC;
```

RANK() AND DENSE_RANK()
==========

These functions are used to assign the ranks for every row based on certain condition.

```
SELECT FNAME,SAL,DENSE_RANK() OVER (ORDER BY SAL DESC)
FROM EMP;
```

143. WAQTD THE DETAILS OF EMP WHO ARE GETTING SALARY MORE THAN KIRAN.

```
SELECT * FROM EMP WHERE SAL>
(SELECT SAL FROM EMP WHERE FNAME='KIRAN');
```

144. WAQTD THE DETAILS OF EMP WHO ARE GETTING SALARY MORE THAN ALL THE SALESMAN.

```
SELECT * FROM EMP WHERE SAL>
(SELECT MAX(SAL) FROM EMP WHERE JOB='SALESMAN');
```

OR

```
SELECT * FROM EMP WHERE SAL>ALL
(SELECT SAL FROM EMP WHERE JOB='SALESMAN');
```

ALL/ANY OPERATOR

These are the multivalued operator which takes single value at LHS and multiple values at RHS.

Syntax: column_name >/< ALL/ANY (SELECT

multi_row_subquery

> ALL Operator works in the form of AND operator.
> ANY Operator works in the form of OR operator.

145. WAQTD THE DETAILS OF EMP WHO ARE GETTING SALARY MORE THAN ANY ONE OF THE DISPATCHER.

```
SELECT * FROM EMP WHERE SAL>ANY
(SELECT SAL FROM EMP WHERE JOB='DISPATCHER');
```

146. WAQTD THE DETAILS OF EMP WHO ARE ALSO THE CUSTOMERS BUT NOT ORDERED ANY PRODUCT.

```
SELECT * FROM EMP WHERE CID NOT IN
(SELECT CID FROM ORDERS);
```

147. WAQTD THE EMP FIRST NAME, LAST NAME,SAL IF THE EMP IS GETTING SALARY MORE THAN AVERAGE SALARY OF DNO 111.

```
SELECT FNAME,LNAME,SAL FROM EMP WHERE SAL>
(SELECT AVG(SAL) FROM EMP WHERE DNO=111);
```

148. WAQTD THE CUSTOMER NAME, WHO HAS ORDERED THE PRODUCT FROM VIJAYANAGAR.

```
SELECT CID FROM CUSTOMER WHERE LID IN
(SELECT LID FROM LOCATION WHERE LOCATION='VIJAYANAGAR')
AND
CID IN (SELECT CID FROM ORDERS);
```

149. WAQTD THE DETAILS OF KIRAN'S MANAGER.

```
SELECT * FROM EMP WHERE EID IN
(SELECT MGR FROM EMP WHERE FNAME='KIRAN');
```

150. WAQTD THE FNAME, LNAME OF FARIYA'S MANAGER.

```
SELECT FNAME,LNAME FROM EMP WHERE EID IN
(SELECT MGR FROM EMP WHERE FNAME='FARIYA');
```

151. WAQTD THE DEPT NAME OF KARAN'S MANAGER.

```
SELECT DNAME FROM DEPT WHERE DNO IN
(SELECT DNO FROM EMP WHERE EID IN
(SELECT MGR FROM EMP WHERE FNAME='KARAN'));
```

152. WAQTD THE WORKING LOCATION OF AMAN'S MANAGER.

```
SELECT LOCATION FROM LOCATION WHERE LID IN
(SELECT LID FROM DEPT WHERE DNO IN
(SELECT DNO FROM EMP WHERE EID IN
(SELECT MGR FROM EMP WHERE FNAME='AMAN')));
```

153. WAQTD THE FULL NAME OF HEMA'S MANAGER.

```
SELECT CONCAT(FNAME,' ',LNAME) FULL_NAME FROM EMP WHERE
```

```
EID IN
(SELECT MGR FROM EMP WHERE FNAME='HEMA');
```

154. WAQTD THE DETAILS OF EMP WHO ARE NOT WORKING IN VIJAYANAGAR.

```
SELECT * FROM EMP WHERE DNO NOT IN
(SELECT DNO FROM DEPT WHERE LID IN
(SELECT LID FROM LOCATION WHERE
LOCATION='VIJAYANAGAR'));
```

155. WAQTD THE KIRAN'S MANAGER'S MANAGER.

```
SELECT * FROM EMP WHERE EID IN
(SELECT MGR FROM EMP WHERE EID IN
(SELECT MGR FROM EMP WHERE FNAME='KIRAN'));
```

156. WAQTD THE DETAILS OF EMPS WHO ARE REPORTING TO SAMEER.

```
SELECT * FROM EMP WHERE MGR IN
(SELECT EID FROM EMP WHERE FNAME='SAMEER');
```

157. WAQTD THE DEPT NAME OF EMPS WHO ARE REPORTING TO SIDDARTH.

```
SELECT DNAME FROM DEPT WHERE DNO IN
(SELECT DNO FROM EMP WHERE MGR IN
(SELECT EID FROM EMP WHERE FNAME='SIDDARTH'));
```

158. WAQTD THE DETAILS OF EMP WHOA ARE REPORTING TO KIRAN'S MANAGER.

```
SELECT * FROM EMP WHERE MGR IN
(SELECT MGR FROM EMP WHERE FNAME='KIRAN');
```

159. WAQTD THE DETAILS OF EMP WHO ARE NOT WORKING AS REPORTING MANAGERS.

```
SELECT * FROM EMP WHERE EID NOT IN
(SELECT DISTINCT MGR FROM EMP WHERE MGR IS NOT NULL);
```

160. WAQTD THE DETAILS OF MANAGER WHO HAS ATLEAST 4 REPORTINGS.

```
SELECT * FROM EMP WHERE EID IN
(SELECT MGR FROM EMP GROUP BY MGR HAVING COUNT(*)≥4);
```

161. WAQTD THE DEPT NAME OF THE EMPS IF THE EMP IS HAVING REPORTINGS BETWEEN 1 AND 4.
(if you have null values, remove it)

```
SELECT DNAME FROM DEPT WHERE DNO IN
(SELECT DNO FROM EMP WHERE EID IN
(SELECT MGR FROM EMP GROUP BY MGR HAVING COUNT(*)
BETWEEN 1 AND 4));
```

162. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY MORE THAN AVG SALARY OF DNO 111.

```
SELECT * FROM EMP WHERE SAL>
(SELECT AVG(SAL) FROM EMP WHERE DNO=111);
```

==========

CO-RELATED SUBQUERY

==========

Here, not only the outer query depends upon the inner query, but also inner query depends upon the outer query.

163. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY MORE THAN AVG SALARY OF HIS DEPT.

```
SELECT * FROM EMP E1 WHERE SAL>
(SELECT AVG(SAL) FROM EMP WHERE DNO=E1.DNO);
```

164. WAQTD THE DETAILS OF EMP WHO ARE ELDER THAN THEIR REPORTING MANAGERS.

```
SELECT * FROM EMP E1 WHERE DOB<
(SELECT DOB FROM EMP WHERE EID=E1.MGR);
```

165. WAQTD THE DETAILS OF EMP IF THE EMP GENDER AND HIS/HER MANAGER GENDER IS SAME.

```
SELECT * FROM EMP E1 WHERE GENDER IN
(SELECT GENDER FROM EMP WHERE EID=E1.MGR);
```

Drawbacks of Sub query
==========

> We cannot able to get the data from multiple tables at a time.
> To connect multiple tables, many inner queries must be involved.

> Co related sub query is tougher to solve when compared to Joins.

*****
=====
JOINS
=====
*****

Joins are used to retrieve the data from multiple tables simultaneously.

Types of Joins
==============
> Cartesian Joins or Cross Join (Not in use)
> Inner Join **
> Self Join **
> Outer Join **
 → LEFT Outer Join

 → RIGHT Outer Join
 → FULL Outer Join
> Natural Join (Not in use)

Cartesian Join or Cross Join

The records of table 1 will be merged with the records of table 2.

In cross joins,
> The number of columns generated= t1+t2
> The number of rows generated= t1*t2

Syntax:

```
SELECT column_name
FROM table_name_1 CROSS JOIN table_name_2;
```

```
SELECT *
FROM EMP CROSS JOIN DEPT;
```

Drawback
————————

We get more number of error records or invalid records when compared to valid records.

INNER JOIN***
==========

> INNER JOINS are used to obtain only the matched records.

Syntax:
```
SELECT column_name
FROM table_name_1 INNER JOIN table_name_2
ON table_name_1.column_name=table_name_2.column_name;
```

Ex: WAQTD THE EMP AND HIS DEPT DETAILS.

```
SELECT *
FROM EMP INNER JOIN DEPT
ON EMP.DNO=DEPT.DNO;
```

166. WAQTD THE EMP FIRST NAME ALONG WITH HIS DEPT NAME.

```
SELECT FNAME,DNAME
FROM EMP INNER JOIN DEPT
ON EMP.DNO=DEPT.DNO;
```

167. WAQTD THE CUSTOMER FIRST_NAME ALONG WITH HIS LOCATION.

```
SELECT FIRST_NAME,LOCATION
FROM CUSTOMER C1 INNER JOIN LOCATION L1
ON C1.LID=L1.LID;
```

168. WAQTD THE EMP FIRST_NAME, LAST_NAME, JOB ALONG WITH HIS DEPT NAME IF THE EMP IS WORKING IN MARKETING DEPT.

```
SELECT FNAME,LNAME,JOB,DNAME
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO
WHERE DNAME='MARKETING';
```

169. WAQTD THE EMP FIRST NAME, LAST NAME IF THE EMP HAS DELIVERED ANY PRODUCT.

```
SELECT DISTINCT FNAME,LNAME
FROM EMP E1 INNER JOIN ORDERS O1
ON E1.EID=O1.EID;
```

170. WAQTD THE EMP FIRSTNAME,LAST NAME, DEPT NUMBER ALONG WITH DEPT NAME.

```
SELECT E1.FNAME,E1.LNAME,E1.DNO,D1.DNAME
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO;
```

171. WAQTD THE CUSTOMER FIRST_NAME, LAST_NAME,ORDER_ID IF THE CUSTOMER HAS ORDERED ANY PRODUCT.

```
SELECT C1.FIRST_NAME,C1.LAST_NAME,O1.ORDER_ID
FROM CUSTOMER C1 INNER JOIN ORDERS O1
ON C1.CID=O1.CID;
```

172. WAQTD THE EMP FIRST_NAME, HIS DEPT NAME ALONG WITH WORKING LOCATION(DEPT LOC).

```
SELECT E1.FNAME,D1.DNAME,L1.LOCATION
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO INNER JOIN LOCATION L1
ON D1.LID=L1.LID;
```

173. WAQTD THE CUSTOMER FULLNAME, PRODUCT NAME ALONG WITH ORDER ID IF HE HAS ORDERED ANY PRODUCT.

```
SELECT CONCAT(C1.FIRST_NAME,' ',C1.LAST_NAME)
FULL_NAME,P1.PNAME,O1.ORDER_ID
FROM CUSTOMER C1 INNER JOIN ORDERS O1
ON C1.CID=O1.CID INNER JOIN PRODUCT P1
ON O1.PRODUCT_ID=P1.PRODUCT_ID;
```

174. WAQTD THE ORDER ID,CUSTOMER FULL NAME,PRODUCT NAME, PRICE AS BILL, DELIVERY DATE OF THE PRODUCT.

```
SELECT O1.ORDER_ID,CONCAT(C1.FIRST_NAME,'
',C1.LAST_NAME) FULL_NAME,P1.PNAME,P1.PRICE BILL,
O1.DELIVERY_DATE
FROM CUSTOMER C1 INNER JOIN ORDERS O1
ON C1.CID=O1.CID INNER JOIN PRODUCT P1
ON O1.PRODUCT_ID=P1.PRODUCT_ID;
```

175. WAQTD THE ORDER ID,CUSTOMER FULL NAME,PRODUCT NAME, PRICE AS BILL, DELIVERY DATE OF THE PRODUCT ALONG WITH LOCATION, CITY AS ADDRESS.

```
SELECT O1.ORDER_ID,CONCAT(C1.FIRST_NAME,'
',C1.LAST_NAME) FULL_NAME,P1.PNAME,P1.PRICE BILL,
O1.DELIVERY_DATE,CONCAT(L1.LOCATION,', ',L1.CITY)
ADDRESS
FROM CUSTOMER C1 INNER JOIN ORDERS O1
ON C1.CID=O1.CID INNER JOIN PRODUCT P1
ON O1.PRODUCT_ID=P1.PRODUCT_ID INNER JOIN LOCATION L1
ON C1.LID=L1.LID;
```

176. WAQTD THE ORDER ID, CUSTOMER NAME WITH INITIALS, PRODUCT NAME, DELIVERY DATE, ADDRESS, EMP FIRST NAME(who delivered the product) AS DELIVERY_BOY IF HE HAS DELIVERED THE PRODUCT TO MUMBAI CITY.

```
SELECT O1.ORDER_ID,
CONCAT(C1.FIRST_NAME,'.',SUBSTR(C1.LAST_NAME,1,1))CUSTOM
ER_NAME,
P1.PNAME,P1.PRICE BILL,
```

```
O1.DELIVERY_DATE,
CONCAT(L1.LOCATION,', ',L1.CITY) ADDRESS,E1.FNAME
DELIVERY_BOY
FROM CUSTOMER C1 INNER JOIN LOCATION L1
ON C1.LID=L1.LID INNER JOIN ORDERS O1
ON C1.CID=O1.CID INNER JOIN PRODUCT P1
ON O1.PRODUCT_ID=P1.PRODUCT_ID INNER JOIN EMP E1
ON O1.EID=E1.EID
WHERE L1.CITY='MUMBAI';
```

177. WAQTD THE EMP FNAME AND MANAGER FNAME

```
SELECT E1.FNAME emp_name, E2.FNAME manager_name
FROM EMP E1 JOIN EMP E2
ON E1.MGR=E2.EID;
```

178. WAQTD THE EMP FNAME, SALARY, MANAGER FIRST AND HIS SALARY.

```
SELECT E1.FNAME,E1.SAL,E2.FNAME,E2.SAL
FROM EMP E1 JOIN EMP E2
ON E1.MGR=E2.EID;
```

179. WAQTD THE EMP FNAME, EMP DOB ALONG WITH MANAGER FNAME AND HIS DOB IF THE EMP IS ELDER THAN HIS MANAGER.

```
SELECT E1.FNAME,E1.DOB,E2.FNAME,E2.DOB
FROM EMP E1 JOIN EMP E2
ON E1.MGR=E2.EID
WHERE E1.DOB<E2.DOB;
```

180. WAQTD THE EMP FNAME, EMP GENDER ALONG WITH MANAGER FNAME, MANAGER GENDER IF THE EMP AND MANAGERS HAS SAME GENDER.

```
SELECT E1.FNAME,E1.GENDER,E2.FNAME,E2.GENDER
FROM EMP E1 JOIN EMP E2
ON E1.MGR=E2.EID
```

```
WHERE E1.GENDER=E2.GENDER;
```

181. WAQTD THE EMP FNAME, HIS DEPT NAME ALONG WITH HIS MANAGER NAME.

```
SELECT E1.FNAME,D1.DNAME,E2.FNAME
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO JOIN EMP E2
ON E1.MGR=E2.EID;
```

182. WAQTD THE EMP FNAME,LAST NAME HIS DEPT NAME, MANAGER NAME, LAST NAME ALONG WITH HIS DEPT NAME.

```
EMP_DATA                    MANAGER_DATA
_________                  ____________________

HEMA SHETTY, HR          SIDDARTH PATIL, TECHNICAL

SELECT
CONCAT(E1.FNAME,' ',E1.LNAME,', ',D1.DNAME) EMP_DATA,
```

```
CONCAT(E2.FNAME,' ',E2.LNAME,', ',D2.DNAME) MANAGER_DATA
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO JOIN EMP E2
ON E1.MGR=E2.EID INNER JOIN DEPT D2
ON E2.DNO=D2.DNO;
```

183. WAQTD THE EMP FNAME,LAST NAME HIS DEPT NAME AND DEPT LOCATION ALONG WITH MANAGER NAME, MANAGER DEPT AND DEPT LOCATION.

```
EMP_DATA
MANAGER_DATA
_________
_________________________

HEMA SHETTY(HR), VIJAYANAGAR          SIDDARTH
PATIL(TECHNICAL), VIJAYANAGAR

SELECT
CONCAT(E1.FNAME,'(',D1.DNAME,'), ',L1.LOCATION)
```

```
EMP_DATA,
CONCAT(E2.FNAME,'(',D2.DNAME,'), ',L2.LOCATION)
MANAGER_DATA
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO INNER JOIN LOCATION L1
ON D1.LID=L1.LID JOIN EMP E2
ON E1.MGR=E2.EID INNER JOIN DEPT D2
ON E2.DNO=D2.DNO INNER JOIN LOCATION L2
ON D2.LID=L2.LID;
```

184. WAQTD THE EMP FNAME, DEPT NAME ALONG WITH MANAGER FNAME, DEPT NAME IF THE EMP AND MANAGER WORKS IN SAME DEPT.

```
SELECT E1.FNAME,D1.DNAME,E2.FNAME,D2.DNAME
FROM EMP E1 INNER JOIN DEPT D1
ON E1.DNO=D1.DNO JOIN EMP E2
ON E1.MGR=E2.EID INNER JOIN DEPT D2
ON E2.DNO=D2.DNO
```

```
WHERE D1.DNAME=D2.DNAME;
```

185. WAQTD THE EMP FIRST NAME, SAL IF THE EMP IS GETTING SALARY MORE THAN DHARANI'S SALARY.

```
SELECT E1.FNAME,E1.SAL
FROM EMP E1 JOIN EMP E2
ON E1.SAL>E2.SAL
WHERE E2.FNAME='DHARANI';
```

E1 → The emp data who are getting salary more than dharani
E2 → Dharani's data

186. WAQTD THE EMP FIRST NAME, LAST NAME IF THE EMP WAS HIRED AFTER 2 YEARS OF SAMEER'S DOJ.

```
SELECT E1.FNAME,E1.LNAME,E1.DOJ,E2.DOJ SAMEER_DOJ
FROM EMP E1 JOIN EMP E2
```

```
ON E1.DOJ>DATE_ADD(E2.DOJ,INTERVAL 2 YEAR)
WHERE E2.FNAME='SAMEER';
```

187. WAQTD THE EMP FNAME, SAL IF THE EMPS ARE GETTING SAME SALARY.

```
SELECT DISTINCT E1.FNAME,E1.SAL,DENSE_RANK() OVER (ORDER
BY E1.SAL) SAME_SAL_HOLDERS
FROM EMP E1 JOIN EMP E2
ON E1.SAL=E2.SAL
WHERE E1.EID≠E2.EID
```

188. WAQTD THE LOCATIO NAMES WHERE NO CUSTOMERS ARE AVAILABLE.

```
SELECT L1.LOCATION
FROM LOCATION L1 LEFT OUTER JOIN CUSTOMER C1
ON L1.LID=C1.LID
WHERE C1.LID IS NULL;
```

189. WAQTD THE CUSTOMER FULL NAME WHERE THEY HAVE NOT ORDERED ANYTHING SO FAR.

```
SELECT CONCAT(C1.FIRST_NAME,' ',C1.LAST_NAME) FULL_NAME
FROM ORDERS O1 RIGHT OUTER JOIN CUSTOMER C1
ON O1.CID=C1.CID
WHERE O1.CID IS NULL;
```

190. WAQTD THE LOCATION NAMES WITH CITY WHERE NO DEPTS ARE AVAILABLE IN THAT LOCATION.

```
SELECT L1.LOCATION,L1.CITY
FROM LOCATION L1 LEFT OUTER JOIN DEPT D1
ON L1.LID=D1.LID
WHERE D1.LID IS NULL;
```

191. WAQTD THE TOP 3 CUSTOMERS BASED ON THE AMOUNT THEY HAVE SPENT.

```
SELECT C1.CID,CONCAT(C1.FIRST_NAME,' ',C1.LAST_NAME)
FULL_NAME,SUM(P1.PRICE) TOTAL
FROM CUSTOMER C1 INNER JOIN ORDERS O1
ON C1.CID=O1.CID INNER JOIN PRODUCT P1
ON O1.PRODUCT_ID=P1.PRODUCT_ID
GROUP BY C1.CID,CONCAT(C1.FIRST_NAME,' ',C1.LAST_NAME)
ORDER BY TOTAL DESC
LIMIT 3;
```

NATURAL JOINS

> Natural joins are used to obtain the matched records by identifying the common column name present in both the table.
> Here, natural joins depends on the column name for merging two tables.
> The common column will be displayed first in the table and only once.

Syntax:

```
SELECT column_name
FROM table_name_1 NATURAL JOIN table_name_2;
```

```
SELECT *
FROM EMP NATURAL JOIN DEPT;
```

Can we duplicate a table?

- Yes, with a different name.

```
CREATE TABLE table_name AS (SELECT statement);
CREATE TABLE EMP_A AS (SELECT * FROM EMP);
```

Can we copy structure of the table?

- Yes

```
CREATE TABLE table_name AS (SELECT statement with false
condition);
CREATE TABLE EMP_B AS (SELECT * FROM EMP WHERE 1=2);

How to add the data from one table to another table?
```

```
INSERT INTO table_name (SELECT statement);

Ex:
CREATE TABLE EMP_C
(
 ID INT,
 NAME VARCHAR(30)
);

INSERT INTO EMP_C (SELECT EID,FNAME FROM EMP);


VIEWS
```

Views

> Views are the virtual tables.
> They don't require any space to store in database.
> Every resultant tables obtained from DQL are views.
> Views are used to overcome the problems of sub tables.

Syntax:

```
CREATE VIEW view_name AS (SELECT statement);

CREATE VIEW V_DISPATCHERS_DATA AS (SELECT * FROM EMP
WHERE JOB='DISPATCHER');
```

Creation of a user in MySQL

Syntax:

```
CREATE USER 'user_name'@'localhost' IDENTIFIED BY
'password';
```

```
CREATE USER 'john'@'localhost' IDENTIFIED BY 'john';
```

TCL (Transation Control Language)

COMMIT: It is used to save all the transactions(DML Operations) permanently inside the database.
Syntax: COMMIT;

Note: To disable auto commit in MySQL, we use
SET AUTOCOMMIT=0;

ROLLBACK: It is used to roll out all the transactions until the previously used commit statement.
Syntax: ROLLBACK;

> We can also rollback upto a certain savepoint.
Syntax: ROLLBACK TO savepoint_name;

SAVEPOINT: It is used to mark a point in the middle of the transaction.
Syntax: SAVEPOINT savepoint_name;

DCL(Data Control Language)

GRANT: This statement is used to grant the permission to access the data for the other user.
Syntax: GRANT sql_statement ON table_name
TO 'user_name'@'localhost';

REVOKE: This statement is used to take back the permission from the other user.
Syntax: REVOKE sql_statement ON table_name
FROM 'user_name'@'localhost';

Key Attribtes:

Super Key: These are the attributes or the combination

of attributes which is used to make one record as unique.

Candidate Key:> It is the minimal subset of super key.
> Candidate key must be unique.
> All the candidate keys are not primary key but primary key is one among the candidate keys.

Primary Key: It is used to uniquely identify the records from the table.

Foreign Key: It is used to establish connection between multiple tables.

Composite Key: It is the combination of two or more key attributes.

Compound Key: It is the combination of two or more key

attributes where atleast one
among them is a foreign key.

Alternate Key: The candidate keys except the primary keys are known as Alternate Keys.