SQL Commands

→ DDL [Data Definition Language]

* CREATE
* ALTER
* DROP
* TRUNCATE

→ DML [Data Manipulation Language]

* INSERT
* UPDATE
* DELETE

→ TCL [Transaction Control Language]

* COMMIT
* ROLLBACK
* SAVEPOINT

→ DCL [Data Control Language]

* GRANT
* REVOKE

→ DQL [Data Query Language]

* SELECT
* JOINS

CREATION OF DATABASE

To create a database,
CREATE DATABASE database-name;

To check the databases available in MySQL,
SHOW DATABASES;

To access a particular DB,
USE database-name;

To display the tables available in the current database,
SHOW TABLES;

DDL - [Data Definition Language]

CREATION OF TABLES

CREATE : This command is used to create database and its objects such as tables, view, user, trigger, procedure etc.

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
);

Note: To describe (To display Table structure)
table,
DESC table_name;

CREATE TABLE ACCOUNTS
(
ACCNO BIGINT PRIMARY KEY,
NAME VARCHAR(15) NOT NULL,
```

```
PHONE BIGINT UNIQUE NOT NULL
CHECK(LENGTH(PHONE)=10),
MAIL VARCHAR(20) UNIQUE NOT NULL
);

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

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 table structure.

> To add a column to existing table

```
ALTER TABLE table_name
ADD column_name DATATYPE CONSTRAINT NULL/NOT
NULL;
```

```
ALTER TABLE ACCOUNTS
ADD ADDRESS TEXT NOT NULL;

ALTER TABLE BRANCH
ADD IFSC_CODE VARCHAR(15) UNIQUE NOT NULL;

ALTER TABLE ACCOUNTS
ADD LNAME VARCHAR(10) NOT NULL AFTER NAME;
```

> To delete a column

```
ALTER TABLE table-name
DROP column-name;
```

```
ALTER TABLE ACCOUNTS
DROP LNAME;

ALTER TABLE LOCATION
DROP AREA;
```

> To change the column-name

```
ALTER TABLE table-name
CHANGE old-column-name new-column-name
existing-datatype NULL/NOT NULL;

ALTER TABLE ACCOUNTS
CHANGE MAIL EMAIL_ID VARCHAR(30) NOT NULL;
```

> To change the table-name

```
ALTER TABLE table-name
RENAME new-table-name;

ALTER TABLE LOCATION
RENAME LOC;
```

> To modify the datatype of a column

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

ALTER TABLE ACCOUNTS
MODIFY EMAIL_ID CHAR(30) NOT NULL;
```

```
ALTER TABLE BRANCH
MODIFY PINCODE BIGINT NOT NULL;
```

> To modify NULL/NOT NULL

```
ALTER TABLE table_name
MODIFY column_name EXISTING-DATATYPE NULL/NOT
NULL;

ALTER TABLE ACCOUNTS
MODIFY NAME VARCHAR(30) NULL;
```

> To add constraints to the existing columns,

```
ALTER TABLE table_name
ADD CONSTRAINT PRIMARY KEY(column-name);
ADD CONSTRAINT UNIQUE(column-name);
ADD CONSTRAINT CHECK(condition);
ADD CONSTRAINT FOREIGN KEY(column-name)
REFERENCES parent-table-name(column-name);

ALTER TABLE ACCOUNTS
ADD CONSTRAINT FOREIGN KEY(BRANCH_ID)
REFERENCES BRANCH(B_ID);
```

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

Syntax:
DROP TABLE table-name;

DROP TABLE ACCOUNTS;

==========

DML [Data Manipulation Language]

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

Syntax1:
INSERT INTO table-name VALUES

```
(v1,v2,..vn),(v1,v2,..,vn),......

Syntax2:
INSERT INTO table-name (col1,col2,..,coln)
VALUES (v1,v2,..,vn),(v1,v2,..,vn),.....

Syntax3:
INSERT INTO table-name (SELECT command);

INSERT INTO PRODUCT VALUES (1,'Dairy
Milk',200);
```

UPDATE : This command is used to modify the

records present in the table.

Syntax:

```
UPDATE table-name
SET col=v1[,col2=v2,..,coln=vn]
WHERE condition;
```

```
UPDATE PRODUCT
SET PRICE=50
WHERE PID=2;
```

```
UPDATE PRODUCT
SET PNAME='Munch',PRICE=10;
```

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

Syntax:

```
DELETE FROM table-name
WHERE condition;

-- To delete all the records,
DELETE FROM PRODUCT;
```

*** Difference between TRUNCATE, DROP, DELETE

TRUNCATE : This command is used to erase all the records permanently from the table. But the table structure remains same.

Syntax:
TRUNCATE TABLE table-name;

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

Syntax:
DROP TABLE table-name;

DELETE : This command is used to delete a particular record from the table

Syntax:
DELETE FROM table-name
[WHERE filter-condition];

==========

DQL [Data Query Language]

Projection: The process of retrieving the data from the table by using column-names.

Syntax:
SELECT column-name
FROM table-name;

FROM : This clause is used to put the table under execution.
> It is the first executable clause.
> It takes table-name as a argument.
> It searches for the given table and put it under execution.

SELECT : This clause is used to display the records present in the table.
> It takes column name as a argument.

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

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

2. WAQTD THE EMP FIRST NAME, JOB, SAL FROM EMP TABLE.

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

3. WAQTD THE EMP FIRST NAME, JOB, SAL, DEPT NO FROM EMP TABLE.

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

4. WAQTD THE DETAILS OF EMPS.

```
SELECT *
FROM EMP;
```

5. WAQTD THE EMP FNAME, SALARY, ANNUAL SALARY FROM EMP TABLE.

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

6. WAQTD THE FNAME, SALARY WITH HALF TERM SALARY.

```
SELECT FNAME,SAL,SAL*6
FROM EMP;
```

7. WAQTD THE EMP FNAME, SALARY, ANNUAL SALARY WITH 50000 BONUS.

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

8. WAQTD THE EMP FNAME, SALARY WITH 1000 RS INCENTIVE.

```
SELECT FNAME,SAL+1000
FROM EMP;
```

9. WAQTD THE EMP FNAME, SALARY WITH 5% INCENTIVE.

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

10. WAQTD THE EMP FNAME, SALARY WITH 10% INCENTIVE.

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

11. WAQTD THE EMP FNAME, SAL WITH 3% DEDUCTION.

```
SELECT FNAME,SAL-(SAL*0.03)
FROM EMP;
```

12. WAQTD THE EMP FNAME, SAL WITH 5% INCENTIVE, COMM WITH 3% DEDUCTION.

```
SELECT FNAME,SAL+(SAL*0.05),COMM-(COMM*0.03)
FROM EMP;

or

SELECT FNAME,SAL+(SAL*0.05) "5% SAL
INCREMENT",
COMM-(COMM*0.03) COMM_DEDUCTION
FROM EMP;
```

ALIAS : It is an alternative name given to the columns or the tables.

> ALIAS name can be passed by using AS keyword or "double quotes".
> With or without using AS keyword, we can able to pass alias name.
> If any special characters or spaces to be provided, we can make use of "double quotes".

Syntax:

```
SELECT column-1 AS alias_1,
column-2 alias-2,
column-3 "alias #3"
FROM table-name;
```

13. WAQTD THE DETAILS ALONG WITH ANNUAL SALARY FROM EMP TABLE.

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

14. WAQTD THE DIFFERENT JOB ROLES AVAILABLE IN EMP TABLE.

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

DISTINCT : Distinct is used to avoid the

duplicate records from the resultant table.
> Either * or distinct must be the very first argument.
> Whenever we pass multiple columns inside DISTINCT, it checks for the combination.

Syntax:
```
SELECT DISTINCT column-name
FROM table-name;
```

---

15. WAQTD THE DIFFERENT DEPTS AVAILABLE IN EMP TABLE.

```
SELECT DISTINCT DNO
FROM EMP;
```

SELECTIONS

The process of retrieving the data from the table by using column names and row data is known as Selections.

Syntax:

```
SELECT column-name
FROM table-name
```

```
WHERE filter-condition;
```

WHERE

WHERE clause is used to filter the records from the table.

Characteristics of WHERE clause

> It executes row by row.
> It executes after the FROM clause
> WHERE clause follows True or False condition.

> We can pass multiple conditions inside the WHERE clause.
> We cannot pass alias name inside WHERE clause.

---

16. WAQTD THE DETAILS OF EMP IF THE EMP FNAME IS AMAN.

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

17. WAQTD THE EMP FNAME, LNAME, JOB IF THE EMP

IS WORKING AS SALESMAN.

```
SELECT FNAME,LNAME,JOB
FROM EMP
WHERE JOB='SALESMAN';
```

18. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY MORE THAN 50000.

19. WAQTD THE EMP FNAME, DOB IF THE EMP WAS BORN AFTER 1995.

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

20. WAQTD THE EMP FNAME, JOB, DOJ IF THE EMP WAS JOINED BEFORE 2020.

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

21. WAQTD THE EMP FNAME, JOB, DNO IF THE EMP IS WORKING IN DEPT 112.

```
SELECT FNAME,JOB,DNO
FROM EMP
WHERE DNO=112;
```

Operators in MySQL

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

---

22. WAQTD THE DETAILS OF EMP IF THE EMP IS

GETTING SALARY MORE THAN 32000 BUT LESS THAN 45000.

```
SELECT *
FROM EMP
WHERE SAL>32000 AND SAL<45000;
```

23. WAQTD THE EMP FNAME, JOB IF THE EMP IS WORKING AS SALESMAN OR MANAGER.

```
SELECT FNAME,JOB
FROM EMP
WHERE JOB='SALESMAN' OR JOB='MANAGER';
```

24. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN AND GETTING SALARY MORE THAN 35000.

```
SELECT *
FROM EMP
WHERE JOB='SALESMAN' AND SAL>35000;
```

25. WAQTD THE EMP FNAME, SAL, DNO IF THE EMP IS GETTING SALARY LESS THAN 50000 BUT WORKING IN DEPT 113 OR 112.

```
SELECT FNAME,SAL,DNO
FROM EMP
WHERE SAL<50000 AND DNO=113 OR DNO=112;
```

26. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN, MANAGER OR DISPATCHER IN DEPT 110,111, 113.

```
SELECT *
FROM EMP
WHERE (JOB='SALESMAN' OR JOB='MANAGER' OR
JOB='DISPATCHER') AND
(DNO=110 OR DNO=111 OR DNO=113);
```

27. WAQTD THE DETAILS OF EMP IF THE EMP IS NOT CEO.

```
SELECT *
```

```
FROM EMP
WHERE JOB≠'CEO';
```

28. WAQTD THE EMP FNAME, JOB ROLE IF THE EMP IS NOT WORKING AS SALESMAN, MANAGER.

```
SELECT FNAME,JOB
FROM EMP
WHERE JOB≠'SALESMAN' AND JOB≠'MANAGER';
```

29. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN OR MANAGER OR DISPATCHER OR DEVELOPER.

```
SELECT *
```

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

or

```
SELECT *
FROM EMP
WHERE JOB IN
('SALESMAN','MANAGER','DISPATCHER','DEVELOPER'
);
```

IN/NOT IN Operator

IN operator is a multivalued operator which accepts single value at LHS and multiple values at RHS.

Syntax:

```
        LHS                     RHS
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.

30. WAQTD THE EMP FNAME, LNAME, DEPT NO IF THE EMPS ARE WORKING IN 110, 111 OR 113 DEPTS.

```
SELECT FNAME,LNAME,DNO
FROM EMP
WHERE DNO IN (110,111,113);
```

31. WAQTD THE EMP FNAME, JOB IF THE EMP IS NOT WORKING AS SALESMAN, MANAGER OR DEVELOPER.

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

32. WAQTD THE EMP FNAME, LNAME, SALARY IF THE EMP IS GETTING SALARY MORE THAN OR EQUAL TO 32000 AND LESS THAN OR EQUAL TO 45000.

```
SELECT FNAME,LNAME,SAL
FROM EMP
WHERE SAL≥32000 AND SAL≤45000;
```

or

```
SELECT FNAME,LNAME,SAL
FROM EMP
WHERE SAL BETWEEN 32000 AND 45000;
```

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;

≥

≤

Note: BETWEEN operator includes the range values

33. WAQTD THE DETAILS OF EMP WHO WERE BORN IN THE YEAR 1995.

```
SELECT *
FROM EMP
WHERE DOB BETWEEN '1995-01-01' AND
'1995-12-31';
```

34. WAQTD THE EMP FNAME, SAL IF THE EMP IS GETTING SALARY MORE THAN 30000 AND LESS THAN 50000.

```
SELECT FNAME,SAL
FROM EMP
```

```
WHERE SAL>30000 AND SAL<50000;
```

35. WAQTD THE EMP FNAME, SAL, JOB, DOJ IF THE EMP WAS NOT JOINED IN 2021.

```
SELECT FNAME,SAL,JOB,DOJ
FROM EMP
WHERE DOJ NOT BETWEEN '2021-01-01' AND
'2021-12-31';
```

36. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN, MANAGER, DISPATCHER, ACCOUNTANT BUT NOT IN DEPT 113 OR 114 AND GETTING SALARY IN THE RANGE OF 30000 AND 55000.

```
SELECT *
FROM EMP
WHERE JOB IN
('SALESMAN','MANAGER','DISPATCHER','ACCOUNTANT
') AND
DNO NOT IN (113,114) AND SAL BETWEEN 30000 AND
55000;
```

IS Operator

IS operator is used to check whether the given condition is NULL or NOT NULL.

Syntax:

```
column-name IS NULL/NOT NULL;
```

37. WAQTD THE DETAILS OF EMP IF THE EMP IS NOT GETTING ANY COMMISSION.

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

38. WAQTD THE EMP FNAME, CID IF THE EMP IS ALSO A CUSTOMER OF THE SAME COMPANY.

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

39. WAQTD THE DETAILS OF EMP WHO DOESNOT HAVE REPORTING MANAGER.

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

40. WAQTD THE EMP FNAME, LNAME, JOB, DNO, SAL,MGR IF THE EMP IS WORKING IN DEPT 110, 111 AND 113 BUT NOT AS ACCOUNTANT OR CEO OR HR BUT GETTING SALARY IN THE RANGE OF 30000 TO 55000

BUT NOT HIRED IN 2022 AND HAVING A REPORTING MANAGER.

```
SELECT FNAME,LNAME,JOB,DNO,SAL,MGR
FROM EMP
WHERE DNO IN (110,111,113) AND JOB NOT IN
('ACCOUNTANT','CEO','HR')
AND SAL BETWEEN 30000 AND 55000 AND
DOJ NOT BETWEEN '2022-01-01' AND '2022-12-31'
AND
MGR IS NOT NULL;
```

LIKE Operator

LIKE operator is used for matching the patterns.

Syntax:

```
column-name LIKE/NOT LIKE 'pattern-to-match';
```

There are 2 wildcards in LIKE Operators:
Percentile[%]: Takes any character, any number of time.
Underscore[_]: Takes any character, but only one character at a time.

41. WAQTD THE EMP FNAME IF THE EMP FNAME STARTS WITH S.

```
SELECT FNAME
FROM EMP
WHERE FNAME LIKE 'S%';
```

42. WAQTD THE DETAILS OF EMP IF THE FNAME ENDS WITH A.

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

43. WAQTD THE EMP FNAME, JOB IF THE JOB ROLES CONTAINS MAN IN IT.

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

44. WAQTD THE DETAILS OF EMP IF THE EMP FNAME LAST 2ND CHARACTER IS A.

```
SELECT *
FROM EMP
WHERE FNAME LIKE '%A_';
```

45. WAQTD THE EMP FNAME, JOB, DOJ IF THE EMP

WAS HIRED IN THE YEAR 2021 BY USING LIKE OPERATOR.

```
SELECT FNAME,JOB,DOJ
FROM EMP
WHERE DOJ LIKE '2021%';
```

'YYYY-MM-DD'

46. WAQTD THE DETAILS OF EMP IF THE EMP FNAME STARTS WTIH S AND LNAME ENDS WITH L.

```
SELECT *
FROM EMP
WHERE FNAME LIKE 'S%' AND LNAME LIKE '%L';
```

47. WAQTD THE EMP FNAME IF THE EMP FNAME STARTS WITH S OR A.

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

48. WAQTD THE DETAILS OF EMP IF THE EMP LNAME ENDS WITH I OR Y.

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

49. WAQTD THE EMP FNAME, DOB IF THE EMP WAS BORN IN THE YEAR 1995.

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

50. WAQTD THE DETAILS OF EMP IF THE EMP FNAME STARTS WITH S OR A AND FNAME ENDS WITH I OR T.

```
SELECT *
FROM EMP
WHERE (FNAME LIKE 'S%' OR FNAME LIKE 'A%') AND
      (FNAME LIKE '%I' OR FNAME LIKE '%T');
```

51. WAQTD THE DETAILS OF EMP WHOSE FNAME DOESNOT START WITH S.

```
SELECT *
FROM EMP
WHERE FNAME NOT LIKE 'S%';
```

52. WAQTD THE EMP FNAME IF THE FNAME DOESNOT START WITH S OR A.

```
SELECT FNAME
FROM EMP
WHERE FNAME NOT LIKE 'S%' AND FNAME NOT LIKE
'A%';
```

53. WAQTD THE DETAILS OF EMP IF THE EMP 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%';
```

54. WAQTD THE EMP FNAME, LNAME IF THE EMP FNAME DOESNOT START WITH VOWELS BUT LNAME END WITH VOWELS.

```
SELECT FNAME,LNAME
FROM EMP
WHERE (FNAME NOT LIKE 'A%' AND
       FNAME NOT LIKE 'E%' AND
       FNAME NOT LIKE 'I%' AND
       FNAME NOT LIKE 'O%' AND
       FNAME NOT LIKE 'U%') AND
       (LNAME LIKE '%A' OR
        LNAME LIKE '%E' OR
        LNAME LIKE '%I' OR
        LNAME LIKE '%O' OR
        LNAME LIKE '%U');

DESC STUD;
```

55. WAQTD THE FNAME OF THE EMP IF THE EMP FNAME CONTAINS ATLEAST 1 E IN IT.

```
SELECT FNAME
FROM EMP
WHERE FNAME LIKE '%E%';
```

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

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

57. WAQTD THE NAME OF THE STUD IF THE NAME

CONTAINS ATLEAST 1 _ IN IT.

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

58. WAQTD THE FNAME OF THE EMP IF THE EMP FNAME CONTAIN ATLEAST 2 A IN IT.

```
SELECT FNAME
FROM EMP
WHERE FNAME LIKE '%A%A%';
```

59. WAQTD THE NAME OF THE STUD IF THE NAME CONTAINS ATLEAST 2 % IN IT.

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

60. WAQTD THE NAME OF THE STUD IF THE NAME CONTAINS ATLEAST 2 _ IN IT.

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

==========

==========

FUNCTIONS

Functions are nothing but the block of code to perform specific task.

Types of Functions

1. Aggregate Functions
2. Character Functions
3. Number Functions
4. Date Functions ***
5. Window Functions

Aggregate Functions

These functions take multiple values as input and generate a single output.

Types of Aggregate Functions

1. MAX(): This function is used to obtain the max value from the given column.
Syntax: MAX(column-name);

2. MIN(): This function is used to obtain the min value from the given column.
Syntax: MIN(column-name);

3. SUM(): This function is used to obtain the

total value from the given column.
Syntax: SUM(column-name);

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

5. COUNT(): This function is used to obtain the number of values 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 display any normal columns along with aggregate functions.
> We cannot nest aggregate functions.
> We cannot pass multiple columns inside a single aggregate functions.
> We cannot pass aggregate functions inside the WHERE clause.
> Aggregate functions ignore the NULL values.
> GROUP BY → We can pass group-by-expression along with aggregate functions.

61. WAQTD THE MAX SALARY AND MIN SALARY GIVEN IN EMP TABLE.

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

62. WAQTD THE AVG COMM GIVEN IN THE EMP TABLE.

```
SELECT AVG(COMM)
FROM EMP;
```

63. WAQTD THE MAX SALARY AND TOTAL SALARY GIVEN TO ALL THE SALESMAN.

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

64. WAQTD THE MIN SALARY, AVG SALARY GIVEN TO THE EMPS WHOSE FNAME STARTS WITH S OR A.

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

65. WAQTD THE TOTAL SALARY GIVEN TO ALL THE EMP OF DNO 112.

```
SELECT SUM(SAL)
```

```
FROM EMP
WHERE DNO=112;
```

66. WAQTD THE NUMBER OF EMPS IN THE COMPANY.

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

67. WAQTD THE NUMBER OF EMPS WHO ARE WORKING AS SALESMAN.

```
SELECT COUNT(*)
FROM EMP
WHERE JOB='SALESMAN';
```

68. WAQTD THE TOTAL NUMBER OF EMPS WORKING IN DEPT 113.

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

69. WAQTD THE TOTAL NUMBER OF DIFFERENT JOB RLES AVAILABLE IN THE COMPANY.

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

70. WAQTD THE TOTAL NUMBER OF EMPS WORKING IN DEPT 112, 113, 114.

```
SELECT COUNT(*)
FROM EMP
WHERE DNO IN (112,113,114);
```

GROUP BY

Group By clause is used to group the records.

Syntax:

```
SELECT aggregate-function/group-by-expression
FROM table-name
```

```
[WHERE filter-condition]
GROUP BY column-name;
```

Characteristics of GROUP BY clause

> GROUP BY clause executes row-by-row.
> It executes after the FROM clause if there is no WHERE clause.
> GROUP BY clause converts all the row records into group records.
> After the execution of GROUP BY clause, all the other clauses execute group-by-group.
> We can pass multiple columns inside GROUYP BY clause.
> We can display only aggregate functions and

group by expression along with GROUP BY clause.

Note: The column which we pass as a argument to GROUP BY clause is known as Group-By-Expression.

---

71. WAQTD THE TOTAL NUMBER OF EMPS IN EACH DEPT.

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

72. WAQTD THE MAXIMUM SALARY, MINIMUM SAALRY GIVEN TO ALL THE EMPS IN EACH DEPT.

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

73. WAQTD THE NUMBER OF EMPS WORKING IN EVERY JOB ROLE.

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

74. WAQTD THE NUMBER OF EMPS WORKING IN EVERY DEPT IF THE EMP JOB ROLE IS SALESMAN, MANAGER OR DISPATCHER.

```
SELECT COUNT(*),DNO
FROM EMP
WHERE JOB IN
('SALESMAN','MANAGER','DISPATCHER')
GROUP BY DNO;
```

75. WAQTD THE ELDEST EMP DOB IN THE COMPANY.

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

76. WAQTD THE RECENT JOINED EMP'S DOJ FROM EMP TABLE.

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

77. WAQTD THE NUMBER OF EMPS WORKING IN EACH DEPT IF THE EMP GETTING SALARY MORE THAN OR EQUAL TO 30000 AND SAL LESS THAN OR EQUAL TO 100000.

```
SELECT COUNT(*),DNO
FROM EMP
WHERE SAL≥30000 AND SAL≤100000
GROUP BY DNO;
```

HAVING clause

HAVING clause is used to filter the group functions.

Syntax:

```
SELECT aggregate-function/group-by-expression
FROM table-name
[WHERE filter-condition]
GROUP BY column-name
HAVING filter-group-function;
```

Characteristics of HAVING clause

> It execute group-by-group.
> HAVING Clause executes after the GROUP BY clause.
> We can pass aggregate functions as argument for HAVING clause.
> It takes aggregate function conditions as argument.
> We can pass multiple conditions inside HAVING clause.

78. WAQTD THE NUMBER OF EMPS WORKING IN EACH DEPT IF THE EMP GETTING SALARY MORE THAN OR EQUAL TO 32000 AND MAX SALARY LESS THAN OR EQUAL TO 150000.

```
SELECT COUNT(*),DNO
FROM EMP
WHERE SAL≥32000
GROUP BY DNO
HAVING MAX(SAL)≤150000;
```

79. WAQTD THE MAX SALARY, MIN SALARY GIVEN TO EVERY EMP IN THE DEPT IF THE DEPT AVG SALARY IS MORE THAN 34000.

```
SELECT MAX(SAL),MIN(SAL),DNO
FROM EMP
GROUP BY DNO
HAVING AVG(SAL)>34000;
```

80. WAQTD THE TOTAL SALARY, AVG SALARY GIVEN IN EACH DEPT IF THE EMP MAX SALARY MUST NOT EXCEED 120000.

```
SELECT SUM(SAL),AVG(SAL),DNO
FROM EMP
GROUP BY DNO
HAVING MAX(SAL)≤120000;
```

81. WAQTD THE NUMBER OF EMPS IN THE DEPT IF

DEPT MAX SALARY IS LESS THAN 150000 BUT MIN SALARY MORE THAN 30000.

```
SELECT COUNT(*),DNO
FROM EMP
GROUP BY DNO
HAVING MIN(SAL)>30000 AND MAX(SAL)<150000;
```

82. WAQTD THE MAX SAL, MIN SALARY, TOTAL SALARY, AVG SALARY GIVEN IN DEVERY DEPT IF THE DEPT HAS ATLEAST 3 EMPS WORKING IN IT.

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

```
HAVING COUNT(*)≥3;
```

83. WAQTD THE TOTAL SALARY OF DEPT IF THE DEPT CONTAINS ATLEAST 2 EMPS AND MAX 4 EMPS WORKING IN DEPT.

```
SELECT SUM(SAL),DNO
FROM EMP
GROUP BY DNO
HAVING COUNT(*) BETWEEN 2 AND 4;
```

84. WAQTD THE NUMBER OF EMPS GETTING SAME SALARY.

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

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

85. WAQTD THE NUMBER OF EMPS GOT HIRED ON SAME DAY.

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

86. WAQTD THE NUMBER OF EMPS WORKING IN SAME DEPT AND GETTING SAME SALARY.

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

ORDER BY

Order By clause is used to arrange the records either in ascending or descending order.

Characteristics of ORDER BY clause

> By default, the table will be arrange

according to Primary key in ascending order.
> Order By clause is the last executable clause as it executes after the SELECT clause.
> Order By clause takes either ASC or DESC as argument.
> BY default, ORDER BY clause takes ASC as argument.
> We can pass multiple columns inside ORDER BY clause.

Syntax

```
SELECT column-name
FROM table-name
ORDER BY column-name ASC/DESC;
```

87. WAQTD THE DETAILS OF EMP ACCORDING TO SALARY FROM MINIMUM TO MAXIMUM.

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

88. WAQTD THE FNAME, LNAME OF EMP IN ALPHABETICAL ORDER.

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

```
ORDER BY FNAME;
```

89. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN, MANAGER OR DISPATCHER. ARRANGE THE RECORDS BASED ON THEIR SALARY IN DESC ORDER.
SECONDARY SORT BY USING DEPTNO IN ASC ORDER.

```
SELECT *
FROM EMP
WHERE JOB IN
('SALESMAN','MANAGER','DISPATCHER')
ORDER BY SAL DESC, DNO ASC;
```

90. WAQTD THE NUMBER OF EMPS WORKING IN EVERY

DEPT IF THE EMP IS WORKING IN DEPT 110,111,112 OR 113 AND MAX SALARY OF DEPT MST BE MORE THAN 70000. ARRANGE THE RECORDS BASED ON NUMBER OF EMPS FROM MAX TO MIN.

```
SELECT COUNT(*),DNO
FROM EMP
WHERE DNO IN (110,111,112,113)
GROUP BY DNO
HAVING MAX(SAL)>70000
ORDER BY COUNT(*) DESC;
```

LIMITS AND OFFSET

LIMIT : It is used to return the specified number of records from the table.

OFFSET : It is used to ignore the specified number of records from the table.

---

91. WAQTD THE TOP 3 RECORDS FROM EMP TABLE.

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

92. WAQTD THE TOP 5 RECORDS FROM EMP TABLE.

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

93. WAQTD THE 1ST RECORD FROM EMP TABLE.

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

94. WAQTD THE TOP 2 RECORDS FROM EMP TABLE.

```
SELECT *
```

```
FROM EMP
LIMIT 2;
```

95. WAQTD THE TOP 2ND RECORD FROM EMP TABLE.

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

96. WAQTD THE TOP 5TH RECORD FROM EMP TABLE.

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

97. WAQTD TOP 10TH AND 11TH RECORD FORM EMP TABLE.

```
SELECT *
FROM EMP
LIMIT 2 OFFSET 9;
```

98. WAQTD THE TOP 7TH RECORD FROM EMP TABLE.

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

99. WAQTD THE LAST RECORD OF EMP TABLE.

```
SELECT *
FROM EMP
ORDER BY EID DESC
LIMIT 1;
```

100. WAQTD THE LAST 5 RECORDS FROM EMP TABLE.

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

101. WAQTD THE LAST 3 RECORDS FROM EMP TABLE.

```
SELECT *
```

```
FROM EMP
ORDER BY EID DESC
LIMIT 3;
```

102. WAQTD THE LAST 4TH RECORD FROM EMP TABLE.

```
SELECT *
FROM EMP
ORDER BY EID DESC
LIMIT 1 OFFSET 3;
```

103. WAQTD THE TOP 3 MINIMUM SALARIES FROM EMP TABLE.

```
SELECT DISTINCT SAL
```

```
FROM EMP
ORDER BY SAL ASC
LIMIT 3;
```

104. WAQTD THE TOP 5 MAXIMUM SALARIES FROM EMP TABLE.

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

105. WAQTD THE 5TH MAX SALARY FROM EMP TABLE.

```
SELECT DISTINCT SAL
```

```
FROM EMP
ORDER BY SAL DESC
LIMIT 1 OFFSET 4;
```

106. WAQTD THE 3RD MIN SALARY FROM EMP TABLE.

```
SELECT DISTINCT SAL
FROM EMP
ORDER BY SAL ASC
LIMIT 1 OFFSET 2;
```

107. WAQTD THE DOB OF 2ND ELDEST EMP.

```
SELECT DISTINCT DOB
FROM EMP
```

```
ORDER BY DOB ASC
LIMIT 1 OFFSET 1;
```

Character Functions

LENGTH(): This function is used to obtain the length of the given string.

UPPER(): This function is used to convert the lower cases into upper cases.

LOWER():This function is used to convert the upper cases into lower cases.

REVERSE(): This function is used to reverse the given string.

CONCAT(): This function is used to combine the given multiple string.

---

108. WAQTD THE FULLNAME OF THE EMPS.

FULLNAME
Siddarth Patil

```
SELECT CONCAT(FNAME,' ',LNAME) FULLNAME
FROM EMP;
```

109. WAQTD THE FULLNAME AND THE LENGTH OF THE FNAME AND LAST NAME OF THE EMP.

```
SELECT CONCAT(FNAME,' ',LNAME)
FULLNAME,LENGTH(FNAME),LENGTH(LNAME)
FROM EMP;
```

SUBSTR(): This function is used to obtain the part of a given string.

Syntax:

```
SUBSTR('string-value',Position[,Length]);

SELECT SUBSTR('BENGALURU',-4,-3);
```

B	E	N	G	A	L	U	R	U
1	2	3	4	5	6	7	8	9
-9	-8	-7	-6	-5	-4	-3	-2	-1

110. WAQTD THE FIRST 3 CHARACTER OF EMP FNAME.

```
SELECT SUBSTR(FNAME,1,3)
FROM EMP;
```

111. WAQTD THE LAST 2 CHARACTER OF EMP LNAME.

```
SELECT SUBSTR(LNAME,-2)
FROM EMP;
```

112. WAQTD THE DETAILS OF EMP IF THE EMP FNAME STARTS WITH S WITHOUT USING LIKE OPERATOR.

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

113. WAQTD THE DETAILS OF EMP IF THE EMP FNAME ENDS WITH A WITHOUT USING LIKE OPERATOR.

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

114. WAQTD THE DETAILS OF EMP IF THE FNAME STARTS WITH VOWELS.

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

115. WAQTD THE DETAILS OF EMP IF THE EMP LNAME ENDS WITH VOWELS.

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

116. WAQTD THE EMP FULLNAME IN THE BELOW FORMAT.

```
   FULLNAME
  ____________

  Siddarth.P
```

```
SELECT CONCAT(FNAME,'.',SUBSTR(LNAME,1,1))
```

```
FULLNAME
FROM EMP;
```

117.WAQTD THE DETAILS OF EMP IF THE EMP FNAME MUST NOT START WITH VOWLES BUT LNAME MUST END WITH VOWELS WITHOUT LIKE OPERATOR.

```
SELECT *
FROM EMP
WHERE SUBSTR(FNAME,1,1) NOT IN
('A','E','I','O','U') AND
     SUBSTR(LNAME,-1) IN
('A','E','I','O','U');
```

118. WAQTD THE FIRST HALF OF FNAME.

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

119. WAQTD THE SECOND HALF OF THE FNAME.

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

120. WAQTD THE FIRST HALF OF THE FNAME IN LOWER CASE AND SECOND HALF IN UPPER REVERSE CASE.

FNAME	firstEMAN

```
Siddarth    siddHTRA

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

REPLACE(): This function is used to substitute a sub string with a new string.

Syntax:

```
REPLACE('string','sub-string','new-string');
```

```
Ex1: SELECT REPLACE('Pentagon','n','s');

Ex2: SELECT REPLACE('Pentagon','a','@');

Ex3: SELECT REPLACE('Pentagon','n','');
```

---

121. WAQTD THE BELOW REQ.

```
Input-1 : MABROOQ
Input-2 : O
Output  : 2
```

```
Input-1 : MALAYALAM
Input-2 : A
Output  : 4

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

122. WAQTD THE FNAME,NUMBER OF 'A' PRESENT IN
THE GIVEN FNAMES OF EMP.

SELECT
FNAME,LENGTH(FNAME)-LENGTH(REPLACE(LOWER(FNAME
),'a','')) COUNT
```

```
FROM EMP;
```

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);

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 argument.
Ex: SELECT POW(3,0);

DATE FUNCTIONS

1. SYSDATE(): This function is used to obtain the current date and time.

2. CURDATE(): This function is used to obtain the current date.

3. CURTIME(): This function is used to obtain

the current time.

4. DATEDIFF(): This function is used to obtain the differences between the given two dates.

5. DATE_ADD(): This function is used to add the date values based on Interval.

6. DATE_SUB(): This function is used to subtract the date values based on Interval.

7. DATE_FORMAT(): This function is used to convert the date values into individual date characters.

Syntax:
DATE_FORMAT(date-value,'format-model');

8. YEAR(): This function is used to obtain only year from the date value.

9. MONTH(): This function is used to obtain only month of the date value.

==========

==========

123. WAQTD THE AVG SALARY GIVEN TO EACH DEPT. ROUND OFF THE VALUE TO NEXT INTEGER VALUE.

SELECT CEIL(AVG(SAL)),DNO

```
FROM EMP
GROUP BY DNO;
```

124. WAQTD THE DETAILS OF EMP WHO HAS EVEN ID.

```
SELECT *
FROM EMP
WHERE MOD(EID,2)=0;
```

125. WAQTD THE EMP FNAME, DOB IN US FORMAT.
- mm/dd/yyyy

```
SELECT FNAME,DATE_FORMAT(DOB,'%m/%d/%Y') DOB
FROM EMP;
```

126. WAQTD THE FNAME, YEAR OF JOINING FROM EMP TABLE.

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

127. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS A SALESMAN .

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

128. WAQTD THE DETAILS OF EMP WHO WAS HIRED IN

THE YEAR 2021 USING DATE FUNCTIONS.

```
SELECT *
FROM EMP
WHERE YEAR(DOJ)=2021;
```

129. WAQTD THE DETAILS OF EMP WHO WEERE HIRED ON SATURDAY OR SUNDAY.

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

130. WAQTD THE EMP FNAME, DOJ IF THE EMP WAS HIRED IN 2016, 2017 OR 2022.

```
SELECT FNAME,DOJ
FROM EMP
WHERE YEAR(DOJ)IN(2016,2017,2022);
```

131. WAQTD THE DETAILS OF EMP WHO WERE BORN DURING THE EVEN YEARS.

```
SELECT *
FROM EMP
WHERE MOD(YEAR(DOB),2)=0;
```

132. WAQTD THE DETAILS OF EMP WHO JOINED THE COMPANY DURING THE LEAP YEAR.

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

133. WAQTD THE NUMBER OF EMPS JOINED THE COMPANY EVERY YEAR.

```
SELECT YEAR(DOJ) JOINING_YEAR,COUNT(*) COUNT
FROM EMP
GROUP BY YEAR(DOJ);
```

134. WAQTD THE EMP FNAME, YEAR OF EXPERIENCE OF THE EMP.

```
SELECT FNAME,YEAR(CURDATE())-YEAR(DOJ)
EXP_IN_YEARS
FROM EMP;
```

135. WAQTD THE DETAILS OF TOP 3 EMP WHO HAS THEIR BDAY UPCOMING.

```
SELECT *
FROM EMP
WHERE
DATE_FORMAT(DOB,'%m-%d')>DATE_FORMAT(CURDATE()
,'%m-%d')
ORDER BY DATE_FORMAT(DOB,'%m-%d') ASC
LIMIT 3;
```

SUB-QUERY

136. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY MORE THAN KIRAN.

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

137. WAQTD THE EMP FNAME, SAL IF THE EMP IS GETTING SALARY MORE THAN PRIYA.

```
SELECT FNAME,SAL FROM EMP WHERE SAL>
```

```
(SELECT SAL FROM EMP WHERE FNAME='PRIYA');
```

138. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY LESS THAN SAMEER.

```
SELECT * FROM EMP WHERE SAL<
(SELECT SAL FROM EMP WHERE FNAME='SAMEER');
```

139. WAQTD THE EMP FNAME, JOB ,SAL IF THE EMP IS WORKING AS DISPATCHER AND GETTING SALARY MORE THAN AMAN.

```
SELECT FNAME,JOB,SAL FROM EMP WHERE
JOB='DISPATCHER' AND SAL>
(SELECT SAL FROM EMP WHERE FNAME='AMAN');
```

140. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SAME JOB ROLE AS KARAN'S JOB ROLE AND NAME MUST STARTS FROM VOWELS.

```
SELECT * FROM EMP WHERE JOB=
(SELECT JOB FROM EMP WHERE FNAME='KARAN') AND
SUBSTR(FNAME,1,1)IN('A','E','I','O','U');
```

141. WAQTD THE EMP FNAME, LNAME, DNO, SAL IF THE EMP IS WORKING IN SAME DEPT AS SIDDARTH'S DEPT BUT GETTING SALARY LESS THAN SAMEER.

```
SELECT FNAME,LNAME,DNO,SAL FROM EMP WHERE DNO=
(SELECT DNO FROM EMP WHERE FNAME='SIDDARTH')
```

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

142. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING ALARY MORE THAN PRIYA BUT LESS THAN RAHUL.

```
SELECT * FROM EMP WHERE SAL>
(SELECT SAL FROM EMP WHERE FNAME='PRIYA')
AND SAL<
(SELECT SAL FROM EMP WHERE FNAME='RAHUL');
```

143. WAQTD THE EMP FNAME,LNAME, DOJ IF THE EMP WAS JOINED AFTER MURALI.

```
SELECT FNAME,LNAME,DOJ FROM EMP WHERE DOJ>
(SELECT DOJ FROM EMP WHERE FNAME='MURALI');
```

144. WAQTD THE DEPT NAME OF KARAN.

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

145. WAQTD THE DEPT DETAILS OF KIRAN'S DEPT.

```
SELECT * FROM DEPT WHERE DNO=
(SELECT DNO FROM EMP WHERE FNAME='KIRAN');
```

146. WAQTD THE DETAILS OF EMP WHO ARE WORKING

IN IT DEPT.

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

147. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING IN IT DEPT AND GETTING SLARY MORE THAN AMAN.

```
SELECT * FROM EMP WHERE DNO=
(SELECT DNO FROM DEPT WHERE DNAME='IT')
AND SAL>
(SELECT SAL FROM EMP WHERE FNAME='AMAN');
```

148. WAQTD THE DETAILS OF CUSTOMER WHO LIVES

IN CHENNAI CITY.

```
SELECT * FROM CUSTOMER WHERE LID IN
(SELECT LID FROM LOCATION WHERE
CITY='CHENNAI');
```

149. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING IN IT OR SALES DEPT.

```
SELECT * FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE DNAME IN
('IT','SALES'));
```

150. WAQTD THE DEPT NAME OF EMP WHO IS GETTING SALARY MORE THAN AMAN.

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

151. WAQTD THE DETAILS OF EMP IF THE EMP IS WORKING AS SALESMAN OR MANAGER AND GETTING SALARY MORE THAN ALL THE SALESMAN.

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

or

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

ALL/ANY OPERATORS

These operators are multi valued operators which takes single value at LHS and multiple values at RHS.

Syntax:

```
column-name >/< ALL/ANY
```

(inner-query/v1,v2,..,vn);

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

---

152. WAQTD THE DETAILS OF EMP IF THE EMP IS GETTING SALARY MORE THAN ALL THE EMPS WORKING IN DEPT 112.

```
SELECT * FROM EMP WHERE SAL>ALL
```

```
(SELECT SAL FROM EMP WHERE DNO=112);
```

153. WAQTD THE EMP FNAME, SAL IF THE EMP IS GETTING SLARY MORE THAN ANY OF THE SALESMAN.

```
SELECT FNAME,SAL FROM EMP WHERE SAL>ANY
(SELECT SAL FROM EMP WHERE JOB='SALESMAN');
```

154. WAQTD THE DETAILS OF CUSTOMER IF HE HAS ORDERED ANY PRODUCT.
[Hint: If a customer orders any product, then only his cid will be tracked in orders table]

```
SELECT * FROM CUSTOMER WHERE CID IN
(SELECT CID FROM ORDERS);
```

155. WAQTD THE DETAILS OF PRODUCT THAT HAVE BEEN ORDERED SO FAR.
[Hint: If any product has been ordered, then only product_id will be tracked in orders table]

```
SELECT * FROM PRODUCT WHERE PRODUCT_ID IN
(SELECT PRODUCT_ID FROM ORDERS);
```

156. WAQTD THE DETAILS OF EMP WHO WORKS IN KORAMANGALA.

```
SELECT * FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE LID IN
```

```
(SELECT LID FROM LOCATION WHERE
LOCATION='KORAMANGALA'));
```

157. WAQTD THE DETAILS OF CUSTOMER WHO HAS ORDERED HEADPHONES.

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

158. WAQTD THE WORKING LOCATION OF SIDDARTH.

```
SELECT * FROM LOCATION WHERE LID IN
(SELECT LID FROM DEPT WHERE DNO IN
```

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

159. WAQTD THE DETAILS OF CUSTOMER WHO ORDERED SMARTPHONE.

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

160. WAQTD THE DETAILS OF EMP WHO HAVE DELIVERED PRODUCTS SO FAR.

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

161. WAQTD THE LOCATION DETAILS OF CUSTOMER WHO HAS ORDERED LAPTOP.

```
SELECT * 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='LAPTOP')));
```

162. WAQTD THE DETAILS OF EMP WHO DELIVERS THE PRODUCT TO CHENNAI 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='CHENNAI')));
```

163. WAQTD THE EMP FNAME, LNAME IF THE EMP IS WORKING IN TELANGANA.

```
SELECT FNAME,LNAME FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE LID IN
(SELECT LID FROM LOCATION WHERE
STATE='TELANGANA'));
```

164. WAQTD THE DETAILS OF PRODUCT DELIVERED TO KARNATAKA.

```
SELECT * FROM PRODUCT WHERE PRODUCT_ID IN
(SELECT PRODUCT_ID FROM ORDERS WHERE CID IN
(SELECT CID FROM CUSTOMER WHERE LID IN
(SELECT LID FROM LOCATION WHERE
STATE='KARNATAKA')));
```

165. WAQTD THE DETAILS OF EMP WHO ARE WORKING IN IT DEPT AND ALSO THE CUSTOMER OF THEIR OWN COMPANY.

```
SELECT * FROM EMP WHERE DNO IN
(SELECT DNO FROM DEPT WHERE DNAME='IT')
 AND CID IN
(SELECT CID FROM CUSTOMER);
```

166. WAQTD THE CUSTOMER DETAILS WHO LIVES IN KARNATAKA AND ORDERED SMARTPHONE.

```
SELECT * FROM CUSTOMER WHERE LID IN
(SELECT LID FROM LOCATION WHERE
STATE='KARNATAKA')
AND CID IN
(SELECT CID FROM ORDERS WHERE PRODUCT_ID IN
(SELECT PRODUCT_ID FROM PRODUCT WHERE
PNAME='SMARTPHONE'));
```

167. WAQTD THE DETAILS OF KIRAN'S MANAGER.

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

168. WAQTD THE FNAME,LNAME OF KARAN'S MANAGER.

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

169. WAQTD THE DEPT DETAILS OF SHIVANI'S MANAGER.

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

170. WAQTD THE KIRAN'S MANAGER'S MANAGER DETAILS.

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