

SQL FULL NOTES

SQL INTRO

Data

Data is a raw fact which describe the property/ attributes of on entity.

Entity/object

Anything which is physically exist we can call entity/object.

Data base

Data base is a place or medium where we store the data in systemic organized.

Basic operation performed on database

CRUD- operation

- create
- read
- update
- delete

DBMS

DATA BASE MANAGEMENT SYSTEM

DBMS software to maintain of mange the data base

DBMS provides to important features

1. security
2. authorization

Type of DBMS

1. network DBMS
2. object oriented DBMS
3. hierachial DBMS
4. **RDBMS**

RDBMS




RELATIONAL DATA BASE MANAGEMENT SYSTEM

RDBMS is a type of DBMS software. Which is used to store the data in the form of tables or relation.

Diagram

- DBMS follows the relational model also call it ha RDBMS
- If DBMS follow EF CADD rule then we call RDBMS
- To communicate with RDBMS software we use an language called **structure query language**

Relational model

-  Relational model is designed by EF CADD
-  Relational model store data in the form of table or relation
-  In relational model we can store meta Data

EF CADD RULES

- ✚ Data should be single value data / atomic data
 - ✚ We can stored data in multiple tables and also we can establish connection any two tables by using key attribute.
-

Data types

Types of data are kind of data which is used to store in memory location.

- ✚ CHAR
 - ✚ VARCHAR
 - ✚ LARGE OBJECT (lob)
 - CHARACTER LARGE OBJECT
 - BINARY LARGE OBJECT
 - ✚ DATE
 - ✚ NUMBER
-

Constraints

Condition / rules given to validate the data.

TYPES OF CONSTARINTS

- UNIQUE
 - NOT NULL
 - CHECK
 - PRIMAY KEY
 - FOREGIN KEY
-

SQL STATEMENT / SQL LANGUAGE

1. DATA DEFINATION LANGUAGE (DDL)
 - a. CREATE
 - b. RENAME
 - c. ALTER
 - d. TRUNCATE
 - e. DROP
2. DATA MANIPULATION LANGUAGE (DML)
 - a. INSERT
 - b. UPDATE
 - c. DELETE
3. TARNSACTION CONTROLL LANGUAGE (TCL)
 - a. COMMIT
 - b. SAVE POINT
 - c. ROLL BACK
4. DATA CONTROL LANGUAGE (DCL)
 - a. GRAND

- b. REVOKE
- 5. DATA QUERY LANGUAGE / SELECT QUERY LANGUAGE (DQL)
 - a. SELECT
 - b. PROJECTION
 - c. SELECTION
 - d. JOINS

DATA QUERY LANGUAGE

SELECT -> This statement is used to retrieve the data from the table.

PROJECTION -> This statement is used to retrieve the data from the table by selecting column name.

NOTE: by default in projection all the record get selected.

SELECTION -> retrieve the data from the table by selecting column name and row name.

JOIN -> retrieve the data from the multiple table simultaneously.

SELECT (retrieve the data from the table)

Emp table

| Id | ename | sal | depno |
|----|-------|-----|-------|
| 1 | Arjun | 500 | 20 |
| 2 | Manoj | 200 | 10 |
| 3 | Muthu | 600 | 20 |
| 4 | Siva | 700 | 10 |
| 5 | Surya | 800 | 30 |

1. WRITE A QUERY TO DISPLAY NAME FROM EMP TABLE

SELECT ename
FROM EMP;

o/p :

| Ename |
|-------|
| Arjun |
| Manoj |
| Muthu |
| Siva |
| Surya |

2. WRITE A QUERY TO DISPLAY SAL , D.NO FROM EMP TABLE

SELECT sal, depno
FROM EMP;

o/p:

| Sal | Depno |
|-----|-------|
| 500 | 20 |
| 200 | 10 |
| 600 | 20 |
| 700 | 10 |
| 800 | 30 |

3. WRITE A QUERY TO DISPLAY DETAILS OF EMP

SELECT *
FROM EMP;

O/P:

| Id | Ename | Sal | Depno |
|----|-------|-----|-------|
| 1 | Arjun | 500 | 20 |
| 2 | Manoj | 200 | 10 |
| 3 | Muthu | 600 | 20 |
| 4 | Siva | 700 | 10 |
| 5 | Surya | 800 | 30 |

PROJECTION (*by default all the record get selected*)

SYNTAX :

```
SELECT * / [DISTINGING] COLUMN / EXPRESSION  
FROM <TABLE_NAME>;
```

Order of execution:

1. FROM
2. SELECT

FROM CLAUSE

- ✚ Always from clause will be execute first
- ✚ We can pass table name as a argument
- ✚

Function of from clause

- ✚ From class goes to database search for the table name and put under the execution.

SELECT CLAUSE

- ✚ Select class execute after from clause
- ✚ We can pass star, column name or expression to the select class

Function of select clause

- ✚ select class goes to the table which is under execution and retrieve the data and display the output

Note : Select class responsible to retrieve the data or display the output.

1. How to display all the table names which is present in database.

```
SELECT *
```

```
FROM TAB;
```

(*it's contains all the table name*)

DEPT (table)

| depno | Dname | Location |
|-------|-------------|-----------|
| 10 | Accounting | Thanjavur |
| 20 | Sales | Chennai |
| 30 | Development | Madurai |

2. WRITE A QUERY TO DISPLAY DEPT NAME AND LOCATION FROM THE DEPT TABLE.

```
SELECT dname, location
FROM DEPT;
```

O/P:

| Dname | Location |
|-------------|-----------|
| Accounting | Thanjavur |
| Sales | Chennai |
| Development | Madurai |

-
3. Write a query to display employee name job and salary given to all the employees table

```
SELECT ename, job, sal
FROM EMP;
```

-
4. Write a query to display employee name higher date department number from emp table

```
SELECT ename, hiredate, depno
FROM EMP;
```

-
5. Write a query to display name job salary along with annual salary from emp table

```
SELECT ename, sal , sal*12
FROM EMP;
```

EXPRESSION

Any thing which gives an output is called expression.

- Operands (imgae)
- Operator

6. Write t a query to display name hiredate half term salary from emp Table

```
SELECT ename, hiredate, sal*6
FROM EMP;
```

| ename | Sal | sal |
|-------|-----|------|
| Arjun | 500 | 6000 |
| Manoj | 200 | 2400 |
| Muthu | 600 | 7200 |
| Siva | 700 | 8400 |
| Surya | 800 | 9600 |

| ename | Sal | Annual sal |
|-------|-----|---------------|
| Arjun | 500 | 6000 |
| Manoj | 200 | 2400 |
| Muthu | 600 | 7200 |
| Siva | 700 | 8400 |
| Surya | 800 | 9600 |

DISTINCT *(used to avoid the duplicate values which are present in result end table)*

- Used to remove duplicate value which are present in present in table
- In select class distinct or star must be first argument
- We can pass multiple column name to distinct clause
- In case of multiple column names that distinct clause remove the combination of duplicates

SELECT sal
FROM EMP;

| SAL |
|-----|
| 100 |
| 200 |
| 300 |
| 200 |
| 500 |

SELECT DISTINCT sal
FROM EMP;

| SAL |
|-----|
| 100 |
| 200 |
| 300 |
| 500 |

SELECT distinct sal,depno
FROM emp;

| sal | Depno |
|-----|-------|
| 100 | 20 |
| 200 | 10 |
| 300 | 30 |
| 500 | 10 |

DIFFERENT BETWEEN UNIQUE AND DISTINCT ?

| UNIQUE | DISTINCT |
|---------------------------------------|---|
| Remove duplicates from original table | Remove duplicates from result end table |

SELECTION

SYNTAX:

SELECT * / [DISTINGING] COLUMN / EXPRESSION
FROM TABLE_NAME
WHERE < TABLE.NAME.CONDITION>;

- It is used to do filter the record from the table not original table
- Wireless always execute row by row
- Where clause always true or false condition
- We can pass multiple condition to the where clause
- We cannot able to use alia's name in where clause

| Id | ename | Sal | depno |
|----|-------|-----|-------|
| 1 | Arjun | 500 | 20 |
| 2 | Manoj | 200 | 10 |
| 3 | Muthu | 600 | 20 |
| 4 | Siva | 700 | 10 |
| 5 | Arjun | 800 | 30 |

WAQTD emp name and sal given to all the emp if the ename is 'arjun'.

```
SELECT ename, sal
FROM emp
WHERE ename = 'arjun' ;
```

arjun = arjun
~~manoj = arjun~~
~~muthu = arjun~~
~~Siva = arjun~~
 Arjun = arjun

O/P: where clause

| Id | Ename | Sal | Depno |
|----|-------|-----|-------|
| 2 | Arjun | 500 | 10 |
| 5 | Arjun | 800 | 30 |

O/P : select clause

| Ename | Sal |
|-------|-----|
| Arjun | 500 |
| Arjun | 800 |

1. WAQTD DETAILS OF AN EMP IF THE EMP NAME IS MANOJ

```
SELECT *
FROM EMP;
WHERE ename = 'manoj';
```

2. WAQTD EMP NAME AND JOB IF THE EMP IS WORKING AS A SALESMAN

```
SELECT ename, job
FROM emp
WHERE job = 'salesman';
```

3. Write a query to display name job and salary given to all the employees if the employee is getting salary more than 1200

```
SELECT ename, job, sal
FROM emp
WHERE sal > 1200;
```

-
4. Write a query to display details of an employee if the employee working in department number 30

```
SELECT *  
  
FROM emp  
  
WHERE depno = 30;
```

-
5. Display employee number employee name and salary given to all the employees if the employee number is 7788

```
SELECT ename, sal,  
  
FROM emp  
  
WHERE empno = 7788;
```

-
6. Write a to display details of an employee who hired after 81

```
SELECT *  
  
FROM emp  
  
WHERE hiredate > 31 / dec / 81;
```

-
7. Details of an employee who heard before 82

```
SELECT *  
  
FROM emp  
  
WHERE hiredate < 1 - jan - 82;
```

-
8. Write a query to display details of all the employees along with annual salary if the annual salary more than 2500

```
SELECT *, sal*12 AS annual_sal  
  
FROM emp  
  
WHERE sal*12 > 2500;
```

-
9. Write a query to display details of an employee except who were working as a president

```
SELECT *  
  
FROM emp  
  
WHERE job != 'president';
```

10. Write a query to display details of an employee if the employee getting commission more than salary

```
SELECT *  
  
FROM emp  
  
WHERE com < sal ;
```

-
11. Write a query to display name job and department number of employee if the employee working as a clerk in department 30

```
SELECT ename, job, depno  
  
FROM emp  
  
WHERE job = 'clerk' AND depno = 30;
```

OPERATORS

- Arithmetic operator (+, -, *, /)
- Comparison operator (=, !=, <>)
- Relational operator (>, <, >=, <=)
- Logical operator (AND, OR, NOT)
- special operator (IN, NOT IN, BETWEEN, NOT BETWEEN, LIKE, NOT LIKE, IS, ESCAP
- concatenation operator (||)
- subquery operator (ALL, ANY, EXIST, NOT EXIST)

AND -> return true if both condition is true.

OR -> return true if any one condition is true.

12. Details of an employee if the employee working as a salesman or manager

```
SELECT *  
  
FROM emp  
  
WHERE job = 'clerk' OR job = 'manager';
```

-
13. Display employee name job department number if the employee is working as a president in department number 10 or 20

```
SELECT ename, job, depno  
  
FROM emp  
  
WHERE job = 'president' AND ( depno = 10 OR depno = 20 );
```

14. Details of an employee who were working as a salesman and getting salary more than 1200

```
SELECT *  
  
FROM emp  
  
WHERE job = 'saleman' AND sal > 1200;
```

15. Write a display details of an employee who were getting commission as 0 or 500 in department number 20 or 30

```
SELECT *  
  
FROM emp  
  
WHERE (com = 0 OR com =500 ) AND ( depno = 20 OR depno = 30 );
```

16. Write a query display details of an employee who were getting salary more than 1250 but less than 4000

```
SELECT *  
  
FROM emp  
  
WHERE sal < 1250 AND sal > 4000;
```

17. Display name job higher date of an employee if the employee is hired after 80 and getting salary more than 1200 and working as a salesman

```
SELECT ename, job, hiredate  
  
FROM emp  
  
WHERE hiredate > 31-dec-81 AND sal >1200 AND job = 'saleman';
```

18. Details of an employee who were hired during the year 81

```
SELECT *  
  
FROM emp  
  
WHERE hiredate < 01-jan-81 AND hiredate > 31-dec-81 ;
```

19. Employee name job department number if the employee is working as a analyst salesman or manager in department number 10 20 or 30


```
SELECT ename, job, hiredate
```

```
FROM emp
```

```
WHERE ( job = 'analyst' OR job = 'salesman' OR job = 'manager' )
```

```
AND ( depno = 10 OR depno = 20 OR depno = 30 );
```

IN OPERATOR

 Multiple value operator which can be accepted multiple value of RHS and single value of LHS

SYNTAX :

```
COLUMN_NAME / EXPRESSION IN (V1,V2,V3 ... Vn);
```

EX: Sal **IN** (100,200,300,400); (like OR operator)

```
300 = 100 -----> false
```

```
300 = 200 -----> false ( in operator return true if any one condition is true )
```

```
300 = 300 -----> true
```

```
300 = 400 -----> false
```

1. Details of on emp if the emp is working in dep no 10,20,30,40

```
SELECT *
```

```
FROM emp
```

```
WHERE depno IN ( 10, 20, 30, 40 );
```

-
2. Details of an emp who where working as a salesman or manager in depno 30 or 40.

```
SELECT *
```

```
FROM emp
```

```
WHERE depno IN ( 30, 40 );
```

NOT IN OPERATOR



Multiple value operator which can be accepted multiple value of RHS and single value of LHS

SYNTAX :

COLUMN_NAME / EXPRESSION **NOT IN** (V1,V2,V3 ... Vn);

EX: Sal **NOTIN** (100,200,300,400); (like AND operator)

300 != 100 -----→ true

300 != 200 -----→ true (not in operator return true if all the condition is true)

300 != 300 -----→ false

300 != 400 -----→ false

1. Details of an emp who where working in dep no 20 or 30 and except analyst ,clerk, president

SELECT *

FROM emp

WHERE depno **NOTIN** (20,30) **AND** job **NOTIN** ('ANALYST', 'clerk', 'preasident');

-
2. Empno, ename , job and commission if the emp is working as a manager or salesman and getting commission 300 or 400 except empno 7788 or 7566.


SELECT enmae, job, com

FROM emp

WHERE job **IN** ('salesman', 'manager') **AND** com **IN** (300,400)

AND empno **NOTIN** (7788,7566);

BETWEEN OPERATOR

 This operator used to whenever we across range of value

SYNTAX:

COLUMN_NAME / EXPRESSION **BETWEEN** lower_rang **AND** higer_rang ;

EX: Sal **BETWEEN** 1250 **AND** 3000;

1250<--|-----|-----|-----|-----> 3000 *(include 1250 and 3000)*
 1350 1500 2000 2500

1. Details of an emp who where getting commission between 300 and 500.
2. Ename ,job, and hire date if the emp where hiredate during the year 81.
3. Details of an emp along with annual salary if the emp is getting sal more than 1250 but less than 3000.
4. Ename, sal, hire date of an emp who where hired after 81 and except who where getting sal 1250.

NOT BETWEEN OPERATOR



This operator used to whenever we across range of value

SYNTAX:

COLUMN_NAME / EXPRESSION **NOT BETWEEN** lower_rang **AND** higer_rang ;

EX: Sal **NOT BETWEEN** 1250 **AND** 3000;

(this rang of value not valid)

1250<--|-----|-----|-----|-----> 3000 *(include 1250 and 3000)*
 1350 1500 2000 2500

1. Ename, sal, com given all emp who where getting commission between 0 to 1200 and except who where getting sal between 2500 to 5000.
2. Details of an emp except who where getting commission between 0 to 1400.
3. Ename, sal if the emp getting sal less than 1250 but more than 2500.
4. Ename , commission given to all emp if the emp getting commission more than 0 but less than equal to 1000.

IS OPERATOR



This operator used to check whether LHS is null or not null

SYNTAX :

COLUMN_NAME / EXPRESSION **IS** NULL / NOTNULL ; (*is like =*)

1. Details of an emp who where getting com
2. Details of an emp not getting com
3. Details of an emp who where having reporting manager.
4. Details of an emp who where not having any reporting manger.

LIKE OPERATOR

- ✚ It is used to perform 'pattern matching'
- ✚ In like operator we can use two special characters for pattern matching
 - % ----> Percentile
 - _ ----> Underscore

| | | |
|-----|---|-----------------------|
| ✚ % | it will be accept <u>any character any number of time</u> | (A-Z) (0 – N) |
| ✚ _ | it can accept <u>any character only once</u> | (special character) |

SYNTAX :

COLUMN_NAME / EXPRESSION **LIKE** 'PARTTEN_MATCH' ;

1. Emp name starts with S.
2. Emp name end with A.
3. Emp name contains A
4. Emp name alternative A.
5. If the emp contains concigitive E (EE)
6. Emp name contains 2 A
7. Emp name second character I
8. Emp name second last char p
9. Name start with S and end with R
10. Name start with M or B
11. Details of an emp who where working as a salesman or manager and getting 3 digit com
12. Details of an emp if the name ends with R or E and getting S and getting 4 getting sal
13. Details of an emp along with annual sal if the emp is getting annual salary which ends with 0.
14. Name, job and hiredate of an emp who where hired during the year 81.
15. Details of an emp who where hired in the month of dec or jan.
16. Details of an emp if they where getting commission with out using is operator.

NOT LIKE OPERATOR

SYNTAX :

COLUMN_NAME / EXPRESSION **NOT LIKE** 'PARTTEN_MATCH' ;

1. Name of an emp if the emp name doesn't stats with s
 2. Details of an emp if the emp job doesn't ends with 'MAN'
 3. Details of an emp if the ename doesn't start with S or M
-

ESCAP

- ✚ keyboard used to remove the special behaviour of character () and create it as a ordinary character which is present next to it. Any character but only one

SYNTAX :

(spl char like @ / # /!)

COLUMN_NAME / EXPRESSION **LIKE** 'PARTTEN_MATCH' **ESCAP** '1 CHAR ' ;

1. Name contains _ in it
2. Name and age of an emp if the ename contains atleast two % in it.
3. Details of an emp the emp name second last name contain character is %.

CONCATINATION (||)

- ✚ Used to do combin of two string
- ✚ Joining any to string is called concatenation

1. Mr. smith your salary is 800 rs.
2. I am ragul from thanjavur.

FUNCTIONS

✚ it is a set of instruction / block of code which is used to perform some specific task.

TYPES OF FUNCTIONS

- ✚ Inbuilt function
 - ❖ Single row function
 - ❖ Multi row function
- ✚ User defined function

MULTIROW FUNCTION

- ✚ It can accept n number of input and generate single output
- ✚ Multi row function execute group by group
- ✚ It also known as aggregate function or group function (image)

MULTI ROW FUNCTIONS TYPE:

- MAX()
- MIN()
- AVG()
- SUM()
- COUNT()

- ✚ Execute group by group
- ✚ We can make multi row function in select clause
- ✚ We cannot able to use multi row function in where clause. because where clause execute row by row but multi row function execute group by group.
- ✚ We cannot able to select column name expression along with the multi-row function
- ✚ Multi-row function ignore null value
- ✚ Along with multi-function we can use only group by expression
- ✚ We cannot able to nest multi row function
- ✚ We can pass only one argument

MAX()

✚ This function is used to obtain maximum values from given table.

SYNTAX:

MAX (COLUMN_NAME / EXPRESSION) ;


MIN()

✚ This function is used to obtain maximum values from given table.

SYNTAX:

MIN (COLUMN_NAME / EXPRESSION) ;


AVG()

 This function is used to obtain average values from given table.

SYNTAX:

AVG (COLUMN_NAME / EXPRESSION) ;

SUM()

 This function is used to obtain sum of values from given table.

SYNTAX:

SUM (COLUMN_NAME / EXPRESSION) ;

COUNT()

 This function is used to obtain total number of values from given table.

SYNTAX:

COUNT (COLUMN_NAME / EXPRESSION) ;

-
1. Maximum sal given to all the emp from emp table.
 2. Total sal needed to pay for all the salesman
 3. Total salary and average salary needed to pay for all the employee except president
 4. Number of employees working as a manager
 5. Number of employee who were not working as a clerk or analyst except department number 10 or 20
 6. Maximum salary total salary along with minimum commission given to all the employee if the employee does not starts with S and does not ends with A.
 7. Number of employee getting commission in department number 20 or 30 and they were working as a salesman
 8. Number of present in the emp table
 9. Number of department present in the emp table
 10. Recently hired employee
 11. First hired employee

SINGLE ROW FUNCTION

- ✚ It can accept n number of input and generate n number of output
- ✚ Single function rope by row execute

DUAL → it is dummy table which is used to get some output.

D → Dummy

X → value

LOWER ('str') → to convert the given character into lower case

```
SELECT LOWER ( 'PENTAGON' )  
FROM DUAL;                O/P: pentagon
```

UPPER ('str') → To convert the given character into uppercase

```
SELECT UPPER ( 'pentagon' )  
FROM DUAL;                O/P: PENTAGON
```

REVERSE ('str') → To convert the given character into reverse

```
SELECT REVERSE ( 'PENTAGON' )  
FROM DUAL;                O/P: NOGATNEP
```

LENGTH ('str') → To obtain the length of given string

```
SELECT LENGTH ( 'PENTAGON' )  
FROM DUAL;                O/P: 8
```

INITCAP ('str') → To convert the first letter of the word into uppercase

```
SELECT LOWER ( 'this is word ' )  
FROM DUAL;                O/P: This Is Word
```

MOD (m,n) → Used to do get reminder

```
SELECT MOD (9,3)  
FROM DUAL;                O/P: 0
```

CANCAT ('str 1', 'str 2')→To combing any two string

```
SELECT CONCAT ('mr. ', ( CONCAT ( ename, ( CONCAT ( 'your sal is ', ( CONCAT (sal, 'rs')))))));  
FROM EMP;
```

1. To convert all the employee name into reverse
2. Find the length of the name
3. To convert all the name of a to upper case
4. To convert all the name should lowercase
5. To create first letter of the word is uppercase

SUBSTRING



This function used to do obtain substance from the original string

SYNTAX:

SUBSTR ('original_string', 'position', [length]); ([] = *optional*)

1. Details of an employee if the employee name starts with M without using like operator
2. Employee name job if the employee name ends with or without using like operator
3. Employee name job and higher rate if the employee job starts with MAN or SAL
4. Details of an employee if the employee name starts with m and second character is a without using like operator
5. Details of an employee first of of the employee name is lower
6. Second half of the name in reverse format
7. Details of an employee first half of the name in lower case and second of in reverse case
8. Details of an employee if the employee name does not starts with M without using like operator
9. Name job salary and commission given to all the employee if the employee name having six characters and getting 4 digit salary
10. Details of an employee if the employee name second last character is E

REPLACE



This function used to replace the substring from the given new string in the original string


SYNTAX:

```
REPLACE ('original_string', 'sub_string', 'new_sting' );
```

Ex:

1. Number of time particular character occurred in given string for example how many times a present in the string
2. Employee name if the employee name contains exactly 1A without using like operator
3. Name salary job given all the employee if the employee job contains exactly 2 S without using like operator

INSTR(INDEX OF STRING)


 This function used to obtain index value of substring from the original string

SYNTAX:

INSTR (' original_string', 'sub_string', 'position', [Nth occurrence]) ;

1. name contains at least one way without using like and substring
2. Name contains at least 2a without using like and substring
3. Contains exactly one A init

SYSDATE


 The function used to obtain current date from the system where RDBMS software installed.

SELECT SYSDATE

FROM DUAL;

O/P: 15-JUN-22

STSTIME STAMP


 function used to do after in clock time along with the time zone

SELECT SYSTIME STAMP

FROM DUAL;

O/P: 15-JUN-22 12:00 05:30

TO-CHAR()

 Character from the given date based on format model

SYNTAX:

TO_CHAR (date, 'formate_model');

FORMAT – MODEL (15 – MAR – 2020)

YYYY – 2020

YY – 20

YEAR – MARCH

MON – MAR

MM – 03

DAY – MONDAY
SAT – 7

SUNDAY – 1, MON – 2, TUES – 3 , W – 4 , THU – 5 , FRI – 6,

DD – 15

DY- MON

MI – 28 (MIN)

SS – 54 (SEC)

Ex

SELECT **TO_CHAR** (SYSDATE, 'YEAR')

FROM DUAL;

SELECT **TO_CHAR** (SYSDATE, 'YYYY')

FROM DUAL;

1. Details of an emp if emp hired in the month of December.

SELECT *

FROM EMP

WHERE **TO_CHAR** (HIREDATE, 'MON') = ' DEC' ;

-
2. Name, job, hiredate if emp were hired during the year 81

SELECT ename, job, hiredate

FROM EMP

WHERE **TO_CHAR** (HIRED , 'YEAR') = 81 ;

-
3. Emp name, hiredate if the emp hired in the mon of feb , sep, dec.

SELECT ENAME, HIREDATE

FROM EMP

```
WHERE ( HIREDATE , 'MON' ) IN ( 'FEB', 'SEP', 'DEC' );
```

4. Details of an emp who were hired on 17th or 25th

```
SELECT *  
FROM EMP  
WHERE TO_CHAR ( HIREDATE, 'DD' ) IN ( 17, 25);
```

5. If the emp doesn't start with s and the were hired on 'sun' 'mon' 'sat'

```
SELECT *  
FROM EMP  
WHERE SUBSTR (ENAME , 1 ,1 ) != 'S' AND TO_CHAR (HIREDATE , 'DY' )  
                                     IN ( 'SAT', 'SUN' , 'MON' );
```

6. All the emps hiredate in US formate

```
SELECT TO_CHAR ( hiredate, 'mm - dd - yyyy' )  
FROM EMP;
```

US → MM – DD – YYYY

IND→DD – MM – YYYY

NVL



This function used to replace null value from the given value.

SYNTAX:

NVL (arg1 , arg2); (by default 0)

arg 1 - in this we need to pass column name which can be NULL

arg2- we need to pass values and this value will be considered when argument one is null

emp

| ENAME | SAL | COMM | SELECT SAL + COM |
|-------|-----|------|-------------------|
| Arjun | 400 | 20 | FROM EMP; |
| Muthu | 500 | NULL | 400 + 20 → 420 |
| manoj | 600 | 40 | 500 + NULL → NULL |
| | | | 600 + 40 → 640 |

1. Need to pay for all the employees salary plus commission if commission value null it will be considered zero

```
SELECT NVL ( SAL, 0 ) + NVL ( COM, 0 )  
FROM EMP;
```

-
2. Sal added with commission of all the EMPs.

GROUP BY

EMP

| ID | EMP | SAL | DNO |
|----|---------|-----|-----|
| 1 | SUNDARA | 500 | 20 |
| 2 | SUNDARI | 600 | 10 |
| 3 | MACHA | 900 | 20 |
| 4 | MACHI | 800 | 10 |
| 5 | RAMESH | 400 | 30 |

ORDER OF EXECUTION

1. FROM
2. WHERE
3. GROUP BY

Group by clause always executes row by row

d.no = 20

| Id | Name | sal | Dno |
|----|--------|-----|-----|
| 1 | Sundar | 500 | 20 |
| 3 | macha | 900 | 20 |

d.no = 10

| Id | Name | sal | Dno |
|----|---------|-----|-----|
| 2 | Sundari | 600 | 10 |
| 4 | Machi | 400 | 10 |

Dno = 30

| Id | Name | sal | Dno |
|----|--------|-----|-----|
| 5 | ramesh | 400 | 30 |

SYNTAX:

```
SELECT * / [DISTINGUISHING] COLUMN / EXPRESSION  
FROM TABLE_NAME  
WHERE < TABLE.NAME.CONDITION >  
GROUP BY ( COLUMN_NAME );
```

CHARACTERISTICS OF GROUPBY CLAUSE:

- ✚ It is used to do group the records
- ✚ Group A class executes row by row
- ✚ Without using where clause we can write group by clause
- ✚ Group A class executes after the execution of from clause
- ✚ Group A class executes before but after execution it creates a group
- ✚ Any class which executes after group A class it executes as group by group
- ✚ We can pass multiple column names in group clause
- ✚ We can select multi-row function and group by expression along with the group by clause

1. Minimum salary to all the employees in each department

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

2. Total salary needed to pay for all the employees in each job

```
SELECT SUM ( SAL )  
FROM EMP  
GROUP BY DNO ;
```

3. Number of employees working as a manager in each department

```
SELECT COUNT ( * )  
FROM EMP  
WHERE JOB = 'MANAGER'  
GROUP BY DNO ;
```

4. Average salary need to pay for all the clerk and analyst in each department

```
SELECT AVG ( SAL )  
FROM EMP  
WHERE JOB IN ( 'CLERK', 'ANALYST' );  
GROUP BY DEPNO;
```







5. Number of employee getting salary more than 1200 in each department
6. Maximum salary mini salary average salary and total salary need to pay for all the employee if the employee name contains A in each department

HAVING (execute group by group)

SYNTAX:

```
SELECT * / [DISTINGING] COLUMN / EXPRESSION  
FROM TABLE_NAME  
WHERE < TABLE.NAME.CONDITION>  
GROUP BY ( COLUMN_NAME )  
HAVING < GROUP_FILTER. CONDITION>;
```

CHARACTERISTICS :

-  Having class is used to do filter the group function or multi-function
-  Having execute after group class
-  Execute group by group
-  We can pass multirow function as argument
-  Execute true or false condition
-  We can pass multiple condition

1. Maxim salary given to all the employee if the employee getting salary more than 2000 and maximum salary less than 4200 in each department

```
SELECT MAX ( SAL )  
FROM EMP  
WHERE SAL < 2000  
GROUP BY DNO  
HAVINNG MAX (SAL ) > 4200 ;
```

2. Min salary given to all the all the manager in each department if they are getting min salary more than 500

```
SELECT MIN ( SAL )  
FROM EMP  
WHERE JOB = 'MANAGER'  
GROUP BY DNO  
HAVING MIN ( SAL ) < 500 ;
```

3. Average salary needed to pay for all the analyst in department if they are getting AVG salary more than 2200









```
SELECT SUM ( SAL )  
FROM EMP  
WHERE JOB = 'ANALYST'  
GROUP BY DNO  
HAVING AVG ( SAL ) > 2200 ;
```

4. Number of employee present in each job in which at least two employee are working

```
SELECT COUNT ( * )  
FROM EMP  
GROUP BY DNO  
HAVING COUNT ( * ) >= 2 ;
```

-
5. Maximum salary given to department number 30
6. Average salary given to all the employees in each department if they are getting average salary between 2250 and 3000
7. Maximum commission and minimum commission to all the salesman in each department if they are getting salary more than 1250
8. Number of employee working in each except president in which at least three employee and atmost 6 employee where working
9. Duplicate salary present in employee table
10. Hire date of an employee if there hired same day
11. Number of employees getting same salary in same department
12. Number of employees hired on same date in same department

ORDER BY

-  Is used to arrange the data either in ascending or descending order
-  Order by execute after the execution of select class
-  Must be the last argument for any query
-  By default order by class take ascending order
-  All the tables by default arranged in ascending order based on primary key attribute
-  We can pass alia's name in order by class
-  We can pass number as a argument for order by class
-  We can pass multiple column name to order by class

SYNTAX:

```
SELECT * / [DISTINGING] COLUMN / EXPRESSION  
  
FROM TABLE_NAME  
  
WHERE < TABLE.NAME.CONDITION>  
  
GROUP BY ( COLUMN_NAME )  
  
HAVING < GROUP_FILTER. CONDITION>  
  
ORDER BY ( COLUMN_NAME / EXPRESSION ASC/ DESC );
```

1. Salary of employee in descending order

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

-
2. Name of the employee in ascending order

```
SELECT NAME  
FROM EMP  
ORDER BY NAME ASC;
```

-
3. Department number in ascending order

```
SELECT DNO  
FROM EMP  
ORDER BY DNO;
```

SUBQUERY

WORKING PRINCIPLE OF SUB QUERY

OUTER QUERY

INNER QUERY

✚ Inner query first execute and the output of the inner query, outer query executes

take the input form the inner query.

WHY / WHEN

✚ When ever we come across the unknown values use subquery.

RULES TO WIRTE SUB QUERY

- We can select only on column in inner query (where sal > (com , dno) can't use
Sal > (100 , 500) can't use)
- The column slected in inner query and column return outer query must have same data type.

Where sal > (select ename (can't use)
Sal > (com (we can use)
Sal > (sal (we can use)

1. Who getting salary more than sundari salary

```
SELECT SAL
FROM EMP
WHERE SAL > ( SELECT SAL
              FROM EMP
              WHERE NAME = 'SUNDARI' );
```

-
2. Details of an employee if the employee getting salary more than Manoj salary

```
SELECT *
FROM EMP
WHERE SAL > ( SELECT SAL
              FROM EMP
              WHERE ENAME = 'MANOJ' );
```

3. Details of an employee who hired after Muthu

4. Name and job if an employee except manager and if they were getting salary less than Arjun

```
SELECT ENAME
FROM EMP
WHERE JOB != 'MANAGER' AND SAL > ( SELECT SAL
                                   FROM EMP
                                   WHERE ENAME = 'ARJUN');
```

5. Name and salary along with job if the employee is working as a salesman and getting commission and higher date before Lakshmanan

```
SELECT NAME , SAL, JOB
FROM EMP
WHERE JOB = 'SALESMAN' AND COM IS NOT NULL AND
      HIREDATE > ( SELECT HIREDATE
                  FROM EMP
                  WHERE ENMAE = 'LAKESHPMANAN' );
```

6. Details of an employee if they were working in same job as Surya's job

7. Number of employee working in Arjun department

8. Details of an employee who were working is a manager at a getting salary more than Smith salary

9. Employee name salary and commission given to all the employees if the employee getting salary more than salesman salary and name contains at least 1A and they were getting 4 digit salary

10. Details of an employee if the employee getting salary more than Arjun salary but less than Muthu salary

11. Details of an employee the employee were hired after Muthu but before Venkatesh

12. Details of an employee if the employee hired after 81 but before Muthu

Case 2

EMP

| ID | ENAME | DNO FK |
|----|-------|----------|
| 1 | ARJUN | 10 |
| 2 | MANOJ | 30 |
| 3 | MUTHU | 20 |

DEPT

| DNO PK | DNAME | LOCATION |
|----------|----------|-----------|
| 10 | ACCOUNTS | THARJAVUR |
| 20 | SALES | MADURAI |
| 30 | RESEARCH | CHENNAI |

Data to be displayed from one table of the condition to be executed from another table sub query case 2.

1. Department name of sundara

```
SELECT DNAME
FROM EMP
WHERE DNO = ( SELECT DNO
              FROM EMP
              WHERE ENAME = 'SUNDARI' );
```

-
2. Department name of Arjun

```
SELECT DANAME
FROM DEPT
WHERE DNO = ( SELECT DNO
              FROM EMP
              WHERE ENAME = 'ARJUN' );
```

-
3. Location name of Ajay

```
SLEECT LOCATION
FROM DEPT
WHERE DNO = ( SELECT DNO
              FROM EMP
              WHERE ENME = 'AJAY' );
```

-
4. Number of employee working in salesman

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

5. Details of an employee whoever working in Chennai location

```
SELECT *  
FROM EMP  
WHERE DNO = ( SELECT DNO  
              FROM DEPT  
              WHERE LOCATION ='CHENANI' ) ;
```

-
6. Department name and location of all employee except president
7. Details of an employee whoever working in accounting department
8. Details of an employee whoever working in accounting or sales department
9. Single row subquery
10. Details of an employee who were working in Chennai or Coimbatore location
11. Details of an employee who where working sales department and more than Manoj salary
12. No.of employee in Arjun department and they where working to accounting or sales dep
13. Details of an employee if the employee hired after Shiva working in Chennai Coimbatore
Madurai
14. Details of an employee who were getting salary more than kamesh but less than Vicky and
working in sales account and research department
15. Department name in which at least three salesman are working
16. Details of an employee who were getting salary more than miller salary

Multirow subquery

All operator

It is a multi value operator which can accept multiple values at RHS and single value in LHS along with relation operator.

SYNTAX:

CLOUMN_NAME / EXPRESSION **RELATION_OPERATOR ALL** (V1, V2, V3 ... VN);

SAL > ALL (100, 200, 300, 400);

300 > 100 true

300 > 200 true (and operation)

~~300 > 300~~ false

~~300 > 400~~ false

1. Details of an employee who were getting salary more than salesman

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

-
2. Details of an employee who were getting salary more than manager

Any operator

It is a multi value operator which can accept multiple values at RHS and single value in LHS along with relation operator.

SYNTAX:

CLOUMN_NAME / EXPRESSION **RELATION_OPERATOR ANY** (V1, V2, V3 ... VN);

SAL > ANY (100, 200, 300, 400);

300 > 100 true

300 > 200 true (or operation)

~~300 > 300~~ false

~~300 > 400~~ false

1. Details of an employee who were getting salary more than any of the manager

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


2. Name job salesman given to all the employees if the employee getting salary more than Arjun and working Chennai or Coimbatore as a salesman or manager or analyst
3. Location in which at least two analyst where working
4. Department name in which at least four employee or working and atmost 6 employee or working
5. First maximum salary
6. Find nth maximum salary

Employee & manager relationship

EMP

| <u>ID</u> | <u>ENAME</u> | <u>MGR</u> |
|-----------|--------------|------------|
| 1 | ARJUN | 2 |
| 2 | MANOJ | 4 |
| 3 | MUTHU | NULL |
| 4 | SIVA | 1 |

Type 1 (to find manager)

1. Manager name of Arjun


```
SELECT NAME
FROM EMP
WHERE ID = ( SELECT MGR
             FROM EMP
             WHERE ENAME = 'ARJUN');
```

2. Manager name of Muthu




```
SLEECT NAME
FROM EMP
WHERE ID = ( SELECT MGR
             FROM EMP
             WHERE ENAME = 'MUTHU' );
```

3. Smith manager details
4. Details of an president manager
5. Department name of Shiva manager
6. Location name of Surya manager
7. Arjun managers manager details
8. Details of an EMP manages manager
9. Akash's managers manager's managers manager

Type 2 (to find the reporter)

1. Name of employee who were reporting to Arjun
2. Number of employee reporting to kamesh
3. Details of an employee who were reporting to Amarnath
4. Details of an employee who were reporting to analyst
5. Name of the employee who were reporting to arjun's managers
6. Department name of of an employee reporting to Arjun's manager
7. Name of the employee who were acting Asia reporting manager
8. Name of the employee who were not acting is here reporting manager
9. Name of the employee who were having at least three reporters
10. Details of an employee who were have at least one reporting at most two reporting persons
11. Department name who were having at least four reporters
12. Number of employees direct reporting to Arjun
13. Number of employee in directly reporting to ArjuN





JOINS

-  Joining any two table to obtain from multiple table simultaneously
-  Joins compare to subquery more efficient.

Types of joints

1. Cartesian join or cross join
2. Inner join
3. Natural join
4. Outer join left outer join right outer join full outer join
5. Self join

CARTESIAN JOIN OR CROSS JOIN

-  Cartesian join is a type of join on which the join table will be joined to obtained cartesian product of two table
-  In cartesian join record of table 1 will be merged with the all the record of table 2
-  If the two tables are joined in cartesian join that total number of records we got
 - Total number of record = $T1 * T2$
-  If the tables are joined in cartesian join
 - Total no of column = $T1 + T2$

SYNTAX:

ANSI: → AMERICAN NATIONAL STANDARD INSTITUTE

ANSI

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 CROSS JOIN TABLE_NAME 2;
```

ORACLE

```
SELECT COLUMN_NAME
FROM TABLE_NAME 1 , TABLE_NAME 2;
```

Boys

| Bid | Bname | Gid |
|-----|--------|-----|
| 1 | Munna | 2 |
| 2 | Chinna | 1 |
| 3 | Nibba | 3 |

Girls

| Gid | Gname |
|-----|-------|
| 1 | Chimi |
| 2 | Munni |
| 3 | Nibbi |

1. WAQTD details from boys and girls table ?

ANSI

```
SELECT *
FROM BOYS CROSS JOIN GIRLS;
```

ORACLE

```
SELECT *
FROM BOYS, GIRLS;
```

boys


| Bid | Bname | Gid | Gid | Gname | |
|-----|--------|-----|-----|-------|-------|
| 1 | Munna | 2 | 1 | Chimi | False |
| 1 | Munna | 2 | 2 | Munni | True |
| 1 | Munna | 2 | 3 | Nibbi | False |
| 2 | Chinna | 1 | 1 | Chimi | True |
| 2 | Chinna | 1 | 2 | Munni | False |
| 2 | Chinna | 1 | 3 | Nibbi | False |
| 3 | Nibba | 3 | 1 | Chimi | False |
| 3 | Nibba | 3 | 2 | Munni | False |
| 3 | Nibba | 3 | 3 | nibbi | True |

Total.no of rec = 9 .

DRAW BACK

We will get more no.of invalid rec / error rec .

INNER JOIN

 Used to obtain only match record

SYNTAX:

ANSI

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 INNER JOIN TABLE_NAME 2  
ON < JOIN_CONDITION >;
```

ORACLE

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 , TABLE_NAME 2  
WHERE < JOIN_CONDITION >;
```

ON → it is a keyword on which we write join condition.

<join_condition> → Table_name1.column_name = Table_name2. column_name;

-
1. WAQTD details from boys and girls table ?

ANSI

```
SELECT *  
FROM boys INNER JOIN girls  
ON boys.gid = girls.gid ;
```

ORACLE

```
SELECT *  
FROM boys , girls  
WHERE boys.gid = girls.gid ;
```

| | | |
|------------------|------------------|------------------|
| 2 = 1 | 1 = 1 | 3 = 1 |
| 2 = 2 | 1 = 2 | 3 = 2 |
| 2 = 3 | 1 = 3 | 3 = 3 |

OUTPUT OF SELECT CLAUSE

| Bid | Bname | Gid | Gid | Gname |
|-----|--------|-----|-----|-------|
| 1 | Munna | 2 | 2 | Munni |
| 2 | Chinna | 1 | 1 | Chimi |
| 3 | Nibba | 3 | 3 | nibbi |

Only matched rec based on equal condition.

2. Employee name and department name from emp table and department table

ANSI

```
SELECT ename , dname
FROM emp INNER JOIN dept
ON emp.deptno = dept.deptno ;
```

ORACLE

```
SELECT ename, dname
FROM emp,dept
WHERE emp.deptno = dept.deptno ;
```

-
3. Employee name and department details of an employee if the employee working in Chicago or Dallas

ANSI

```
SELECT ename , dept.*
FROM emp INNER JOIN dept
ON emp.deptno = dept.deptno AND
WHERE dept.location IN ( 'chicago', 'dallas');
```

4. Employee name job salary location along with department number location if the employee working salesman or analyst and getting salary more than 1250 and working in a accounting or sales department

ORACLE

SELECT ename, job, sal, dname


FROM emp,dept

WHERE emp.deptno = dept.deptno **AND**

Job **IN**('saleman', 'analyst') **AND** sal >1250

AND dname **IN** ('sales', 'accounting');

OUTER JOIN

 used to do obtained record along with unmature records.


Boys

| Bid | Bname | Gid |
|-----|--------|-----|
| 1 | Munna | 2 |
| 2 | Chinna | 1 |
| 3 | Nibba | 3 |
| 4 | macha | |

Girls

| Gid | Gname |
|-----|-------|
| 1 | Chimi |
| 2 | Munni |
| 3 | Nibbi |
| 4 | machi |

LEFT OUTER JOIN

 To obtain match today and unmatched record of left table we can use left outer join

SYNTAX:

ANSI

SELECT COLUMN_NAME

FROM TABLE_NAME 1 [**LEFT**] **OUTER JOIN** TABLE_NAME 2

ON < JOIN_CONDITION >;

ORACLE

SELECT COLUMN_NAME

FROM TABLE_NAME 1 , TABLE_NAME 2


WHERE TABLE_NAME 1. COLUMN_NAME = TABLE_NAME 2. COLUMN_NAME (+);

5. Match table and unmached table of boys table

```
SELECT *  
FROM boys LEFT OUTER JOIN girls  
ON boys.gid = girls.gid ;
```

| Bid | Bname | Gid | Gid | Gname |
|-----|--------|-------------|-------------|-------------|
| 1 | Munna | 2 | 1 | Chimi |
| 2 | Chinna | 1 | 2 | Munni |
| 3 | Nibba | 3 | 3 | Nibbi |
| 4 | macha | NULL | NULL | NULL |

RIGHT OUTER JOIN

 To obtain match and unmatched record of right table

SYNTAX:

ANSI

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 [ RIGHT ] OUTER JOIN TABLE_NAME 2  
ON < JOIN_CONDITION >;
```

ORACLE

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 , TABLE_NAME 2  
WHERE TABLE_NAME 1. COLUMN_NAME ( + )= TABLE_NAME 2. CLOUMN_NAME ;
```

| Bid | Bname | Gid | Gid | Gname |
|-------------|-------------|-------------|-----|-------|
| 1 | Munna | 2 | 1 | Chimi |
| 2 | Chinna | 1 | 2 | Munni |
| 3 | Nibba | 3 | 3 | Nibbi |
| NULL | NULL | NULL | 4 | machi |

FULL OUTER JOIN



To obtain both the table matched and unmatched record of both table

SYNTAX:

ANSI

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 [ FULL ] OUTER JOIN TABLE_NAME 2  
ON < JOIN_CONDITION >;
```

6. Match and match record of goals table and boys table

```
SELECT *  
FROM boys FULL OUTER JOIN girls  
ON boys.gid = girls.gid;
```

| Bid | Bname | Gid | Gid | Gname |
|------|--------|------|------|-------|
| 1 | Munna | 2 | 1 | Chimi |
| 2 | Chinna | 1 | 2 | Munni |
| 3 | Nibba | 3 | 3 | Nibbi |
| 4 | macha | NULL | NULL | NULL |
| NULL | NULL | NULL | 4 | machi |

-
7. Employee name and department name if the employee working or not

```
SELECT *  
FROM emp LEFT OUTER JOIN dept  
ON emp. Deptno = dept.deptno;
```

-
8. Employee name and department name if the department have some employee working in it or not

```
SELECT ename, dname  
FROM boys RIGHT OUTER JOIN girls  
ON emp. Deptno = dept.deptno ;
```


-
9. Employee name department name if the employee is not working in any department

```
SELECT *  
  
FROM emp LEFT OUTER JOIN dept  
  
ON emp. Deptno = dept.deptno  
  
WHERE dept.dname IS null.
```

-
10. Employee name and department name if the department does not have any employee working in it

```
SELECT *  
  
FROM emp RIGHT OUTER JOIN dept  
  
ON emp. Deptno = dept.deptno  
  
WHERE emp.empno IS null.
```

-
11. Employee details along with department details irrespective of employees working in a department and irrespective of department contains employee

```
SELECT emp.*, dept.*  
  
FROM emp FULL OUTER JOIN dept  
  
ON emp. Deptno = dept.deptno
```

SELF JOIN



Joining same table it's call self join

Whenever the data present in same table but they are present in different columns

SYNTAX:

ANSI

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 JOIN TABLE_NAME 2  
ON < JOIN_CONDITION >;
```

ORACLE

```
SELECT COLUMN_NAME  
FROM TABLE_NAME 1 , TABLE_NAME 2  
WHERE < JOIN_CONDITION >;
```

Emp

| ENO | ENAME | MGR |
|-----|-------|------|
| 1 | A | 3 |
| 2 | B | NULL |
| 3 | C | 2 |
| 4 | D | 3 |

12. Employee name and manager name from emp table

SUBQUERY

```
SELECT ENAME  
FROM EMP ( not efficient )  
WHERE EMPNO = ( SELECT MGR  
FROM EMP );
```

JOINS

```
SELECT E1.ENAME , E2.ENAME  
FROM EMP "E1", EMP "E2"  
WHERE E1.MGR = E2.ENO ;
```

EMP (employee E1)

EMP (MANAGER E2)

| ENO | ENMAE | MGR | ENO | ENMAE | MGR |
|-----|-------|-----|-----|-------|------|
| 1 | A | 3 | 3 | C | 2 |
| 3 | C | 2 | 2 | B | NULL |
| 4 | D | 3 | 3 | C | 2 |

13. Employee name salary along with manager name salary from emp table

ANSI

```
SELECT E1.ENAME , E1.SAL , E2.NAME, E2.SAL
FROM EMP "E1" JOIN EMP "E2"
ON E1.MGR = E2.ENO ;
```

ORACLE

```
SELECT E1.ENAME , E1.SAL , E2.NAME, E2.SAL
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO ;
```

14. Employee name job salary and commission along with manager name and salary from emp table if the employee getting salary more than 1250

```
SELECT E1.ENAME , E1.SAL E1.JOB, E1.SAL, E1.COM, E2.NAME, E2.SAL
FROM EMP "E1" JOIN EMP "E2"
ON E1.EMGR = E2.ENO AND E1.SAL > 1250 ;
```

15. Employee details along with the manager job if the manager is working a analyst

```
SELECT E1.* , E2.JOB
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO AND E2.JOB = 'ANAYLST' ;
```

16. Employee name and commission along with manager name if the employee is getting commission under the manager is not getting commission

```
SELECT E1.NAME, E1.COM, E2.NAME
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO AND E2.COM IS NOT NULL AND E1.COM IS NULL ;
```

17. Employee name commission higher date along with manager hired if the employee hired after 86 but manager hired before 86

```
SELECT E1.NAME , E1.COM, E1.HIREDATE, E2.HIREDATE
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO AND HIREDATE > 31-DEC-86 AND HIREDATE < 1-JAN-85.
```

18. Employee name higher date along with manager name and higher date if the employee where hired after manager

```
SELECT E1.ENAME, E1.HIREDATE, E2.ENAME, E2.HIREDATE
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO AND E1.HIREDATE > E2.HIREDATE ;
```

19. Employee name job department along with his manager name and job if there employee is working as a clerk or salesman in department number 20 or 30 and manager's working as actual manager

```
SELECT E1.ENAME, E1.JOB, E1.DEPTNO, E2.ENAME, E2.JOB
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO AND E1.JOB IN ('CLERK', 'SALEMAN' )
AND E1.DEPTNO IN ( 20, 30 ) AND E2.JOB = 'MANAGER';
```

20. Employee name along with his manager name

```
SELECT E1.ENAME, E2.ENAME
FROM EMP "E1", EMP "E2"
WHERE E1.MGR = E2.ENO ;
```

21. Employee name and manager name along with employee department name

```
SELECT E1.ENAME, E2.ENAME, D1.DNAME
FROM EMP "E1", EMP "E2", DEPT "D1"          ( self and inner join )
WHERE E1.MGR = E2.ENO AND E1.DEPTNO = D1.DEPTNO;
```

ANSI

```
SELECT E1.ENAME , D1.DNAME, E2.ENAME,
FROM EMP "E1" JOIN EMP "E2"
ON E1.MGR = E2.ENO AND INNER JOIN DEPT "D1"
ON E1.DEPTNO = D1.DEPTNO;
```

22. Employee name and manager name along with manager department name

```
SELECT E1.ENAME, D2.DNAME, E2.ENAME
FROM EMP "E1", EMP "E2", DEPT "D2"
WHERE E1.MGR = E2.MGR AND E2.DEPTNO = D2.DEPTNO;
```

23. Employee name and manager name along with the managers manager names

```
SELECT E1.ENAME , E2.ENAME, E3.ENAME
FROM EMP "E1", EMP "E2", EMP "E3"
WHERE E1.MGR = E2.ENO AND E2.MGR = E3.ENO;
```

24. Employee name management name and managers manager name along with their department name

```
SELECT E1.ENAME , E2.ENAME, E3.ENAME , D1.DNAME, D2.DNAME, D3.DNAME
FROM EMP "E1", EMP "E2", EMP "E3", DEPT "D1", DEPT "D2", DEPT"D3"
WHERE E1.MGR = E2.ENO AND E2.MGR = E3.ENO
AND D1.DEPTNO = E1.DEPTNO AND E2.DEPTNO = D2.DEPTNO
AND D3.DEPTNO = E3.DEPTNO;
```

ANSI

```
SELECT E1.ENAME , E2.ENAME, E3.ENAME, D1.DNAME, D2.DNAME, D3.DNAME
FROM EMP "E1" JOIN EMP "E2"
ON E1.MGR = E2.ENO AND JOIN EMP "E3"
ON E2.MGR = E3.EMPNO INNER JOIN DEPT "D1"
ON E1.DEPTNO = D1.DEPTNO INNER JOIN DEPT "D2"
ON E2. DEPTNO = D2.DEPTNO INNER JOIN DEPT "D3"
ON E3.DEPTNO = D3.DEPTNO.
```

25. Employee name job and is department name along with manager's name job if the employee is working as salesman or clerk in research or accounting department

```
SELECT E1.ENAME , E1.JOB, D1.DNAME, E2.ENAME
FROM EMP "E1", EMP "E2",DEPT "D1"
WHERE E1.MGR = E2.ENO AND E1.DEPTNO = E3.DEPTNO
AND JOB IN ( ' CLERK', 'SALESMAN' ) AND
D1.NAME IN ( 'RESEARCH', 'ACCOUNTING' );
```

26. Number of employee reporting to king

SUBQUERY

```
SELECT COUNT( * )
FROM EMP
WHERE MGR IN ( SELECT EMPNO
FROM EMP
WHERE ENAME = 'KING' );
```

JOINS

```
SELECT COUNT( * )
FROM EMP "E1" , EMP "E2"
WHERE E1.MGR = E2. ENO AND E2.ENAME = 'KING';
```

27. Number of employee working in Chennai

SUBQUERY

```
SELECT COUNT( * )  
FROM EMP  
WHERE DEPTNO IN ( SELECT DEPTNO  
                  FROM DEPT  
                  WHERE LOCATION = 'CHENNAI' );
```

JOINS

```
SELECT COUNT( * )  
FROM EMP "E1" , DEPT "D1"  
WHERE E1.DEPTNO = D1.DEPTNO AND D1.LOC = 'CHENNAI';
```

28. Department name of Arjun's manager

```
SELECT D2.DENAME  
FROM EMP "E1", EMP "E2", DEPT "D2"  
WHERE E1.MGR= E2.EMPNO AND E2.DEPTNO = D2.DEPTNO  
      AND E1.ENAME = 'ARJUN';
```

29. Employee name and salary if they getting same salary

```
SELECT E1.ENAME, E1.SAL  
FROM EMP E1, EMP E2  
WHERE E1.SAL = E2.SAL AND E1.EMPNO != E2.EMPNO ;
```

30. Employee name if the emp having same reporting manager

```
SELECT DISTINCT E1.ENAME, E1.MGR  
FROM EMP "E1", EMP "E2"  
WHERE E1.MGR = E2.EMPNO AND E1.ENO != E2.NO ;
```

FIND 2nd max

SUBQUERY

```
SELECT MAX(SAL)
FROM EMP
WHERE SAL < ( SELECT MAX (SAL )
              FROM EMP );
```

JOINS

```
SELECT MAX ( SAL )
FROM EMP E1, EMP E2
WHERE E1. SAL < E2.SAL ;
```

3rd MAX

JOINS

```
SELECT MAX ( SAL)
FROM EMP E1, EMP E2, EMP E3
WHERE E1.SAL < E2. SAL AND E2.SAL < E3. SAL ;
```