SQL

Introduction of SQL

- SQL stand for Structure Query Language
- It is language which is used to communicate with database
- Developed by Raymond Boyce in 1970
- First name of SQL is SEQUEL (simple English query language)
- 1986 -> change name SEQUEL to SQL

Data

- Data is a row fact which describe attribute of an entity
- Properties are also called as Attribute
- Entity is also called as object
- Example

Height: - 20cm

Color: - green

Capacity: -500ml;

Water

Bottle

Database

- Database is place or medium in which we can store data in organized manner
- Basic operation perform on database are

1] Insert / Create

2] Read / Retrieve

3] Update / Modify

4] Delete / Drop

- This operation are referred to as Crud Operation

DBMS

- It is a software which is used to maintain & manage database
- DBMS Stand for Database management system
- In DBMS software have two main feature

1] security

2] authorization

- In DBMS we store data in file format
- To communicate with DBMS use query language

RDBMS

- RDBMS stand for relational database management system
- It is a software which is used to maintain & manage database
- In RDBMS we can store data in Table (Rows & Columns) format
- To communicate with RDBMS using structure query language
- RDBMS software have Two main feature

1. security
2. authorization

RDBMS	DBMS
RDBMS stand for relational database management system	DBMS Stand for Database management system
It is a software which is used to maintain & manage database	It is a software which is used to maintain & manage database
In RDBMS we can store data in Table (Rows & Columns) format	In DBMS we store data in file format
To communicate with RDBMS using structure query language	To communicate with DBMS use query language

Relational Model

- It is developed by famous Data Scientist E.F.CODD
- It is relational model store data in form of rows and columns
- Any DBMS follows RDBMS then it became Relational model
- Any DBMS follow rule of E.F.CODD

RDBMS

DBMS

Relational model

Table

- Logical organization of data which consist rows and columns is known as Table

Cell

Rows /

Records

Columns / fields

Columns: - Describe the single properties of all entity is known as columns

Rows: - describe the all properties of single entity

Cell: - it is smallest unit of table or intersection between rows and columns generated a space that is called as cell

Rules of E.F. CODD

1. data entered into cell must me always single values data

‘Eid	Ename	PhoneNo
101	Sam	78451230
		1245652321
102	tom	
		4512635210

1. in RDBMS we can store everything in the form of table including metadata

Photo

ImageName

Size

resolution

Eid	Ename	photo
102	Sam	
125	tom	
103	maxi	

Metadata: data about data is known as metadata

Metastable: tables which stores metadata is called as metastable

1. we can store data in multiple tables if needed we can establish connection between tables using key

Need to establish connection

EID	EName
102	tom
105	Sam

Did	Dname
1	BBA
2	BCA

1. data entered into table can verify in two step

2. by using datatype
3. by using constraints

datatypes in SQL

specify the type of data will be stored in particular memory location is known as datatype

types

1. **char**

- used to store alpha-numeric character
- char follows fixed length memory allocation
- character should be enclosed within single quote
- syntax

char(size)

- Example

Char (3)

S A M

Memory used unused memory (memory not free)

1. **varchar(size)**

- used to store character, number & alpha-numeric character
- varchar should be enclosed within single quote
- follow variable length memory location
- syntax

varchar(size)

- example

varchar (3)

S A M

memory used unused memory (memory free)

1. **varchar2(size)**

- updated version of varchar
- store up to 4000 character

1. **date**

- used to store date in particular format
- used date in oracle specified format
- Syntax:

DD-MM-YY -> 12-JAN-24

DD-MM-YYYY -> 25-JAN-2023

1. **Number**

- Used to store numeric value
- Syntax
- Number (precision , [scale])
    1. Precision: - used to determine no of digit used to store integer value max size is 28
    2. Scale: - used to determine no of digit used to store decimal value within precision maximum is 127

1. **large object**

used to store large amount of data

1. character large object [CLOB]: - used to store up to 4GB data

syntax:

CLOB

1. binary large object: - used to store image, video, files in form of binary

syntax:

BLOB

Constraints

- Constraints is rule which is assign to column for validating data
- Following are the type constrains

1. Unique constraints
2. Not null constraints
3. Check constraints
4. Primary key
5. Foreign key

Unique constraints

- It is type of constraints assign to column which should not accept duplicated values
- We can assign more than one unique constraints in one table
- Example

EID	ENAME	SAL	ADDS
101	ram	5 lac	Pune
102	Raju	6 lac	Beed
103	Sam	2 lac	Mumbai
104	Dip	3 lac	Nashik
101	Sandip	4 lac	Jalna

Unique Constraints

Not accept duplicate values

NOT NULL

- It is constrains assign column which should not accept any null value or null cell in table
- In one table we can assign not null with multiple column
- When the not null constraint apply with column then column should not accept any null value
- Example

NOT ACCEPT NULL VALUES

EID	ENAME	SAL	AGE
102		1.2LAC	35
103	RAM	1.5LAC	45
105	DINGA	5LAC	56
	SAM	4LAC	45

EID & ENAME ASSIGN WITH NOT NULL CONSTRAINT COCONSTRAINT

Check constraints

- It is constrains which is used for extra validation along with check constraints we must have to pass condition if condition true it will accept value otherwise reject value
- Give multiple condition in one check
- Also used multiple check constraints in one table
- Example

EID	ENAME	SAL	AGE
1	SOHEL	1 LAC	32
2	ARMAN	5 LAC	152
3	DINGA	2 LAC	45
4	SAM	1.5LAC	-78

NOT INSERTED THIS ROWS

CHECK CONSTRAINTS AGE > 0 AND AGE < 90

Primary key

- Used to identify record uniquely from table assign column with primary key constraint
- When column assign with primary key constraint then in given table we can insert null record, duplicate record
- Characteristics

1. It doesn't allow null values
2. It doesn't allow duplicate values
3. There can be only one primary key in the table
4. It is combination on NOT NULL and UNIQUE constraint
5. It is not mandatory, but highly recommended

- Example

DUPLICATE EID NOT ALLOW

EID	ENAME	SAL	AGE
1	TONY	5	95
2	DANI	2	47
3	MAXI	3	45
1	SAM	4	12

EID ASSIGN WITH PRIMARY KEY

FOREIGN KEY

- It is constraint used to establish connection between table
- The key which is act as primary key in one table and used in another table as reference of this is called as foreign key
- Characteristic

1. There can multiple foreign key in table
2. It allows duplicated values
3. Any attribute become foreign key first it should be primary key in it own table
4. It represents child table but actually it belongs to parent table
5. It is also called as referential integrity constraints

- Example

JOING TWO TABLE

EID	ENAME	SAL	AGE	DEPTNO
101	ABC	540	45	10
102	PQR	140	23	10
103	MNO	250	30	30

DEPTNO	DNAME	TOTALEMP
20	D1	10
30	D2	05
40	D3	03

PRIMARY KEY

PRIMARY KEY

FOREIGN KEY

Statement

- Statement are used to perform CRUD operation on database
- Types of statement
    1. DDL (Data definition language)
    2. DML (Data manipulation language)
    3. TCL (Transaction control language)
    4. DCL (Data control language)
    5. DQL (Data query language)

DQL (Data query language)

- It is statement which is used to retrieve data from database

1. Select: - it is statement which is used to retrieve data from table & display it
2. Projection: - it is statement which used to retrieve data from table by selecting one column
3. Selection: - it is a statement which is used to retrieve data from table by selecting both rows and column
4. Joins: - it is used to retrieve data from multiple table simultaneously

Projection

- Syntax

Select * / [Distinct] column_name / expression [Alise] from table_name;

- Order of execution

From

Select

- Example

1. WAQTD name & empno of emp;

SELECT ENAME

FROM EMP;

1. WAQTD NAME, SAL, EMPNO FROM EMP;

SELECT ENAME, SAL, EMPNO

FROM EMP;

Expression

- Combination of operand and operator is known as expression
- Statement which give result is called as expression
- Example

1] WAQTD NAME & ANNUAL_SAL OF EMP

SELECT ENAME, SAL * 12

FROM EMP;

2] WAQTD NAME, SALARY, OF EMP ALONG WITH BOUNS OF 2000 IN SALARY

SELECT ENAME, SAL, SAL+200

FROM EMP;

3] WAQTD NAME, SALARY OF EMP WITH 10% HIKE

SELECT ENAME, SAL, SAL+SAL*0.1

FROM EMP;

SELECT ENAME, SAL, SAL + SAL/100*10

FROM EMP;

Alias

- It is alternative names given to column or expression in result table
- Alias name we can used with or without as keyword
- Alias name we should assign in select clause
- If multiple words are these use underscore or quotes to give alias name

1. Select sal*12 as annualsalary from emp;
2. Select sal*12 annualSalary from emp;
3. Select sal*12 from annual_Salary from emp;
4. Select sal*12 from “annual salary” from emp;

1] WAQTD ANNUAL SALARY ALONG WITH 10 % HIKE

SELECT SAL*12 AS ANNUAL_SALARY, SAL+SAL*0.1 HIKE_SALARY

FROM EMP;

DISTINCT

- It is keyword which is used to remove duplicated value in result table
- Distinct keyword we have to assign as first argument in select clause

- Along with distinct we have to pass column_name as argument

1] WAQTD EMP DETAILS WHERE SALARY WILL NOT BE REPEATED

SELECT DISTINCT SAL

FROM EMP;

2] WAQTD EMP DETAILS WHERE SALARY AND DEPTNO WILL NOT BE REPEATED

SELECT DISTINCT SAL ,DEPTNO

FROM EMP;

Selection

- Used to retrieve data from table from selecting both rows and columns
- Order of execution is

1. From
2. Where
3. select

- syntax

SELECT * /[DISTINCT] COL_NAME/EXPRESSION [ALIAS] FROM TABLE_NAME WHERE <CONDITION>

WHERE

- WHERE IS CLAUSE USED TO FILTER THE RECORDS
- IT EXECUTES AFTER THE EXECUTION OF THE FROM CLAUSE
- It executes row by row
- In where clause we pass the condition and we can also pass multiple condition using logical operator

1] WAQTD DETAILS OF EMPLOYEES WHO ARE EARNING MORE THAN 2000

SELECT *

FROM EMP

WHERE SAL > 2000;

2] WAQTD NAME OF THE EMPLOYEES WHO ARE WORKING AS ANALYST

SELECT ENAME

FROM EMP

WHERE JOB = ‘ANALYST’;

3] WAQTD DETAILS OF EMPLOYEES ALONG WITH ANNUAL SALARY IF EMPLOYESS ARE WORKING IN DEPTNO 20

SELECT EMP.*, SAL*12 ANNUAL_SALARY

FROM EMP

WHERE DEPTNO=20;

4] WAQTD NAME OF EMPLOYEES HIRE BEFORE 1982

SELECT *

FROM EMP

WHERE HIREDATE < ’01-JAN-1982’;

5] WAQTD DETAILS OF EMPLOYEES ALONG WITH 10% HIKE AND ANNUAL SALARY IF EMPLOYEES ARE WORKING IN DEPTNO 20

SELECT EMP.*, SAL*12 ANNUAL_SALARY, SAL+SAL*0.1 HIKE_SALARY

FROM EMP

WHERE DEPTNO=20;

OPERATOR

- Operator is special symbol used to perform specific task

- Types
- Arithmetic operator ===> +, -, *, /
- Relational operator ===> (>, <, >=, <=)
- Comparison operator ===> [=,!=, <>]
- Concatenation operator ===> ||
- Logical operator ===> AND, OR, NOT
- Special operator ===> IN, NOT IN, BETWEEN, NOT BETWEEN, IS, NOT BETWEEN, IS, NOT IS, LIKE, NOT LIKE
- Subquery operator ===> ALL, ANY, EXITS, NOT EXITS

Concatenation operator

- Used to merge given two string
- Syntax

Select 'string1' || 'string2' from table name

1]WAQTD ENAME PREFIX WITH MR STRING FROM EMPLOYESS TABLE

SELECT 'MR.' || ENAME

FROM EMP;

2] WAQTD EMP NAME WITH CREATE EMAIL FOR EACH EMP

SELECT ENAME, ENAME || '@GMAIL.COM' AS EMAIL

FROM EMP;

Logical operator

1] AND: = used whenever we have to satisfy all condition

1] WAQTD NAME OF EMPLOYEE EARNING MORE THAN 2000 AND WORKING IN DEPTNO 20

SELECT ENAME

FROM EMP

WHERE SAL > 2000 AND DEPTNO = 20;

2] WAQTD DETAILS OF EMPLOYEE WORKING AS ANALYST AND EARNING MORE THAN 2000

SELECT ENAME

FROM EMP

WHERE SAL > 2000 AND JOB = 'ANALYST';

3] WAQTD NAME OF AN EMPLOYEE EARNING MORE THAN 2000 AND WORKING IN DEPTNO 10 AND WORKING AS CLERK

SELECT ENAME

FROM EMP

WHERE SAL > 2000 AND DEPTNO=20 AND JOB = 'CLERK';

2] OR operator

- Used whenever we have satisfied any one of condition

1] WAQTD NAME OF EMP WHERE EMP WORKING AS ANALYST OR CLERK

SELECT ENAME

FROM EMP

WHERE JOB = 'ANAYLST' OR JOB = 'CLERK';

3] NOT OPERATOR

- Used whenever we have to inverse conditions

1] WAQTD NAME OF EMP EXCEPTBEMP WORKING IN DEPTNO 10

SELECT ENAME

FROM EMP

WHERE NOT DEPTNO=10;

Special operator

1] IN

- It is a multivalued operator which accept multiple values at R.H.S
- Syntax:

Col_name IN (v1,v2,v3…..vn);

1] WAQTD DETAILS OF EMP ALONG WITH ANNUAL-SAL IF EMP ARE HIRED AFTER 81 & WORKING IN DEPTNO 10 OR 20 AS AN ANALYST OR CLECK

SELECT EMP.*, SAL * 12 ANNUAL_SAL

FROM EMP

WHERE HIREDATE > '31-DEC-1981' AND DEPTNO IN(10,20) AND JOB IN ('ANALYST' , 'CLERK');

NOT IT OPERATOR

- It is same like as in operator but instead of accepting value it reject values at RHS
- Syntax

Col_name NOT IN (V1,V2,V3……..VN);

1] WAQTD DETAILS OF EMP IF THEY ARE WORKING IN DEPTNO 10,20,30 EXCEPT EMP WHO ARE WORKING AS SALESMAN & HIRED IN YEAR 81

SELECT *

FROM EMP

WHERE DEPTNO IN(10,20,30) AND JOB NOT IN ('SALESMAN') AND HIREDATE = '31-DEC'1981';

BETWEEN OPERATOR

- Used to compare values present in range
- It works by including range
- Syntax

Col_name BETWEEN lower_range AND upper_range;

1] WAQTD NAME OF EMP ALONG WITH JOB IF EMP ARE EARNING MORE THAN 2000 AND LESS THAN 6000 & HIRED AFTER 1982 AND BEFORE 1988

SELECT ENAME , JOB

FROM EMP

WHERE SAL BETWEEN 2001 AND 5999 AND HIREDATE BETWEEN '01-JAN-1983' AND '31-DEC-1987';

NOT BETWEEN

- It is same like between operator but instead of accepting range it will reject range
- Syntax:

Col_name BETWEEN 'lower_range' AND 'higher range';

1] WAQTD NAME & JOB OF EMP EXCEPT EMP WORKING AS SALESMAN AND WORKING DEPTNO 10 OR 20 OR 30 EXCEPT THOSE EMP WHO ARE EARNING IN RANGE 1000 TO 3000

SELECT ENAME, JOB

FROM EMP

WHERE JON NOT IN('SALESMAN') AND DEPTNO IN(10,20,30) AND SAL NOT BETWEEN 1000 AND 3000;

ASSIGNMENT 1:

1] WAQTD THE ANNUAL SALARY OF THE EMPLOYEE WHOSE NAME IS SMITH.

SELECT SAL * 12 ANNUAL_SALARY

FROM EMP

WHERE ENAME='SMITH'

2] WAQTD NAME OF THE EMPLOYEE WORKING AS CLERK.

SELECT ENAME

FROM EMP

WHERE JOB = 'CLERK';

3] WAQTD SALARY OF THE EMPLOYEE WORKING AS SALESMAN.

SELECT SAL

FROM EMP

WHERE JOB = 'SALESMAN';

4] WAQTD DETAILS OF THE EMPLOYEES WHO EARNS MPRE THAN 2000.

SELECT *

FROM EMP

WHERE SAL > 2000;

5] WAQTD DETAILS OF THE EMPLOYEE WHOSE NAME IS JONES.

SELECT *

FROM EMP

WHERE ENAME ='JONES';

6] WAQTD DETAILS OF THE EMPLOYEE WHO HIRED AFTER ‘1-JAN 81’.

SELECT *

FROM EMP

WHERE HIREDATE > '1-JAN-1981'

7] WAQTD DIFFERENT SALARIES OF THE EMPLOYEES.

SELECT DISTINCT SAL FROM EMP;

8] WAQTD NAME AND SALARY ALONG WITH ANNUAL SALARY IF ANNUAL SALARY IS MORE THAN 12000.

SELECT ENAME,SAL,SAL*12

FROM EMP

WHERE SAL*12 > 12000

9] WAQTD EMPNO OF THE EMPLOYEES WORKING IN DEPT NO 30.

SELECT EMPNO

FROM EMP

WHERE DEPTNO = 30;

10] WAQTD ENAME AND HIREDATE IF THEY HIRED BEFORE 1981.

SELECT EMPNO

FROM EMP

WHERE HIREDATE < '01-JAN-1981';

11] WAQTD DETAILS OF THE EMPLOYEES WORKING AS MANAGER.

SELECT * FROM

EMP WHERE JOB = 'MANAGER';

12] WAQTD NAME & SAL GIVEN TO EMPLOYEE IF EMPLOYEE EARNS A COMMISION OF RUPEES 1400.

```
SELECT ENAME,SAL
FROM EMP
WHERE COMM> 1400;
```

13] WAQTD DETAILS OF EMPLOYEE HAVING COMMISSION MORE THAN SAL.

```
SELECT * FROM
EMP WHERE SAL < COMM;
```

14] WAQTD EMPNO OF THE EMPLOYEES HIRED BEFORE THE YEAR 87.

```
SELECT *
FROM EMP
WHERE HIREDATE < '01-JAN-1987';
```

15] WAQTD DETAILS OF TH EMPLOYEES WORKING AS AN ANALYST.

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

16] WAQTD DETAILS OF THE EMPLOYEES EARNING MORE THAN 2000 RUPESS PER MONTH.

```
SELECT *
FROM EMP
WHERE SAL > 2000
```

1. IN Operator

1. Retrieve employees who work in departments **10, 20, or 30**.

```
SELECT *
FROM EMP
WHERE DEPTNO IN(10,20,30)
```

1. Find employees whose job is either **'CLERK', 'ANALYST', or 'SALESMAN'**.

```
SELECT *
FROM EMP
WHERE JOB IN('CLERK','ANALYST','SALESMAN');
```

1. Get employees whose manager is **7566, 7698, or 7839**.

```
SELECT *
FROM EMP
WHERE MGR IN (7566,7698,7839);
```

1. Retrieve employees who were hired in **'FEB-81', 'SEP-81', or 'DEC-81'**.

```
SELECT *
FROM EMP
WHERE HIREDATE IN('01-FEB-1981','01-SEP-1981','01-DEC-1981');
```

1. Select employees whose salaries are among **800, 1100, 1600, or 2850**.

```
SELECT *
FROM EMP
```

WHERE SAL IN(800,1100,1600,2850);

1. Find employees whose commission (COMM) is **0, 300, or 500**.

SELECT *

FROM EMP

WHERE COMM IN(300,500,0);

2. NOT IN Operator

1. Find employees who **do not** work in departments **10, 20, or 30**.

SELECT *

FROM EMP

WHERE DEPTNO NOT IN(10,20,30);

1. Retrieve employees whose job is **not** 'CLERK', 'MANAGER', or 'ANALYST'.

SELECT *

FROM EMP

WHERE JOB NOT IN('CLERK','ANALYST','SALESMAN');

1. Get employees who **do not** have manager ID **7566, 7698, or 7839**.

SELECT *

FROM EMP

WHERE SAL NOT IN(800,1100,1600,2850);

1. Find employees whose salary is **not** 800, 1100, 1600, or 2850.

SELECT *

FROM EMP

WHERE SAL NOT IN(800,1100,1600,2850);

1. Retrieve employees who **do not** receive a commission (COMM).

SELECT *

FROM EMP

WHERE COMM NOT IN(300,500,0,1400);

3. BETWEEN Operator

1. Retrieve employees whose salary is **between 1000 and 3000**.

SELECT *

FROM EMP

WHERE SAL BETWEEN 1001 AND 2999;

1. Find employees who were hired **between '01-JAN-81' and '31-DEC-81'**.

SELECT *

FROM EMP

WHERE HIREDATE BETWEEN '02-JAN-1981' AND '30-DEC-1981';

1. Get employees whose commission (COMM) is **between 200 and 1000**.

SELECT *

FROM EMP

WHERE COMM BETWEEN 201 AND 999;

1. Select employees whose EMPNO falls **between 7500 and 7900**.

SELECT *

FROM EMP

WHERE EMPNO BETWEEN 7501 AND 7699;

4. NOT BETWEEN Operator

1. Retrieve employees whose salary is **not** between 1000 and 3000.

SELECT *

FROM EMP

WHERE SAL NOT BETWEEN 1001 AND 2999;

1. Find employees who were **not** hired between '01-JAN-81' and '31-DEC-81'.

SELECT *

FROM EMP

WHERE HIREDATE NOT BETWEEN '02-JAN-1981' AND '30-DEC-1981';

1. Get employees whose commission (COMM) is **not** between 200 and 1000.

SELECT *

FROM EMP

WHERE COMM NOT BETWEEN 201 AND 999;

1. Select employees whose EMPNO is **not** between 7500 and 7900.

SELECT *

FROM EMP

WHERE EMPNO NOT BETWEEN 7501 AND 7699;

IS operator

- Used to compare NULL value
- Syntax

Col_name/expression is NULL;

1] WAQTD TO DISPLAY ENAME, COMM WHERE EMP HAVE NO COMMISION

SELECT ENAME, COMM

FROM EMP

WHERE COMM IS NULL;

IS NOT

- Used to compare with not null;
- Syntax

Col_name is not null;

1] WAQTD DETAILS OF EMP WHO ARE EARNING SOME COMMISSION

SELECT *

FROM EMP

WHERE COMM IS NOT NULL;

LIKE

- LIKE IS AN OPERATOR WHICH IS USED TO MATCH THE PATTERN
- To achieve pattern, we have 2 special character such as a

1] %

2] _

- Syntax

Col_name / expression like ‘pattern_to_match’;

1] %

- It can accept any no of character any no of time and no character
- Example
    1. Start with character A: ‘A%’
    2. Ends with character A: ‘%A’
    3. Has Char A: ‘%A%’
    4. Has character a present twice: ‘%A%A%A’
    5. Has two consecutive A: ‘%AA%’
    6. Start with A and ends with N: ‘A%N’

2] _ (underscore)

- It can accept any character but only one character
- Example

1. Second character is A: ‘_A%’
2. Third character is A: ‘__A%’
3. Second last character is A: ‘%A_’
4. Third last character is A: ‘%A___’
5. Has 5 character: ‘_____’

Questions

1] WAQTD NAMES OF THE EMPLOYEES IF THE EMPLOYEES’ SALARY’S LAST 2 DIGIT IS 50 RUPEES

SELECT *

FROM EMP

WHERE SAL LIKE ‘%50’;

2] WAQTD DETAILS OF THE EMPLOYEES WHO HIRED IN NOVEMBER.

SELECT *

FROM EMP

WHERE HIREDATE LIKE ‘%NOV%’;

3] WAQTD DETAILS OF THE EMPLOYEES WHOSE NAME THIRD LAST CHARACTER IS N AND HIRED IN THE YEAR OF 81.

SELECT *

FROM EMP

WHERE HIREDATE LIKE ‘%N__’ AND HIREDATE BETWEEN ’01-JAN-1981’ AND

’31-DEC-1981’;

4] WAQTD DETAILS OF THE SALESMEN WHOSE NAME FIRST CHARACTER IS A AND HIRED ON FEBRUARY.

SELECT *

FROM EMP

WHERE JOB = ‘SALESMAN’ AND ENAME LIKE ‘A%’ AND HIREDATE LIKE ‘%FEB%’;

NOT LIKE

It is similar to like operator instead of selecting the pattern it will reject those pattern

SYNTAX:

COLUMN_NAME/EXPRESSION NOT LIKE ‘PATTERN_TO_MATCH’;

ASSIGNMENT 2

- List all the employees whose commission is null.

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

- List all the employees who don't have any reporting manager.

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

- List all the salesmen in deptno.30 and having salary greater than 1500.

```
SELECT *
FROM EMP
WHERE DEPTNO = 30 AND SAL > 1500;
```

- List all the employees whose name start with character S or A.

```
SELECT *
FROM EMP
WHERE ENAME LIKE 'A%' AND ENAME LIKE 'S%';
```

- List all the employees except those who are working in deptno 10 or 20.

```
SELECT *
FROM EMP
WHERE NOT IN(10,20);
```

- List all the employees who are having reporting manager in deptno 10.

```
SELECT *
FROM EMP
WHERE MGR IS NOT NULL AND DEPTNO = 10;
```

- List all the employees whose commission is null and working as a clerk.

```
SELECT *
FROM EMP
WHERE COMM IS NULL AND JOB = 'CLERK';
```

- List all the employees who don't have a reporting manager(mgr.) In deptno 10 or deptno 30.

```
SELECT *
FROM EMP
WHERE MGR IS NOT NULL AND DEPTNO IN(10,20);
```

- List all the details of employee along with annual salary except those who are working in deptno 30.

```
SELECT EMP. *,SAL*12 ANNUAL_SALARY
FROM EMP
WHERE DEPTNO NOT IN(10);
```

- List all the employees who are having reporting manager(mgr.) in deptno 10 along with 10% hike.

```
SELECT EMP.*,SAL+SAL*0.1 HIKESALARY
FROM EMP
```

WHERE MGR IS NOT NULL AND DEPTNO = 20;

- List all the employees who are joined in February.

SELECT *

FROM EMP

WHERE DATE LIKE '%FEB%';

- List all the employees whose name start with 'S' or 'J'

SELECT *

FROM EMP

WHERE ENAME LIKE 'S%' AND ENAME LIKE 'J%';

- Display all the employees who are salesmen having 'E' as last character in their ename and salary having exactly 4 character.

SELECT *

FROM EMP

WHERE JOB IN ('SALESMAN') AND ENAME LIKE '%E%' AND SAL LIKE '____';

FUNCTIONS

- FUNCTION IS BLOCK OF CODE OR SET OF NSTRUCTION USED TO EXECUTE SPECIFIC TASK
- THERE ARE TWO TYPE OF FUNCTION

1. Pre-define function
2. User defined function

Built in functions

- The functions which has already define in SQL is called as built in function
- It has future divided in 2 type

1] single row function

2] multi-row function

Single row function

- Execute row by row
- It takes one input executes & generate one output & goes to next input
- If we pass 'n' no of input it return 'n' no of output
- Ex

outputs

inputs

Single

Row

function

Multi row function

- It is also known as aggregate / group function
- It executes row by row
- It takes all input at once aggregate it & generate single output
- If we pass 'n' no of inputs it will return single output
- List of MRF functions

-
    1. MAX(): - used to obtain maximum value present in column
    2. MIN():- used to obtain minimum value present in column
    3. SUM():- used to obtain summation of value present in column
    4. AVG():- used to obtain average value present in column
    5. COUNT():- used to obtain no of values in column

Characteristics of multi row functions

1. In multi row function we can pass only one argument
2. Along with multi row function we cannot pass any other column name in select clause
3. It ignores null
4. In where clause we cannot pass multi row function
5. Sum() and avg() accept only number type as an arguments
6. Count is only multi row function which accept * as an argument

Examples

1. WAQTD MAXIMUM SALARY GIVEN TO A MANAGER.

```
SELECT MAX(SAL)

FROM EMP

WHERE JOB IN('MANAGER');
```

1. WAQTD TOTAL SALARY GIVEN TO A DEPTNO 10.

```
SELECT SUM(SAL)

FROM EMP

WHERE DEPTNO=10;
```

1. . WAQTD NUMBER OF EMPLOYEES EARNING MORE THAN 1500 IN DEPTNO 20.

```
SELECT COUNT(*)

FROM EMP

WHERE DEPTNO = 20 AND SAL > 1500;
```

1. WAQTD NUMBER OF EMPLOYEES HAVING 'E' IN THEIR NAMES.

```
SELECT COUNT(*)

FROM EMP

WHERE ENAME LIKE '%E%';
```

1. WAQTD MINIMUM SALARY GIVEN TO THE EMPLOYEES WORKING AS A CLERK IN DEPTNO 10 OR DEPTNO 20.

```
SELECT MIN(SAL)

FROM EMP

WHERE DEPTNO IN(10,20) AND JOB = 'CLERK';
```

1. WAQTD NUMBER OF EMPLOYEES HIRED AFTER 1982 AND BEFORE 1985 IN TO DEPT NO 10 OR 30.

```
SELECT COUNT(*)

FROM EMP

WHERE HIREDATE > '01-JAN-1983' AND HIREDATE < '31-DEC-1984';
```

1. WAQTD NUMBER OF EMPLOYEES WHO ARE GETTING COMMISSION.

```
SELECT COUNT(*)

FROM EMP

WHERE COMM IS NOT NULL;
```

1. WAQTD NUMBER OF UNIQUE SALARIES PRESENT IN EMPLOYEE TABLE.

```
SELECT COUNT(DISTINCT SAL)

FROM EMP ;
```

1. WAQTD AVERAGE SALARY GIVEN TO THE CLERK.

```
SELECT AVG(SAL)

FROM EMP
```

WHERE JOB= ‘CLERK’;

1. WAQTD AVERAGE SALARY NEEDED TO PAY ALL THE EMPLOYEES.

SELECT AVG(SAL)

FROM EMP ;

GROUP BY

- It is used to group the records
- Group by clause executes row by row
- After execution of group by clause we are getting groups
- Group by clause can be used with or without where clause
- Column name used for grouping can be written in select clause will be called as group by expression
- Order of execution

1] from

2] where ---→ optional

3] group by

4] select

- Syntax

Select group function / expression

From table_name

Where <filter_condition>

Group by column_name/expression;

Example

WAQTD NUMBER OF EMPLOYEES WORKING IN EACH DEPARTMENT.

SELECT COUNT(*),DEPTNO

FROM EMP

GROUP BY DEPTNO;

WAQTD number of employees working in each department except the Employee working as analyst.

SELECT COUNT(*)

FROM EMP

WHERE JOB <> ‘ANALYST’

GROUP BY DEPTNO;

WAQTD maximum salary given to each job.

SELECT MAX(SAL),JOB

FROM EMP

GROUP BY JOB;

WAQTD number of employees working in each job if the employees Have character 'A' in their names.

SELECT COUNT(*) , JOB

FROM EMP

WHERE ENAME LIKE ‘%A%’

GROUP BY JOB;

WAQTD number of employees getting commission in each department

SELECT COUNT(*), DEPTNO

FROM EMP

WHERE COMM IS NOT NULL

GROUP BY DEPTNO;

HAVING CLAUSE

- Having clause is used to filter the group
- It will execute after the execution of group by clause
- It executes group by group
- We can pass multi-row function in having clause
- To use having clause group by clause is mandatory
- We can pass multiple condition in having clause by using logical operators
- Order of execution

From

Where === > optional

Group by

Having

Select

1] WAQTD MAXIMUM SAL OF EMP WORKIING IN EACH DEPARTMENT IF THEIR MAXIMUM SALARY IS MORE THAN 3450

SELECT MAX(SAL),DEPTNO

FROM EMP

GROUP BY DEPTNO

HAVING MAX(SAL) > 3450;

2] WAQTD THE DESIGNATION IN WHICH THERE ARE ATLEAST 2 EMPLOYEES PRESENT.

SELECT COUNT(*) .DEPTNO

FROM EMP

GROUP BY DEPTNO

HAVING COUNT(*) >=2;

3] WAQTD THE NAMES THAT ARE REPEATED.

SELECT COUNT(*)

FROM EMP

GROUP BY ENAME

HAVING COUNT(*) > 1;

4] WAQTD JOB AND TOTAL SALARY OF EACH JOB IF THE TOTAL SALARY OF EACH JOB IS GREATER THAN 3450.

SELECT SUM(SAL) , SAL

FROM EMP

GROUP BY SUM(SAL)

HAVING SUM(SAL) > 3450;

5] WAQTD THE NAMES THAT ARE REPEATED EXACTLY TWICE.

SELECT COUNT(*)

FROM EMP

GROUP BY ENAME

HAVING COUNT(*) = 2;

WHERE	HAVING

GROUP BY CLAUSE IS NO MANDATARY	GROUP BY CLAUSE IS NO MANDATARY
USED TO FILTER THE RECORD	USED TO FILTER THE GROUP
EXECUTE ROW BY ROW	EXECUTE GROUP BY GROUP
IN WHERE WE CAN NOT USE MULTI-ROW FUNCTION	IN HAVING WE CAN USE MULTI-ROW FUNCTION
EXECUTE BEFORE GROUP BY CLAUSE	EXECUTE AFTER THE GROUP BY CLAUSE

ORDER BY

- Order by clause used to sort the record in ascending or descending order
- It must be written as last clause in statement
- We can use alise name in the in order by clause
- It will be executing after the select statement
- we can pass column name or expression as an argument in order by clause
- by default, order by clause sort record in ascending order
- syntax

SELECT COLUMN_NAME/EXPRESSION

FROM TABLE_NAME

WHERE <CONDITIO>

GROUP BY COLUMN_NAME/EXPRESSION

HAVING <GROUP_CONDITION>

ORDER BY [ASC]/DESC;

OR

SELECT COLUMN_NAME/EXPRESSION

FROM TABLE_NAME

ORDER BY [ASC]/DESC;

- FLOW OF EXECUTION

From

Where

Group by -----> optional

Having -------- > optional

Select

Order by

Q1] WAQTD SALARY OF EMP IN ARRAING ORDER

SELECT SAL

FROM EMP

ORDER BY SAL;

Q2] WAQTD NAME, SAL OF EMP IN DESCENDING ORDER

SELECT ENAME, SAL

FROM EMP

ORDER BY DESC;

SUBQUERY

- Query inside query is called as subquery
- The query which contain other query or subquery is called as outer query or parent query
- Query which is inside the query is called as inner query
- The result of inner query is used as input to outer query

Outer query

input

Inner query

- Syntax

SELECT COLUMN_NAME/EXPRESS

FROM TABLE_NAME

WHERE <CONDITION> (SELECT COLUMN_NAME

FROM TABLE_NAME

WHERE <CONSITION>);

- Sub query has two cases

Case 1: = whenever there is unknown present we go for subquery

Q1] WAQTD SAL, JOB OF EMP WHO ARE IN SAME DEPTMENT AS JONES

SELECT SAL, JOB

FROM EMP

WHERE DEPTNO IN (SELECT DEPTNO

FROM EMP

WHERE ENAME = 'JONES');

Q2] WAQTD SAL, ENAME, JOB AND ANNUAL_SAL IF THE ANNUAL SALLARY IS GREATER THAN TURNER

SELECT ENAME, SAL, JOB, SAL*12

FROM EMP

WHERE SAL*12 > (SELECT SAL*12

FROM EMP

WHERE ENAME = 'TURNER');

SUBQUERY CASE 1 Assignment

1. WAQTD NAME OF THE EMPLOYEES WHO ARE EARNING MORE THAN ADAMS.

SELECT ENAME

FROM EMP

WHERE SAL >(SELECT SAL

FROM EMP

WHERE ENAME = 'ADAMS');

1. WAQTD NAME AND SALARY OF THE EMPLOYEES EARNING LESS THAN KING.

SELECT ENAME, SAL

FROM EMP

WHERE SAL <(SELECT SAL

FROM EMP

WHERE ENAME = 'KING');

1. WAQTD NAME AND DEPTNO OF THE EMPLOYEE IF THEY ARE WORKING IN SAME DEPT AS JONES.

SELECT DEPTNO, ENAME

FROM EMP

WHERE DEPTNO IN (SELECT DEPTNO

```
FROM EMP

WHERE ENAME = 'JONES');
```

1. WAQTD NAME AND JOB OF THE EMPLOYEES WORKING IN THE SAME DESIGNATION AS JAMES.

```
SELECT ENAME, JOB

FROM EMP

WHERE JOB IN (SELECT JOB

FROM EMP

WHERE ENAME = 'JAMES');
```

1. WAQTD EMPNO AND ENAME ALONG WITH ANNUAL SALARY OF ALL THE EMPLOYEES IF THEIR ANNUAL SALARY IS GREATER THAN THE WARD'S ANNUAL SALARY.

```
SELECT ENAME, EMPNO, SAL*12

FROM EMP

WHERE SAL*12 > (SELECT SAL*12

FROM EMP

WHERE ENAME = 'WARD');
```

1. WAQTD NAME AND HIREDATE OF THE EMPLOYEES IF THEY ARE HIRED BEFORE SCOTT

```
SELECT ENAME, HIREDATE

FROM EMP

WHERE HIREDATE < (SELECT HIREDATE

FROM EMP

WHERE ENAME = 'SCOTT');
```

1. WAQTD NAME AND HIREDATE OF THE EMPLOYEES IF THEY HIRED AFTER PRESIDENT.

```
SELECT ENAME, HIREDATE

FROM EMP

WHERE HIREDATE > (SELECT HIREDATE

FROM EMP

WHERE JOB = 'PRESIDENT');
```

1. WAQTD NAME AND SAL OF THE EMPLOYEES IF THEY ARE EARNING SAL LESS THAN THE EMPLOYEES WHOSE EMPNO IS 7839.

```
SELECT ENAME, SAL

FROM EMP

WHERE SAL < (SELECT SAL

FROM EMP

WHERE EMPNO = 7839);
```

1. WAQTD ALL THE DETAILS OF THE EMPLOYEES IF THE EMPLOYEES ARE HIRED BEFORE MILLER.

```
SELECT *

FROM EMP

WHERE HIREDATE < (SELECT HIREDATE

FROM EMP

WHERE ENAME = 'MILLER');
```

1. WAQTD ENAME AND EMPNO OF THE EMPLOYEES IF EMPLOYEES ARE EARNING MORE THAN ALLEN.

```
SELECT ENAME, EMPNO
FROM EMP
WHERE SAL > (SELECT SAL
FROM EMP
WHERE ENAME = ‘ALLEN’);
```

1. WAQTD ENAME AND SALARY OF ALL THE EMPLOYEES WHO ARE EARNING MORE THAN MILLER AND LESS THAN ALLEN

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL BETWEEN(SELECT SAL+1
FROM EMP
WHERE ENAME=’MILLER’) AND
(SELECT SAL-1
FROM EMP
WHERE ENAME=’ALLEN’)
```

1. WAQTD ALL THE DETAILS OF THE EMPLOYEES WORKING IN DEPTNO 20 AND WORKING IN THE SAME DESIGNATION AS SMITH.

```
SELECT *
FROM EMP
WHERE DEPTNO=20 AND JOB IN (SELECT JOB
FROM EMP
WHERE ENAME = ‘SMITH’);
```

1. WAQTD ALL THE DETAILS OF THE EMPLOYEES WORKING AS MANAGER IN THE SAME DEPT AS TURNER.

```
SELECT *
FROM EMP
WHERE JOB =’MANAGER’ AND DEPTNO IN(SELECT DEPTNO
FROM EMP
WHERE ENAME=’TURNER’);
```

1. WAQTD NAME AND HIREDATE OF THE EMPLOYEES HIRED AFTER 1980 AND BEFORE KING.

```
SELECT ENAME, HIREDATE
FROM EMP
WHERE HIREDATE > ’31-DEC-1980’
AND HIREDATE < (SELECT HIREDATE
FROM EMP
WHERE ENAME = ‘KING’);
```

1. WAQTD ALL THE DETAILS OF THE EMPLOYEES WHO EARN MORE THAN SCOTT BUT LESS THAN THE KING.

```
SELECT SAL
FROM EMP
WHERE SAL BETWEEN (SELECT SAL+1
FROM EMP
```

WHERE ENAME='SCOTT')

AND (SELECT SAL-1

FROM EMP

WHERE ENAME='KING');

1. WAQTD NAME OF THE EMPLOYEES WHOSENAME START'S WITH 'A' AND WORKING IN THE SAME DEPT AS BLAKE.

SELECT ENAME

FROM EMP

WHERE ENAME LIKE 'A%' AND DEPTNO IN(SELECT DEPTNO

FROM EMP

WHERE ENAME = 'BLAKE');

1. WAQTD NAME AND COMM IF EMPLOYEES EARN THE COMMISSION AND WORK IN THE SAME DESIGNATION AS SMITH.

SELECT ENAME,COMM

FROM EMP

WHERE COMM IS NOT NULL AND JOB IN(SELECT JOB

FROM EMP

WHERE ENAME='SMITH');

1. WAQTD DETAILS OF THE EMPLOYEES WORKING AS CLERK IN THE SAME DEPT AS TURNER.

SELECT *

FROM EMP

WHERE JOB IN('CLERK') AND

DEPTNO IN (SELECT DEPTNO

FROM EMP

WHERE ENAME='TURNER');

WAQTD ENAME ,SAL,DESIGNATION OF THE EMPLOYEES WHOSE ANNUAL SALARY IS MORE THAN SMITH AND LESS THAN KING

SELECT ENAME, SAL

FROM EMP

WHERE SAL*12 BETWEEN (SELECT SAL*12+1

FROM EMP

WHERE ENAME = 'SMITH') AND (SELECT SAL*12-1

FROM EMP

WHERE ENAME = 'KING');

CASE 2: whenever data to be selected & condition to be executed are on 2 different table

Q1] WAQTD NAME OF EMP WHO ARE WORKING IN CHICAGO

SELECT *

FROM EMP

WHERE DEPTNO IN (SELECT DEPTNO

FROM DEPT

WHERE LOC = 'CHICAGO');

Q2] WAQTD THE DNAME WHERE IN ENAME LL IN PRESENT TWICE

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE ENAME LIKE '%LL%');
```

Q3] WAQTD DETAILS OF EMP WHERE IN DEPARTMENT NAME S IS PRESENT AT THE END

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
FROM DEPT
WHERE DNAME LIKE '%S');
```

Q4] WAQTD DETAILS OF EMP WHICH HIREDATE SAME AS FORD

```
SELECT *
FROM EMP
WHERE HIREDATE IN(SELECT HIREDATE
FROM EMP
WHERE ENAME = 'FORD');
```

Q5] WAQTD DETAILS OF EMP WHERE WORKING AS MANAGER AND LOC IS CHICAGO

```
SELECT *
FROM EMP
WHERE JOB = 'MANAGER' AND DEPTNO IN(SELECT DEPTNO
FROM DEPT
WHERE LOC ='CHICAGO');
```

Q6] WAQTD DETAILS OF THE DEPT IN WHCH DEPT START WITH S AND WORKING AS CLERK

```
SELECT *
FROM DEPT
WHERE DNAME LIKE 'S%' AND DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE JOB = 'CLERK');
```

SUBQUERY CASE 2 ASSIGNMENT

1. WAQTD DNAME OF THE EMPLOYEES WHOSE NAME IS SMITH

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO IN(SELECT DEPTNO
FROM EMP
WHERE ENAME = ‘SMITH’);
```

1. WAQTD DNAME AND LOC OF THE EMPLOYEE WHOSE ENAME IS KING

```
SELECT DNAME, LOC
FROM DEPT
```

WHERE DEPTNO IN(SELECT DEPTNO

FROM EMP

WHERE ENAME = ‘KING’);

1. WAQTD LOC OF THE EMP WHOS EMPLOYEE NUMBER IS 7902

SELECT LOC

FROM DEPT

WHERE DEPTNO IN(SELECT DEPTNO

FROM EMP

WHERE EMPNO = 7902);

1. WAQTD DNAME AND LOC ALONG WITH DEPTNO OF THE EMPLOYEE WHOSE NAME ENDS WITH 'R' .

SELECT LOC, DNAME, DEPTNO

FROM DEPT

WHERE DEPTNO IN(SELECT DEPTNO

FROM EMP

WHERE ENAME LIKE ‘%R’);

1. WAQTD DNAME OF THE EMPLOYEE WHOSE DESIGNATION IS PRESIDENT

SELECT DNAME

FROM DEPT

WHERE DEPTNO IN(SELECT DEPTNO

FROM EMP

WHERE JOB = ‘PRESIDENT’);

1. WAQTD NAMES OF THE EMPLOYEES WORKING IN

ACCOUNTING DEPARTMENT

SELECT ENAME

FROM EMP

WHERE DEPTNO IN(SELECT DEPTNO

FROM DEPT

WHERE DNAME=’ACCOUNTING’);

1. WAQTD ENAME AND SALARIES OF THE EMPLOYEES WHO ARE WORKING IN THE LOCATION CHICAGO

SELECT ENAME, SAL

FROM EMP

WHERE DEPTNO IN(SELECT DEPTNO

FROM DEPT

WHERE LOC=’CHICAGO’);

1. WAQTD DETAILS OF THE EMPLOYEES WORKING IN SALES

SELECT *

FROM EMP

WHERE DEPTNO IN(SELECT DEPTNO

FROM DEPT

```
WHERE DNAME='SALES');
```

1. WAQTD DETAILS OF THE EMP ALONG WITH ANNUAL SALARY IF EMPLOYEES ARE WORKING IN NEW YORK

```
SELECT EMP.*, SAL*12
FROM EMP
WHERE DEPTNO IN(SELECT DEPTNO
FROM DEPT
WHERE LOC='NEW YORK');
```

1. WAQTD NAMES OF EMPLOYEES WORKING IN OPERATIONS DEPARTMENT

```
SELECT ENAME
FROM EMP
WHERE DEPTNO IN(SELECT DEPTNO
FROM DEPT
WHERE DNAME='OPERATIONS');
```

SUBQUERY CASE 1 AND SUBQUERY CASE 2 ASSIGNEMT

1. WAQTD NAMES OF THE EMPLOYEES EARNING MORE THAN SCOTT IN ACCOUNTING DEPT

```
SELECT ENAME
FROM EMP
WHERE SAL > (SELECT SAL
FROM EMP
WHERE ENAME = 'SCOTT') AND DEPTNO IN
(SELECT DEPTNO
FROM DEPT
WHERE DNAME='ACCOUNTING');
```

1. WAQTD DETAILS OF THE EMPLOYEES WORKING AS MANAGER IN THE LOCATION CHICAGO

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE JOB = 'MANAGER' AND DEPTNO IN
(SELECT DEPTNO
FROM DEPT
WHERE LOC='CHICAGO'));
```

1. WAQTD NAME AND SAL OF THE EMPLOYEES EARNING MORE THAN KING IN THE DEPT ACCOUNTING

```
SELECT ENAME,SAL
FROM EMP
WHERE SAL > (SELECT SAL
FROM EMP
WHERE ENAME = 'KING' AND DEPTNO IN
(SELECT DEPTNO
```

```
FROM DEPT
WHERE DNAME=’ACCOUNTING’));
```

1. WAQTD DETAILS OF THE EMPLOYEES WORKING AS SALESMAN IN THE DEPARTEMENT SALES

```
SELECT *
FROM EMP
WHERE JOB IN (‘SALESMAN’) AND
DEPTNO IN(SELECT DEPTNO
FROM DEPT
WHERE DNAME=’SALES’);
```

1. WAQTD NAME, SAL, JOB, HIREDATE OF THE EMPLOYEES WORKING IN OPERATIONS DEPARTMENT AND HIRED BEFORE KING

```
SELECT ENAME,SAL,JOB,HIREDATE
FROM EMP
WHERE HIREDATE < (SELECT HIREDATE
FROM EMP
WHERE ENAME=’KING’ AND
DEPTNO IN(SELECT DEPTNO
FROM DEPT
WHERE DNAME=’OPERATION’));
```

1. DISPLAY ALL THE EMPLOYEES WHOSE DEPARTMET NAMES ENDING 'S'.

```
SELECT *
FROM EMP
WHERE DEPTNO IN (SELECT DEPTNO
FROM DEPT
WHERE DNAME LIKE ‘%S’);
```

1. WAQTD DNAME OF THE EMPLOYEES WHOS NAMES HAS CHARACTER 'A' IN IT.

```
SELECT DNAME
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE DNAME LIKE ‘%A%’);
```

1. WAQTD DNAME AND LOC OF THE EMPLOYEES WHOS SALARY IS RUPEES 800.

```
SELECT DNAME, LOC
FROM DEPT
WHERE DEPTNO IN (SELECT DEPTNO
FROM EMP
WHERE SAL=800);
```

1. WAQTD DNAME OF THE EMPLOYEES WHO EARN COMISSION

```
SELECT DNAME, LOC
FROM DEPT
```

WHERE DEPTNO IN (SELECT DEPTNO

FROM EMP

WHERE COMM IS NOT NULL);

1. WAQTD LOC OF THE EMPLOYEES IF THEY EARN COMISSION IN DEPT 40

SELECT LOC

FROM DEPT

WHERE DEPTNO IN (SELECT DEPTNO

FROM EMP

WHERE COMM IS NOT NULL AND DEPTNO = 40);

Max and min using subquery

Q1] WAQTD NAME OF EMP WHO IS EARNING FIRST MAXIMUM SALARY

SELECT ENAME

FROM EMP

WHERE SAL IN(SELECT MAX(SAL)

FROM EMP);

Q2] WAQTD DEPT DETAILS OF EMP WHO HIRED LAST

SELECT * FROM

EMP WHERE HIREDATE IN(SELECT MIN(HIREDATE)

FROM EMP);

MIN AND MAX SUBQUERY ASSIGNMENT

1. WAQTD NAME OF THE EMPLOYEE EARNING MAXIMUM SALARY

SELECT ENAME

FROM EMP

WHERE SAL IN (SELECT MAX(SAL)

FROM EMP);

1. WAQTD NAME OF THE EMPLOYEE EARNING MINIMUM SALARY

SELECT ENAME

FROM EMP

WHERE SAL IN (SELECT MIN(SAL)

FROM EMP);

1. WAQTD NAME AND HIREDATE OF THE EMPLOYEE HIRED BEFORE ALL THE EMPLOYEES (FIRST EMP)

SELECT ENAME, HIREDATE

FROM EMP

WHERE HIREDATE IN (SELECT MAX(HIREDATE)

FROM EMP);

1. WAQTD NAME AND HIREDATE OF THE EMPLOYEES HIRED AT THE LAST

SELECT ENAME, HIREDATE

FROM EMP

WHERE HIREDATE IN (SELECT MIN(HIREDATE)

FROM EMP);

1. WAQTD NAME, COMM OF THE EMPLOYEE WHO EARNS MIN COMISSION

```
SELECT ENAME, COMM
FROM EMP
WHERE COMM IN (SELECT MIN(COMM)
FROM EMP);
```

1. WAQTD NAME, SAL AND COMM OF THE EMPLOYEE EARNING MAXIMUM COMISSION

```
SELECT ENAME, COMM
FROM EMP
WHERE COMM IN (SELECT MAX(COMM)
FROM EMP);
```

1. WAQTD DETAILS OF THE EMPLOYEE WHO HAS

GREATEST EMPNO

```
SELECT *
FROM EMP
WHERE EMPNO IN (SELECT MAX(EMPNO)
FROM EMP);
```

1. WAQTD DETAILS OF THE EMPLOYEES HAVING THE LEAST HIREDATE

```
SELECT *
FROM EMP
WHERE HIREDATE IN (SELECT MIN(HIREDATE)
FROM EMP);
```

1. WAQTD DETAILS OF THE EMPLOYEES EARNING LEAST ANNUAL SALARY

```
SELECT *
FROM EMP
WHERE SAL*12 IN (SELECT MIN(SAL*12)
FROM EMP);
```

1. WAQTD NAME, ANNUAL SALARY OF THE EMPLOYEES IF THEIR ANNUAL SALARY IS MORE THAN ALL THE SALESMAN

```
SELECT ENAME, SAL*12 AS ANNUL_SAL
FROM EMP
WHERE SAL*12 > (SELECT MAX(SAL*12)
FROM EMP
WHERE JOB='SALESMAN')
```

Nested Subquery

- Subquery inside another subquery is called as nested subquery
- We can next up to 255 subquery

1. WAQTD 2ND MAXIMUM SAL

```
SELECT *
FROM EMP
```

WHERE SAL IN (SELECT MAX(SAL)

FROM EMP WHERE SAL < (SELECT MAX(SAL)

FROM EMP));

1. WAQTD DETAILS OF EMP WHO IS EARNING SECOND MAXIMUM SAL

SELECT *

FROM EMP

WHERE SAL IN (SELECT MIN(SAL)

FROM EMP WHERE SAL > (SELECT MIN(SAL)

FROM EMP));

Types of subquery

1. Single row subquery
2. Multi-row subquery

Single row subquery

- Any subquery that return exactly single output are known as single row subquery

1] WAQTD DETAILS OF EMP WHO IS EARNING GREATER THAN SMITH

SELECT *

FROM EMP

WHERE SAL > (SELECT SAL

FROM EMP

WHERE ENAME='SMITH')

MULTI-ROW SUBQUERY

- Any subquery that return one or more than one output such kind of query is called as multi-row subquery

1] WAQTD DETAILS OF EMP WHO ARE WORKING IN SAME SEPARTMENT AS SMITH & JONES

SELECT *

FROM EMP

WHERE DEPTNO IN(SELECT DEPTNO

FROM EMP

WHERE ENAME = 'SMITH' OR ENAME = 'JONES');

SUBQUERY ASSIGNMENT

1. WAQTD 2ND MINIMUM SALARY.

SELECT DISTINCT(SAL)

FROM EMP

WHERE SAL =(SELECT MIN(SAL)

FROM EMP

WHERE SAL >(SELECT MIN(SAL)

FROM EMP));

1. WAQTD 5TH MAXIMUM SALARY.

SELECT DISTINCT(SAL)

FROM EMP

WHERE SAL =(SELECT MAX(SAL)

FROM EMP

WHERE SAL <(SELECT MAX(SAL)

FROM EMP

WHERE SAL<(SELECT MAX(SAL)

FROM EMP

WHERE SAL<(SELECT MAX(SAL)

FROM EMP

WHERE SAL<(SELECT MAX(SAL)

FROM EMP)));

1. WAQTD NAME OF THE EMPLOYEE EARNING 3RD

MAXIMUM SALARY.

SELECT DISTINCT(SAL)

FROM EMP

WHERE SAL =(SELECT MAX(SAL)

FROM EMP

WHERE SAL <(SELECT MAX(SAL)

FROM EMP

WHERE SAL<(SELECT MAX(SAL)

FROM EMP)));

1. WAQTD EMPNO OF THE EMPLOYEE EARNING 2D MAXIMUM SALARY.

SELECT DISTINCT(SAL)

FROM EMP

WHERE SAL =(SELECT MAX(SAL)

FROM EMP

WHERE SAL <(SELECT MAX(SAL)

FROM EMP));

1. WAQTD DEPARTMENT NAME OF AN EMPLOYEE GETTING 4TH MAX SAL.

SELECT DISTINCT(SAL)

FROM EMP

WHERE SAL =(SELECT MAX(SAL)

FROM EMP

WHERE SAL <(SELECT MAX(SAL)

FROM EMP

WHERE SAL<(SELECT MAX(SAL)

FROM EMP

WHERE SAL <(SELECT MAX(SAL)

FROM EMP)));

1. WAQTD DETAILS OF THE EMPLOYEE WHO WAS HIRED 2nd.

```
SELECT *
FROM EMP
WHERE HIREDATE IN (SELECT MAX(HIREDATE)
FROM EMP
WHERE HIREDATE < ( SELECT MAX(HIREDATE)
FROM EMP));
```

1. WAQTD NAME OF THE EMPLOYEE HIRED BEFORE THE LAST EMPLOYEE.

```
SELECT *
FROM EMP
WHERE HIREDATE IN (SELECT MIN(HIREDATE)
FROM EMP
WHERE HIREDATE >( SELECT MIN(HIREDATE)
FROM EMP));
```

1. WAQTD LOC OF THE EMPLOYEE WHO WAS HIRED FIRST.

```
SELECT LOC
FROM DEPT
WHERE DEPTNO IN(SELECT DEPTNO
FROM EMP
WHERE HIREDATE IN(SELECT MAX(HIREDATE)
FROM EMP));
```

1. WAQTD DETAILS OF THE EMPLOYEE EARNING 7TH MINIMUM SALARY.

```
SELECT DISTINCT(SAL)
FROM EMP
WHERE SAL =(SELECT MIN(SAL)
FROM EMP
WHERE SAL >( SELECT MIN(SAL)
FROM EMP
WHERE SAL>( SELECT MIN(SAL)
FROM EMP
WHERE SAL >( SELECT MIN(SAL)
FROM EMP
WHERE SAL>( SELECT MIN(SAL)
FROM EMP
WHERE SAL>( SELECT MIN(SAL)
FROM EMP
WHERE SAL>( SELECT MIN(SAL)
FROM EMP
)))))));
```

ANY and ALL OPERATOR ASSIGNMENT

1. WAQTD NAME OF THE EMPLOYEES EARNING SAL MORE THAN SALESMAN

```
SELECT ENAME
FROM EMP
WHERE SAL >ALL(SELECT SAL
FROM EMP
WHERE JOB IN('SALESMAN'));
```

1. WAQTD SETAILS OF THE EMPLOYEES HIRED AFTER ALL THE CLERKS

```
SELECT *
FROM EMP
WHERE HIREDATE >ALL(SELECT HIREDATE
FROM EMP
WHERE JOB IN('CLERK'));
```

1. WAQTD NAME AND SALARY FOR ALL THE EMPLOYEES IF THEY ARE EARNING LESS THAN ATLEAST A MANAGER.

```
SELECT ENAME, SAL
FROM EMP
WHERE SAL <ANY(SELECT SAL
FROM EMP
WHERE JOB IN('MANAGER'));
```

1. WAQTD NAME AND HIREDATE OF EMPLOYEES HIRED BEFORE ALL THE MANAGERS

```
SELECT ENAME, HIREDATE
FROM EMP
WHERE HIREDATE <ALL(SELECT HIREDATE
FROM EMP
WHERE JOB IN('MANAGER'));
```

1. WAQTD NAME OF THE EMPLOYEES HIRED AFTER ALL THE MANAGERS AND EARNING SALARY MORE THAN ALL THE CLERKS

```
SELECT ENAME
FROM EMP
WHERE HIREDATE >ALL(SELECT HIREDATE
FROM EMP
WHERE JOB IN('MANAGER') AND SAL >ALL(SELECT SAL
FROM EMP
WHERE JOB IN('CLERK')));
```

1. WAQTD DETAILS OF THE EMPLOYEES WORKING AS CLERK AND HIRED BEFORE ATLEAST A SALESMAN

```
SELECT *
FROM EMP
WHERE HIREDATE <ANY (SELECT HIREDATE
FROM EMP
WHERE JOB IN('SALESMAN') AND JOB IN(SELECT JOB
FROM EMP
```

WHERE JOB = ‘CLERK’));

1. WAQTD EMP NAMES IF EMPLOYEES ARE HIRED AFTER ALL THE EMPLOYEES OF DEPT 10.

SELECT ENAME

FROM EMP

WHERE HIREDATE >ALL(SELECT HIREDATE

FROM EMP

WHERE DEPTNO=10);

1. DISPLAY ALL THE EMPLOYEES WHO ARE EARN LESS THAN ANY OF THE SALESMAN?

SELECT *

FROM EMP

WHERE SAL <ANY(SELECT SAL

FROM EMP

WHERE JOB IN(‘SALESMAN’));

EMP-MANAGER RELATIONSHIP

EMP4

EMP3

EMP2

EMP1

MANAGER2

MANAGER 1

CEO

- In the EMP-MGR relationship there are two case:
    1. Identifying manager
    2. Identifying employees

1]identifying managers

Q1] WAQTD MANAGER NAME OF SMITH

SELECT ENAME

FROM EMP

WHERE EMPNO IN(SELECT MGR

FROM EMP

WHERE ENAME=’SMITH’);

Q2] WAQTD MILLER MANAGER MANAGER’S NAMES

SELECT ENAME

FROM EMP

WHERE EMPNO IN(SELECT MGR

FROM EMP

WHERE EMPNO IN(SELECT MGR

FROM EMP

WHERE ENAME=MILLER);

NOTES:

To identify manager ===> MGR in subquery

To identify employees ======> EMPNO In subquery

Identifying employees: =

Q1] WAQTD NO OF EMP REPORTING TO FORD MANAGER

```
SELECT ENAME

FROM EMP

WHERE MGR IN(SELECT EMPNO

FROM EMP

WHERE ENAME = 'FORD');
```

Q2] WAQTD NO OF EMP REPORTING TP SCOTT MANAGER'S MANAGER

```
SELECT COUNT (*)

FROM EMP

WHERE MGR IN (SELECT EMPNO

FROM EMP'

WHERE MGR IN (SELECT EMPNO

FROM EMP

WHERE ENAME='SCOTT'));
```

Q3] WAQTD MAX SAL AMONG EMP REPOSTING KING

```
SELECT MAX(SAL)

FROM EMP

WHERE MGR IN(SELECT EMPNO

FROM EMP

WHERE ENAME='KING');
```

Joins

- Join is type of DQL
- Join is used to retrieve data from two or more than two table simultaneously
- Types of join

1. Cross join / Cartesian join
2. Inner join / equi join
3. Natural join
4. Outer join
5. Left outer join
6. Right outer join
7. Full outer join
8. Self-join

Cross join

- Cross join is type of join
- Cross join is used to join records in one table with record from another table
- Cross join provides more mismatched data
- Cross join = total records Table1 * total record table2
- Syntax

1] ANSI: American national standard institute

```
SELECT COL_NAME / EXPRESSION

FROM TABLE1 CROSS JOIN TABLE2;
```

2] ORACLE

SELECT COL_NAME / EXPRESSION

FROM TABLE1, TABL2;

Q1. WAQTD TO DESPLAY MATCHED AS WELL AS MISMATCHED RECORED IN DEPT AND EMP TABLE

SELCT *

FROM DEPT,EMP;

Application

- To analyses data
- Generating test data
- Implement the concept permutation and combination

2] inner join

- Type of join
- Used to retrieve data from table to get only matching records
- Syntax

1] ANSI

SELECT COLUMN_NAME / EXPRESSION

FROM TABLE1 INNER JOIN TABLE2

ON <CONDITION>

2] ORACLE

SELECT COLUMN_NAME / EXPRESSION

FROM TABLE1, TABLE2

WHERE <CONDITION>

Q1] WAQTD EMP NAME ALONG WITH ITS DNAME

SELECT ENAME, DNAME

FROM EMP, DEPT

WHERE EMP.DEPTNO = DEPT.DEPTNO;

OR

SELECT ENAME, DNAME

FROM EMP INNER JOIN DEPT

ON EMP.DEPTNO = DEPT.DEPTNO;

NATURAL JOIN

- TYPE OF JOIN
- Used to retrieve data from multiple table at a time
- It gives output as cross join when there is no relation between table
- It gives output as inner join when there is relation between table
- Syntax

SELECT COLUMN_NAME

FROM TABLE1 NATURAL JOIN TABLE2;

- EXAMPLE

SELECT *

FROM EMP NATURAL JOIN DEPT;

OUTER JOIN

- Type of join
- Used to get unmatched records along with matching records
- It has 3 type

1] left outer join

2] right outer join

3] full outer join

1] left outer join

- Type of outer join
- Used to get all matching records from both the table and unmatched records from left table
- **Syntax:**

1] ANIS

SELECT COL_NAME / EXPRESSION

FROM TABLE1 LEFT [OUTER] JOIN TABLE 2

ON TABLE1.COL_NAME = TABLE2.COL_NAME;

2] ORACLE

SELECT COL_NAME / EXPRESSION

FROM TABLE1, TABLE 2

WHERE TABLE1.COL_NAME = TABLE2.COL_NAME (+);

Q1] WAQTD DETAILS OF EMP & DEPT ALOG WITH UNMATCHED RECORDS FROM EMP TAABLE

SELECT *

FROM EMP, DEPT

WHERE EMP.DEPTNO = DEPT.DEPTNO (+);

Right outer join

- Type of outer join used to get all unmatched records from right table and matched records from the both tables
- Syntax

1]ANSI

SELECT COL_NAME / EXPRESSION

FROM TABLE1 RIGHT [OUTER] JOIN TABLE 2

ON TABLE1.COL_NAME (+) = TABLE2.COL_NAME ;

2] ORACLE

SELECT COL_NAME / EXPRESSION

FROM TABLE1, TABLE 2

WHERE TABLE1.COL_NAME (+) = TABLE2.COL_NAME;

Q1] WAQTD DETAILS OF EMP & DEPT ALONG WITH UNMATCHED RECORDS FROM DEPT

SELECT *

FROM EMP, DEPT

WHERE EMP.DEPTNO (+) = DEPT.DEPTNO;

FULL OUTER JOIN

- Type of join used to get matched records from both the tables and unmatched records from both the table
- Syntax

SELECT COL_NAME / EXPRESSION

FROM TABLE1 FULL [OUTER] JOIN TABLE 2

ON TABLE1.COL_NAME = TABLE2.COL_NAME ;

Q1] WAQTD MATCHED AS WELL AS UNMATCHED RECORDS IN DEPT & EMP TABLE

SELECT *

FROM EMP FULL JOIN DEPT

ON EMP.DEPTNO = DEPT.DEPTNO;

SELF JOIN

- Type of join
- Used to combining two same tables are called as self-join
- Whenever the multiple data to be selected from same table but are present in same column different row used self-join
- Syntax

1] ANSI

SELECT COL_NAME/ EXPRESSION

FROM TN1 SELF JOIN TN2

ON <SELF-JOIN CONDITION>

2] ORACLE

SELECT COL_NAME / EXPRESSION

FROM TN1, TN2

WHERE <SELF-JOIN CONDITION>;

Q1] WAQTD ENAME AND ITS MANAGER NAME

SELECT E1.ENAME, E2.ENAME

FROM E1.EMP,E2.EMP

WHERE E1.MGR = E2.EMPNO;

Q2] WAQTD NAME SAL OF EMP ALONG WITH THEIR MANAGER NAME & THEIR SAL

SELECT E1.ENAME EMP_NAME, E1.SAL EMP_SAL, E2.ENAME MGR_NAME, E2.SAL MGR SAL

FROM E1.EMP,E2.EMP

WHERE E1.MGR = E2.EMPNO;

Co-related subquery

- It is type of subquery which works on principle of subquery & joins

Outer query

Final o/p

Partial o/p

Inner query o/p

Inner Query

- In co-related subquery always outer query will execute first & provide partial output
- This Partial output given as input to the inner query
- The inner query execute & provide output
- Output of inner query is given to the outer query
- Outer query execute & provide final result

Q1] WAQTD DNAME OF EMP IN WHICH EMP ARE WORKING

SELECT D1.DNAME

FROM DEPT D1

WHERE D1.DEPTNO IN(SELECT E1.DEPTNO

FROM EMP E1

WHERE D1.DEPTNO = E1.DEPTNO);

- To overcome the drawback of given query we use subquery exists & not exists operator

1] exists

- Unary operator which accept one operand
- It returns true if subquery provide any values other than null

2]not exists

- It is unary operator
- It returns true if the subquery return null & returns false for other value

Q1] WAQTD 10^{TH} MINIMUM SAL IN EMP TABLE

SELECT E1.SAL

FROM EMP E1

WHERE 10-1 IN (SELECT COUNT (DISTINCT E2.SAL)

FROM EMP E2

WHERE E1.SAL < E2.SAL);

DDL [DATA DEFINATION LANGUAGE]

- It is type of statement
- It is used to create an object/table/entity within the database & deals with structure of table
- Types

1. Create
2. Alter
3. To add column
4. To rename column
5. To drop column
6. To modify datatype
7. To modify constraint
8. To assigning foreign key
9. Rename
10. Truncate
11. Drop

Create

- It is type of DDL statement
- It is used to create/build a table/entity in database
- Syntax

CREATE TABLE TABLE_NAME (

COL_NAME DATATYPE CONSTRAINTS,

COL_NAME DATATYPE CONSTRAINTS,

COL_NAME-N DATATYPE CONSTRAINTS

);

- Example:

CREATE TABLE STUDENT (

SID NUMBER PRIMARY KEY,

SNAME VARCHAR (20) NOT NULL,

ADDS VARCHAR (20) NOT NULL

);

RENAME

- It is type of DDL statement
- It is used to give new name to existing table name
- Syntax

RENAME EXISTING_TABLE_NAME TO NEW_TABLE_NAME;

- Example

RENAME STUDENT TO STUD;

Alter

- Alter is a type of DDL statement
- It is used to modify the structure of table

1] adding new column to existing table

- Syntax

Alter table table_name

Add column_name datatype constraints

- Example

Alter table student

Add column phone_no number(10) not null;

2] to renaming existing column

- Syntax

Alter table table_name

Rename column old_col_name to new_col_name;

- Example

Alter table student

Rename phone_no to mobile_no;

3] remove a column from table

- Syntax

Alter table table_name

Drop column column_name;

- Example

Alter table student

Drop column mobile_no;

4] to modify the datatype

- Syntax

Alter table table_name

Modify col_name datatype

- Example

Alter table student

Modify sid varchar (5);

5] to modify the constraints

- Syntax

Alter table table_name

Modify col_name datatype new_constraints (null / not null);

- Example

Alter table student

Modify sname varchar (50) null;

6]to add constraints

- Syntax

Alter table table_name

Add constraints con_ref-name constraint-type(col_name);

- Example

Alter table student

Add constraint uin_email unique(mail);

- We can use the col-ref-name to remove particular constraint from particular column
- It should be unique
- Check assigned constraint to particular column

Select *

From user.constraints

Where table_name = ‘table_name’;

7] to assign a foreign key

- Syntax

Step1: add column

Alter table table-name

Add col_name datatype;

Step2: to add foreign key

Alter table table_name

Add constraint con_ref_name con_name(col-name)

References table_name(col_name);

- Example

Stape1:

Alter table student

Add cid number(5);

Stape2:

Alter table student

Add constraint cid_fk foreign key(cid)

References collage(cid);

8] how to establish connection while creating table

- Example:

Create table TEACHER (

TID NUMBER PRIMARY KEY,

TNAME VARCHAR(40) NOT NULL

)

CREATE TABLE STUDENT (

SID NUMBER PRIMARY KEY,

SNAME VARCHAR(40) NOT NULL,

TID NUMBER,

CONSTRAINT FK_ID FOREIGN KEY(TID)

REFERENCES TEACHER(TID)

);

TRUNCATE

- It is a type of DDL statement
- Used to delete all the record permanently from table but table structure remaining same
- Syntax

Truncate table_name;

- Example

Truncate student;

Drop

- Drop is type of DDL statement
- Used to delete table from database temporary along with structure
- Syntax

Drop table table_name;

- Example

Drop table student;

Flashback

- Used to get back dropped table from recycle bin back to database
- Syntax

Flashback table table_name

To before drop;

- Example

Flashback table student

To before drop;

Purge

- Used to delete table permanently from the recycle bin
- Purge will work only on recycle bin
- Syntax

Purge table table_name;

- Example

Purge table student;

DML (DATA MNIPULATION LANGUAGE)

- It Is statement which is used to modify data present in the table by performing insert, update, delete
- Types

1. Insert
2. Delete
3. Update

1] Insert

- Used to insert records in specific table
- Syntax

Insert into table_name values(RN1,RN2……RNn)

- Example

Insert into student values(12,'Akshay','pune');

2] update

- Type of DML statement
- Used to update or modify records present in table
- Syntax

Update table_name

Set col_name=value

Where =<consition>

- Example

Update student

Set sname='samir'

Where sid = 12;

3] delete

- It is type of DML STATEMENT
- Used to delete specific records from table
- Syntax

Delete from table_name

Where col_name = value;

- Example

Delete from student

Where sid = 12;

Note

DML statement are not auto commit but DDL statement are auto commit

TCL: = [TRANSACTION CONTROL LANGUAGE]

- Type of statement used to control the transaction done by DML statement
- Types
- Commit
- Rollback
- Savepoint

1] commit

- Type of TCL language
- Which is used to save records into the table
- When the transaction is perform using DML statement used commit command to save these tractions into the table
- Syntax

Commit;

- Example

Commit;

2] Rollback

- Type of TCL language
- Used to get back to current recently committed position
- It works like as undo operation
- It will work before commit statement

- Rollback destroy the records after save point
- Syntax

Rollback to savepoint;

- Example

Rollback to s1;

3] savepoint

- It is TCL statement which is used to provide restoration point in the records
- It will work before commit
- after commit save point will destroy
- Syntax

Savepoint savepoint_name

- Example

Savepoint s1;

DCL (DATA CONTROL LANGUAGE)

- Used to control the data flow between two users
- Types
    - 1. Grant
      2. Revoke

1] grant

- Used to provide permission to database to another users
- It is type of DCL
- Syntax

Grant SQL_statement

On table_name

To user_name;

- Example

Grant select

On jobs

To scott;

Revoke

- It is type of DCL
- Used to get back permission from user to who are granted permission to access DB
- Syntax

Revoke SQL_statement

On table

From user_name;

- Example

Revoke select

On jobs

From scott;

Single row functions types

1. Length
2. Concat
3. Mod
4. Upper

5. Lower
6. Initcap
7. Reverse
8. Substr
9. Instr
10. Replace
11. Round
12. Trunc
13. Last_day
14. Month_between
15. To_char
16. NVL
17. Trim

1] Length()

- Used to find out the length of the Given String
- Syntax

Length(String)

- Example

Select Length('Akshay')

From dual;

2] Concat()

- used to merge two string or combine two String
- syntax

concat(str1,str2)

- example

select concat(ename,'@gmail.com')

from emp;

3] mod()

- Used to divide two values & return remainder
- Syntax

Mod(n1,n2);

Q1.WAQTD NAME OF EMP WHO ARE EARNING EVEN SALARY

SELECT ENAME

FROM EMP

WHERE MOD(SAL,2) = 0;

Q2. WAQTD NAME OF EMP WHO ARE EARNING ODD SALARY

SELECT ENAME

FROM EMP

WHERE MOD(SAL,2) != 0;

4] Upper()

- Used to convert all the letter in lower case into upper case but already letter in upper case its remains same
- Syntax

Upper(String)

- Example

Select Upper(ename)

From emp;

5] lower()

- Used to convert all the letter in Upper case into Lower case but already letter in lower case its remains same
- Syntax

Lower(String)

- Example

Select lower(ename)

From emp;

6] initCap()

- It is a function used to convert first letter of the string to capital & remaining letter Lowercase
- Syntax

Initcap(string)

- Example

Select initCap(ename)

From emp;

7] Reverse()

- It is Single row function used to reverse the given String
- Syntax

Reverse(string)

Q1] WAQTD THE EMP NAME IS PALINDROME OR NOT

SELECT ENAME

FROM EMP

WHERE ENAME = REVERSE(ENAME);

Q2] WAQTD JOB OF EMP IN REVERSE ORDER

SELECT REVERSE(JOB)

FROM EMP;

8] SUBSTR()

- It is function which is used to extract part of a String from original String
- Syntax

Substr(‘ originalString ’ , position[length])

Q1. WAQTD NAME OF EMP WHO HAS 3RD LETTER AS R IN THERE NAMES

SELECT ENAME

FROM EMP

WHERE SUBSTR(ENAME,3,1) = ‘R’

Q2. WAQTD THE FIRST HALF OF THE LETTER FROM EACH EMP NAME

SELECT ENAME,SUBSTR(ENAME,1,LENGTH(ENAME)/2)

FROM EMP;

Q3. WAQTD EMP NAME WHO HAS LAST LETTER IS ‘E’

SELECT ENAME

FROM EMP

WHERE SUBSTR(ENAME,LENGTH(ENAME))= ‘E’;

9] INSTR()

- It is function which is used to obtain the position in which string is present in the original string
- Syntax

Instr(string,'string',position,[occurance])

Q1. WAQTD POSITION OF Y LETTER WITHIN THE MALAYALM WHICH IS PRESENT AT SECOND OCCURANCE

SELECT INSTR('MALAYALAM','Y',1,2)

FROM DUAL;

Q2. WAQTD NAME OF EMP WHO HAS CHARACTER A IN THERE NAME

SELECT ENAME

FROM EMP

WHERE INSTR(ENAME,'A',1) !=0;

Q3] WAQTD NAME OF EMP WHO HAS 2 'A' LETTER IN THEIR NAME

SELECT ENAME, INSTR(ENAME,'A',1,2)

FROM EMP

WHERE INSTR(ENAME,'A',1,2) != 0;

Q4] WAQTD NAMES OF EMP WHO ARE HAVING TWO CONSECTIVE 'L' IN THEIR NAME

SELECT ENAME

FROM EMP

WHERE INSTR(ENAME,'LL',1) > 0;

10] Replace()

- It is function used to replace the string with other string within original string
- It is mainly used to find how many time one particular string is available in given string
- Syntax

Replace (original_string, string,[new string]);

Q1] WAQTD DETAILS OF EMP IF THEY HAVE EXACTIL 2 TIME 'R' IN THEIR NAME

SELECT *

FROM EMP

WHERE LENGTH(ENAME) – LENGTH(REPLACE(ENAME,'R')) = 2;

11] Round()

- It is function used to round the decimal value to it nearest value
- Syntax

Round(value);

12] Trunc()

- It is function used to round the to the nearest lowest value
- Syntax

Trunc(value)

13] Months_between()

- It is function used to find out the number of month between to dates
- Syntax

Months_between(date1,date2)

- Date1 ===> is maximum date
- Date 2 ===> is minimum date
- Sysdate
- It is a date which automatically provide by system date

Q1] WAQTD FROM HOW MANY YEAR EACH EMPLOYEES WORK IN COMPANY

```
SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE)

FROM EMP;
```

Q2] WAQTD DETAILS OF EMP WHO ARE WORKING IN COMPANY FROM 40 YEAR

```
SELECT *

FROM EMP

WHERE MONTHS_BETWEEN(SYSDATE,HIREDATE)/12 > 40;
```

14] Last_Day()

- It is function used to get last day of given date
- Syntax

```
Last_day(sysdate)
```

Q1] WAQTD EMP HIRED MONTH LAST DAY ALONG WITH DISPLAY THEIR NAME & HIREDATE

```
SELECT ENAME,HIREDATE,LAST_DAY(HIREDATE)

FROM EMP;
```

15] NVL (null value logic)

- Whenever we perform arithmetic operation with null value it will return null to overcome this drawback of null values we go for NVL()
- Syntax

```
NVL(arg1,arg2)
```

- Arg1 ====> col_name which have null value
- Arg2 ====> replacement value for null.
- Example

```
Select sal+NVL(comm,0) total_sal

From emp;
```

16] To_char()

- It is function used to convert date into string format
- Syntax

```
To_char(date,'format_date')
```

- Format

1. Year : Twenty twenty five
2. YYYY : 2025
3. YY : 25
4. Month: march
5. MON : mar
6. Mm : 03
7. dd : 15
8. day : Saturday
9. d : 6
10. HH24: 20
11. HH12:8
12. MI : 15
13. SS : 18
14. DD-MM-YYYY : 12-02-2025
15. MM-DD-YYYY : 01-05-2025

Ex1

```
Select to_char(sysdate,'dd')

From dual;
```

Ex2

Select to_char(sysdate,’dd-mm-yyyy’)

From dual;

Ex3

Select to_char(sysdate,’mon’)

From dual;

17] trim()

- It is function used to remove the extra space in given string
- Syntax

Trim(string)

Q1] WAQTD NAMES OF EMP WHO GOT HIRED ON SUNDAY

SELECT ENAME

FROM EMP

WHERE TRIM(TO_CHAR(HIREDATE,’DAY’))=’SUNDAY’;

ASSIGNMENT QUATION WITH ANS

1. Display the employee name and the first letter of the employee's name.

SELECT SUBSTR(ENAME,1,1) AS FIRST_LETTER

FROM EMP;

2. List all employee names and their salaries, rounded to two decimal places.

SELECT ROUND(SAL,2)

FROM EMP;

3. Show the employee names and the length of each employee's name.

SELECT ENAME,LENGTH(ENAME) AS ENAME_LENTH

FROM EMP;

4. List the employee names and the uppercase version of their job titles.

SELECT ENAME,UPPER(JOB)

FROM EMP;

5. Show the employee names and the last 3 characters of their job titles.

SELECT ENAME,SUBSTR(JOB,1,3) JOB_FIRST_3LETTER

FROM EMP;

6. Find the employee names and the hire year extracted from their hire date.

SELECT ENAME, EXTRACT(YEAR FROM HIREDATE) AS HIRE_YEAR

FROM EMP;

7. Display the employee names and the month they were hired.

SELECT ENAME, EXTRACT(MONTH FROM HIREDATE) AS HIRE_YEAR

FROM EMP;

9. Display employee names along with their job title, but with the first letter capitalized.

SELECT ENAME,INITCAP(JOB)

FROM EMP;

10. Show the employee names and their salaries, formatted as currency (e.g., "$1,000").

SELECT ENAME , CONCAT('$',SAL)

FROM EMP;

11. Show employee names and the department number they belong to, along with the department number as a 5-digit number (with leading zeros if necessary).

SELECT ENAME, DEPTNO, TO_CHAR(DEPTNO, '00000') AS FORMATTED_DEPTNO

FROM EMP;

13. Display employee names along with their hire date in the format "DD-MON-YYYY".

SELECT ENAME,TO_CHAR(HIREDATE,'DD-MON-YYYY')

FROM EMP;

14. List employee names and show their salaries, rounded to the nearest 100.

SELECT ENAME, SAL, ROUND(SAL, -2) AS SALARY_ROUNDED

FROM EMP;

16. Show the employee names and their job titles, where the job title is displayed with only the first character in uppercase.

SELECT ENAME,INITCAP(JOB)

FROM EMP;

17 Show the employee names and the number of months between their hire date and today.

SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE) TOTAL_MONTH

FROM EMP;

9. Find the employee names and their salaries, rounded to the nearest 1000.

SELECT ENAME , ROUND(SAL,-3)

FROM EMP;

10. Display the employee names and the day of the week they were hired.

SELECT ENAME,MONTHS_BETWEEN(SYSDATE,HIREDATE) /7 WEEK

FROM EMP;

11. List employee names and show the last character of their name.

SELECT SUBSTR(ENAME,-1)

FROM EMP;

12. Show the employee names and the number of days between their hire date and today.

SELECT MONTHS_BETWEEN(SYSDATE,HIREDATE) * 365

FROM EMP;

13. Display employee names and their department number, formatted as a 4-digit string with leading zeros.

SELECT ENAME,TO_CHAR(DEPTNO,'0000')

FROM EMP;

14. List employee names and show their salaries as a percentage (e.g., 2500 becomes 250000%).

SELECT ENAME, SAL, TO_CHAR(SAL * 100) || '%' AS SALARY_PERCENTAGE

FROM EMP;

16. Find the employee names and display the current salary increased by 20%.

SELECT ENAME,TO_CHAR(SAL+SAL*0.20)

FROM EMP;