Select \* from tab

Desc tablename

Select \* from v$version

**Data Definition Language (DDL):-** To Create/Alter/Drop/Truncate/Rename tables

**Data Manipulation Language (DML):-** To Insert/Update/Delete/Merge Columns

**Data Query/Retrieval Language(DQL):-** Select

**Data Control Language(DCL):-** Grant, Revoke

**Transaction Control Language(TCL):-** Commit, Rollback, Savepoint

**Data Types:-**

**Character Data types:**

Char – character data, fixed length, default size 1 byte, maximum size 2000 bytes

Varchar – character data, variable length, no default size, maximum size 2000 bytes

varchar2- character data, variable length, no default size, maximum size 4000 bytes.

Long – character data, fixed length, no default size, maximum size 2gb, should only contain one column.

Clob – to store large character data, no default size, maximum size 128tb,any no. Of columns.

**Number:**

Number(p)- Integer value, p can range from 1 to 38

Number(p,s) – Float Value, s can range from -84 to 127

Default size is 38 digits

**Date:**

For Date & time

It will be in ‘mm-dd-yyyy’ format ex:- ’11-13-1992’, Time (12:00:00:AM)

**External files like .txt,.csv,.doc,.pdf operating system Files:**

bfile – To save operating system files

**Graphic Data Types:-**

RAW – It is used to save binary data like images,photos,signatures,thumb impressions. No default size, Maximum 2000 bytes

LONG RAW – It is used to store binary data like images, videos, audio. No default size, Maximum 2GB,Only one Column

BLOB – It is used to store binary data like images,videos,aaudio. No default size, Maximum 128Tb.

**Data Definition Language(DDL**):-

create table student(s\_id number(5), s\_name varchar(30), doj date, gender char)

**Data Manipulation Language(DML):-**

INSERT INTO student(s\_id,s\_name,doj,gender) VALUES(100,’Srikanth’,’11-13-1992’,’M’)

**Data Retrieval/Query Language(DRL/DQL):-**

Select \* from tablename

Select column\_1,column\_3,column\_5 from tablename

Select column\_1 from tablename

**Distinct/Unique : To get unique values for specified columns**

Select distinct column\_name from tablename

Select unique column\_name from tablename

Select distint/unique column\_name,column\_name from tablename – **To get disctinct values from specified columns**

**Creating alias name for columns: To create alias names for already defined columns or for newly created columns**

Select ename,sal,hiredate,(sysdate-hiredate)365 as experience from emp

Select ename ‘Employee Name’,sal ‘Salary’ from emp

**Order by Clause:** Used to arrange the data in ascending or descending order based on one or more columns. Default order is Ascending.**Null is the greatest value when compared to not null values.**

select \* from emp order by deptno desc

select \* from emp order by deptno nulls first

select \* from emp order by comm desc nulls last

select empno,ename,sal,(sal\*12) as ann\_sal from emp order by 4/ann\_sal/(sal\*12) desc

**Where Clause:-** Used to filter the data based on condition known as Restriction.

Select [distinct] column\_names from tablename where <condition> [order by column\_name]

\*\***Alias name can not be applied in Where Clause can be only possible in order Clause**

**SQL Operators:**

**Arithmetic Operators:- +,-,\*,/**

**Relational Operators:- <,>,<=,>=,=,in between, LIKE, is null**

**Relational Negation Operators: !=(or)n=(or)<>,not in, not between, not LIKE, is not null**

**Logical Operators: and, or, not**

**Set Operators: Union, Union all, Intersect, Minus**

**IN Operator:- works like it OR operator which checks in every value in ()**

**Between Operator:- which checks for values between given range**

**LIKE Operator:- To check pattern matching we should include % or \_**

**% :- it matches one or more strings**

**\_ :- it matches single letter**

select \* from emp where hiredate like '04%'

**Constraints:-** It is a rule or restriction which is enforced during creation of the table. The constraint raises automatically when user perform DML operation on the table.

Three types of Constraints:

1. Domain Integrity Constraint: Not Null, Check

2. Entity Integrity Constraint: Unique, Primary Key

3. Referential Integrity Constraint: References

**Not Null:-** It is used to restrict null values in to the column value.

**Check:-** It is used to check whether the given values in the defined range.

Create table(sal number(8,2) check(sal between 1500 and 2500))

Create table(job varchar(25) check(job in (‘Manager’,’Salesman’,’Clerk’))

**Unique:** It is to restrict duplicate values in to the column. We can give any number of null values in that column.

**Primary Key:** It is a combination of **Not Null** and **Unique** values. Only one primary key should be in one table. Only 32 columns can be in the table when primary key is defined.

**Data Redundancy:** Repetition of same data again and again in the table is called as Data Redundancy.

**Data Inconsistency:** Invalid data or incorrect data in the columns.

**Normalization:** Breaking down the data and storing in to multiple tables in order to avoid Data Redundancy and Inconsistency and null entries. The main concept of normalization is functional dependency (direct dependency).

**Reference:** It is used to maintain relationship between columns of two different tables. The table on which reference constraint applied is known as child table or dependent table. The column in which Reference constraint is applied known as **foreign key.** The table to which foreign key is reffering to is called as parent table. Parent table must contain one **Primary Key.** The 2 clauses created along with reference constraint is **on delete cascade, on delete set null.**

**On delete cascade:** if data in parent table is deleted, the values in child table will also be deleted without any notice.

**On delete set null:** If data in parent table is deleted, the values in child table will set as null without any notice.

**Inner JOIN:**

SELECT \*

FROM left\_table

INNER JOIN right\_table

ON left\_table.id = right\_table.id;

If columns of different tables have same name use **tablename.columnname**. **if** column name is unique just use column name as it is while joining tables.

SELECT \*

FROM left\_table

INNER JOIN right\_table

ON left\_table.id = right\_table.id

INNER JOIN another\_table

ON left\_table.id = another\_table.id;

SELECT \*

FROM countries

INNER JOIN economies

ON countries.code = economies.code

You can use USING as a shortcut:

SELECT \*

FROM countries

INNER JOIN economies

USING(code)

SELF INNER JOIN

-- Select fields with aliases

SELECT p1.country\_code,

       p1.size AS size2010,

       p2.size AS size2015,

       -- Calculate growth\_perc

       ((p2.size - p1.size)/p1.size \* 100.0) AS growth\_perc

-- From populations (alias as p1)

FROM populations AS p1

  -- Join to itself (alias as p2)

  INNER JOIN populations AS p2

    -- Match on country code

    ON p1.country\_code = p2.country\_code

        -- and year (with calculation)

        AND p1.year = p2.year - 5;

**CASE WHEN & THEN**

SELECT name, continent, code, surface\_area,

    -- First case

    CASE WHEN surface\_area > 2000000 THEN 'large'

        -- Second case

        WHEN surface\_area > 350000 THEN 'medium'

        -- Else clause + end

        ELSE 'small' END

        -- Alias name

        AS geosize\_group

-- From table

FROM countries;

INTO – To save the result into other table use INTO

SELECT name, continent, code, surface\_area,

CASE WHEN surface\_area > 2000000

THEN 'large'

WHEN surface\_area > 350000

THEN 'medium'

ELSE 'small' END

AS geosize\_group

INTO countries\_plus

FROM countries;