**Create database**

create database databasename;

**List databases**

\list

Or \l

**Connect to database**

\connect databasename

Or \c databasename

**Create user**

create user username with password ‘thepassword’;

**Give access to user for database and operations**

Giving access of database

grant all privileges on database databasename to username;

Giving access of operations

grant all privileges on all tables in schema public to username;

**DDL(Data Definition Language)**

Defining the schema structure so Creating tables, altering tables are part of ddl

**DML(Data Mainipulation Language)**

CRUD operations : Adding new rows using insert, fetching rows using select, update rows using update, deleting rows using delete are part of dml

**Create Table(DDL)**

SQL> create table employees(id int ,name varchar(20), balance double precision);

Table created.

**Schema Description**

\d tablename

**Insert row into table(DML)**

insert into employees(id,name, balance) values(1 ,'ram' , 300.50);

**Select data from table(DML)**

select \* from employees;

**Select columnnames for which result should be fetched(DML)**

select id from employee;

**Delete row from the table(DML)**

delete from employees where name='dharma';

This will delete rows for employees with name dharma

**Change Schema structure(DDL)**

Use Alter command

**Rename table(DDL)**

alter table oldtablename rename to newtablename;

**Adding column(DDL)**

alter table employees add columnname datatype;

**Drop column(DDL)**

alter table employees drop columnname ;

**Add Not null constraint(DDL)**

alter table employees alter column columnname set not null

**Add unique constraint(DDL)**

alter table employees add constraint employee\_name\_unique unique(name);

**Add Primary key(DDL)**

alter table employees add constraint employees\_id\_pk primary key(id);

**Update row(s)**

update employees set name='bluetooth' where id=6;

**Create another Table**

Create table departments(id number, deptname varchar2(20));

**Add Primary key constraint**

alter table departments add constraint dept\_id\_pk primary key(id);

Table altered.

Adding column

SQL> alter table employees add did number;

Table altered.

**Dropping column**

Alter table employees drop column did;

**Add Foreign key constraint**

SQL>alter table employees add constraint employees\_did\_fk foreign key(did) references departments(id);

Now you cant add a row in employees table with did value which does NOT exist as primary key in references table departments

Null Foreign key is permissible though

**Select matching rows**

Fetch rows where employees id exist in departments

SQL> select \* from employees e,departments d where e.did=d.id;

Fetch rows where employees id exist in departments and employee’s id should be 2

SQL> select \* from employees e,departments d

where e.did=d.id and e.id=2;

**IN clause**

Only find employees where id is 1 or 2 or 3 using in clause

select \* from employees where id in (1,2,3);

**Nested Inner query**

Fetch those employees who have department testing

select \* from employees where deptid in (select id from departments where dept\_name='testing');

In the above query, inner query will execute first , its result will be used in outer query

**Not in clause**

Only find employees where id is NOT 1 or 2 or 3 using not in clause

select \* from employees where id not in (1,2,3);

Above nested inner query can be used with not in clause, only mention **not in** where you are mentioning **in**

**LIKE clause**

Find employees where names start from sc

select \* from employees where name like 'sc%';

Find employees where names ends with sc

select \* from employees where name like '%sc';

Find employees where names contain sc

select \* from employees where name like '%sc%';

**Between**

select \* from employees where salary between 1000 and 2000;

**On Clause**

**INNER JOIN**

SELECT \* FROM employees e INNER JOIN departments d ON e.did=d.id;

SELECT \* FROM employees e INNER JOIN departments d ON e.did=d.id;

ID NAME DID ID NAME

---------- -------------------- ---------- ---------- --------------------

1 akash 11 11 it

2 ab 12 12 hr

On clause with whereto restrict

SELECT \* FROM employees e INNER JOIN departments d ON e.did=d.id where e.id=2;

**LEFT JOIN**

SELECT \* FROM employees e LEFT JOIN departments d ON e.did=d.id;

This will fetch the rows where there is a foreign key and primary key match and also the rows where foreign key is null

insert into employees(id,name) values(3,'chandra');

SELECT \* FROM employees e LEFT JOIN departments d ON e.did=d.id;

ID NAME DID ID deptNAME

---------- -------------------- ---------- ---------- --------------------

1 akash 11 11 it

2 ab 12 12 hr

3 chandra

**Functions**

select \* from employees;

ID NAME DID SALARY

---------- -------------------- ---------- ----------

1 akash 11 1000

2 ab 12 2000

3 chandra 3000

select sum(salary) from employees;

SUM(SALARY)

-----------

6000

select avg(salary) from employees;

AVG(SALARY)

-----------

2000

select max(salary) from employees;

MAX(SALARY)

-----------

3000

select min(salary) from employees;

MIN(SALARY)

-----------

1000

Count of number of rows

select count(id) from employees;

COUNT(ID)

----------

4

**Order by**

Descending order

select \* from employees order by salary desc;

ID NAME DID SALARY

---------- -------------------- ---------- ----------

3 chandra 3000

2 ab 12 2000

1 akash 11 1000

**Ascending Order**

select \* from employees order by salary asc;

ID NAME DID SALARY

---------- -------------------- ---------- ----------

1 akash 11 1000

2 ab 12 2000

3 chandra 3000

**Group By**

The GROUP BY Statement in SQL is used to arrange identical data into groups with the help of some functions.

Total salaries paid in every department

select did, sum(salary) from employees group by did;

Average salary of employees with same name

select name, avg(salary) from employees group by name;

Count of Employees with same name

select name, count(name) from employees

group by name;

Find count of all the employees grouped by age;

select age, count(id) from employees group by age;

**Having**

Where clause is used to put conditions on columns, Having clause is used to put conditions on groups

Only show those names which are repeated

select name, count(name) from employees

group by name having count(name) >1;

Total salaries paid in every department

select did, sum(salary) from employees group by did;