create table emp1

(name varchar(10),

ssn char(2),

primary key(ssn)

);

insert into emp1 values(‘seena’,11);

insert into emp1 values(‘veena’,22);

insert into emp1 values(‘aarav’,33);

insert into emp1 values(‘ansh’,44);

insert into emp1 values(‘megha’,55);

CONSTRAINTS

NOT NULL

CHECK

UNIQUE

DEFAULT

Primary key

Foreign Key

create table emp2

(name varchar(10) not null,

ssn char(2),

primary key(ssn)

);

Create table election

(

Name varchar(20) not null,

Age int,

ID char(4),

Primary key(ID));

Create table election3

(

Name varchar(20) not null,

Age int,

ID char(4),

**Check(age>=18),**

Primary key(ID));

Create table emp

( name varchar(20),

age int ,

emp \_id char(7),

salary varchar(10),

primary key(emp\_id),

**check( age >=24 and age <=58)**

);

// age if more than 58 ,then its retirement age

**DEFAULT CONSTRAINT**

**Create table** Emp

( name varchar(20) not null ,

Address varchar (30),

Ssn char(5),

Dno int not null **default** 11,

Primary Key(Ssn)

)

* If explicitly no values is given to **Dno,** by default the value 11 is taken.
* If **not null** and **default** constraint are not specified , then null values are taken for the attributes.

**BASIC QUERIES IN SQL**

*Queries on a single table*

1. Select \*

from emp1; // all rows and columns are displayed

2. Select \*

from emp1

where ssn = 11; // all cols but only rows that satisfy the condition

3. select name

from emp1; // only name col but all rows

4. select name

from emp1

where ssn= 22; // only name col and only those rows satisfying the condition

**ALTER command**

* Alters the table by adding or dropping of columns, adding or dropping of constraints

Syntax: Alter table <table\_name> Add column <col\_name> <data\_type>;

Eg: **Alter table** Emp **Add column** JOB\_NAME Varchar(20);

**Alter table** Emp **Drop column** JOB\_NAME **Cascade**;

**Alter table** Emp **Drop column** JOB\_NAME **Restrict**;

**Alter table** Dept **Alter Column** Mgr\_ssn **Drop** DEFAULT;

**Alter table** Emp **drop constraint** EMPSUP **Cascade**; // while creating table emp whatever constraint ,give name to the constraint

**BASIC QUERIES IN SQL**

*Queries on two or more tables*

1. Retreive the birth date and address of the employee(s) whose name is ‘ Amit G Halgekar’.

**SELECT** Bdate, Address //projection attributes

**FROM**  Emp

**WHERE** Fname = ‘Amit’ AND Minit = ‘G’ AND Lname = ‘Halgekar’; //selection condition

2. Retrieve the name and address of all employees who work for the “Research” department.

**SELECT** Name , Address

**FROM** Emp, Dept

**WHERE** Dname = ‘Research’ and Dnumber = Dno;

3. For every project located in ‘stafford’ , list the project number,the controlling dept number, and the dept *managers* last name, address and birth date.

**SELECT** Pnumber, Dnum, Lname,Address, Bdate

**FROM** Proj, Dept, Emp

**WHERE** Plocation = ‘stafford’ and Dnum=Dnumber and Mgr\_ssn=Ssn

**Aliasing** for ambiguous attribute names(when two different tables have same attribute names with same spelling)

*\*\*Suppose while creating employee and dept tables both have* ***Dno*** *as columns*

**SELECT** Name , Address

**FROM** Emp, Dept

**WHERE** Dname = ‘Research’ and Emp.Dno = Dept.Dno;

\*\* *Suppose while creating* *employee and dept tables both have Employee name as ‘****Name’*** *and dept name as also* ***‘Name’,*** *ambiguity arises.*

**SELECT** Name , Address

**FROM** Emp, Dept

**WHERE** Dept.Name = ‘Research’ and Emp.Dno = Dept.Dno;

* **Distinct and All**

1. **SELECT ALL** Salary

**FROM** Emp; // selects all salary rows with duplicate values too

2**. SELECT** **DISTINCT** Salary

**FROM** Emp; //selects only distinct salary rows with no duplicate values too

**\*\* Substring Pattern Matching and arithmetic operators**

String pattern matching allows comparison condition on parts of a character string, using LIKE comparison operator.

2 reserved characters % and an underscore ( \_ ) is used.

% replaces 0 or more characters

\_ replaces a single character

Example 1: Retrieve all employees whose address is in Houston, Texas.

**SELECT** Fname, Lname

**FROM** Emp

**WHERE** Address **LIKE** ‘%Houston,Texas%’ ;

Example 2 : Find all employees who were born during the 1950s.

**SELECT** Fname,Lname

**FROM** Emp

**WHERE** Bdate **LIKE** ‘\_ \_5\_ \_ \_ \_ \_ \_ \_’ ;

**\*\* Use of arithmetic operators**

Operators like + ,- , \* and / may be applied to attributes with numeric domains.

Example 1: Show the resulting salaries if every employee working on the ‘ProductX’ project is given a 10 percent raise.

**SELECT** Fname, Lname, 0.1\*Salary **AS** Increased\_Sal

**FROM** Emp, Works\_on,Project

**WHERE** Pname **=** ‘ProductX’ and Ssn=Essn and Pno=Pnumber

Example 2: Retrieve all employees in department 5 whose salary is in the range of 30,000 and 50,000.

**SELECT**  \*

**FROM** Emp

**WHERE** (Salary **BETWEEN** 30000 and 50000) and Dno=5;

//use of BETWEEN operator for comparison

**\*\* Ordering of Query Results**

Sql allows the rows/tuples in the result of a query to be ordered in some manner by the values of the attributes. ORDER BY clause is used, by default the ordering is ascending.

Example: Retrieve a list of employees and projects they are working on, ordered by department , and order the names alphabetically.

**SELECT** Dname, Fname, Lname, Pname

**FROM** Emp, Dept, Works\_on, Proj

**WHERE** Dnumber = Dno and Ssn = Essn and Pno= Pnumber

**ORDER BY** Dname,Lname, Fname;

**SELECT** Dname, Fname, Lname, Pname

**FROM** Emp, Dept, Works\_on, Proj

**WHERE** Dnumber = Dno and Ssn = Essn and Pno= Pnumber

**ORDER BY** Dname **desc**, Lname **asc**, Fname **asc**;

**\*\* UPDATE Command**

**Syntax:**

**UPDATE <table\_name>**

**SET <attribute\_new\_value>**

**WHERE <condition>;**

**Example: UPDATE Project**

**SET Plocation= ‘Houston’ , Dnum= 5**

**WHERE Pnumber=10;**

**//**The tuple with Pnumber 10 will be updated with new values and the existing values will be deleted.

\*\* **Delete Command**

Deletes tuples. It can delete all tuples leaving an empty table.

Drop deletes the entire table along with its definition.

**Delete from** emp

**where** Lname = ‘Brown’ ;

**Delete from** Emp

**Where** Ssn = ‘123456789’ ;

**Delete from** Emp; // deletes all tuples