Class: D10A Seat No.____

Roll No: 01

VIVEKANAND EDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

Hashu Advani Memorial Complex, Collector's Colony, R C Marg, Chembur, Mumbai-400074

CERTIFICATE

Certified that Mr./~~Miss~~ **Aamir Z Ansari** of **D10A** has satisfactorily completed a course of the necessary experiments/assignments in **Database Management System** under my supervision in the Institute of technology in the year 2020-2021

Principal Head of Department

Lab In-charge Subject Teacher

Vivekanand Education Society's Institute of Technology

Department of Information Technology

Lab Plan

Course Title: Database Management Systems **Year:-2020-2021**

Branch: INFT (III) **Class: D10A**

Faculty In-charge: Ms. Sukanya Roychowdhury

Lab In-charge: Ms. Sukanya Roychowdhury

Email : sukanya.roychowdhury@ves.ac.in

Course objectives:

- Learn and practice data modeling using the entity-relationship and developing database designs.
- Understand the use of Structured Query Language (SQL) and learn SQL syntax.
- Understand the needs of database processing and learn techniques for controlling the consequences of concurrent data access

Course Outcomes:

Student should be able to:

CO 1. Explain the features of database management systems and Relational database
CO 2. Design conceptual models of a database using ER modeling for real life applications and also construct queries in Relational Algebra.
CO 3. Create and populate a RDBMS for a real life application, with constraints and keys, using SQL.
CO 4. Retrieve any type of information from a database by formulating complex queries in SQL.
CO 5. Analyze the existing design of a database schema and apply concepts of normalization to design an optimal database.
CO 6. Build indexing mechanisms for efficient retrieval of information from a database

Vivekanand Education Society's Institute of Technology

Department of Information Technology

INDEX

Sr. No	Experiments	CO's	Grade
1.	Experiment to study different phases of database design. Design ER and EER diagram for company database	CO1,CO2	10
2.	To Map Er/EER to Relational Schema Model for Company Database	CO1,CO2,CO5	10
3.	Experiment to study DDL statements and Integrity constraint	CO3	10
4.	Experiment to study DML commands.	CO3	10
5.	Experiment to study Simple queries and Nested Queries.	CO2,CO4	10
6.	Experiment to study complex and Co-related queries	CO2,CO4	10
7.	Experiment to study different types of Joins.	CO4	10
8.	Experiment to study View.	CO4	10
9.	Execution of procedure and functions by using SQL Server.	CO4	10
10.	Assignment 1 Quiz	CO1, CO2, CO3, CO4, CO5	10

Subject In-charge **Overall Grade**

SDS Page No
Date / /

COMPANY DATABASE

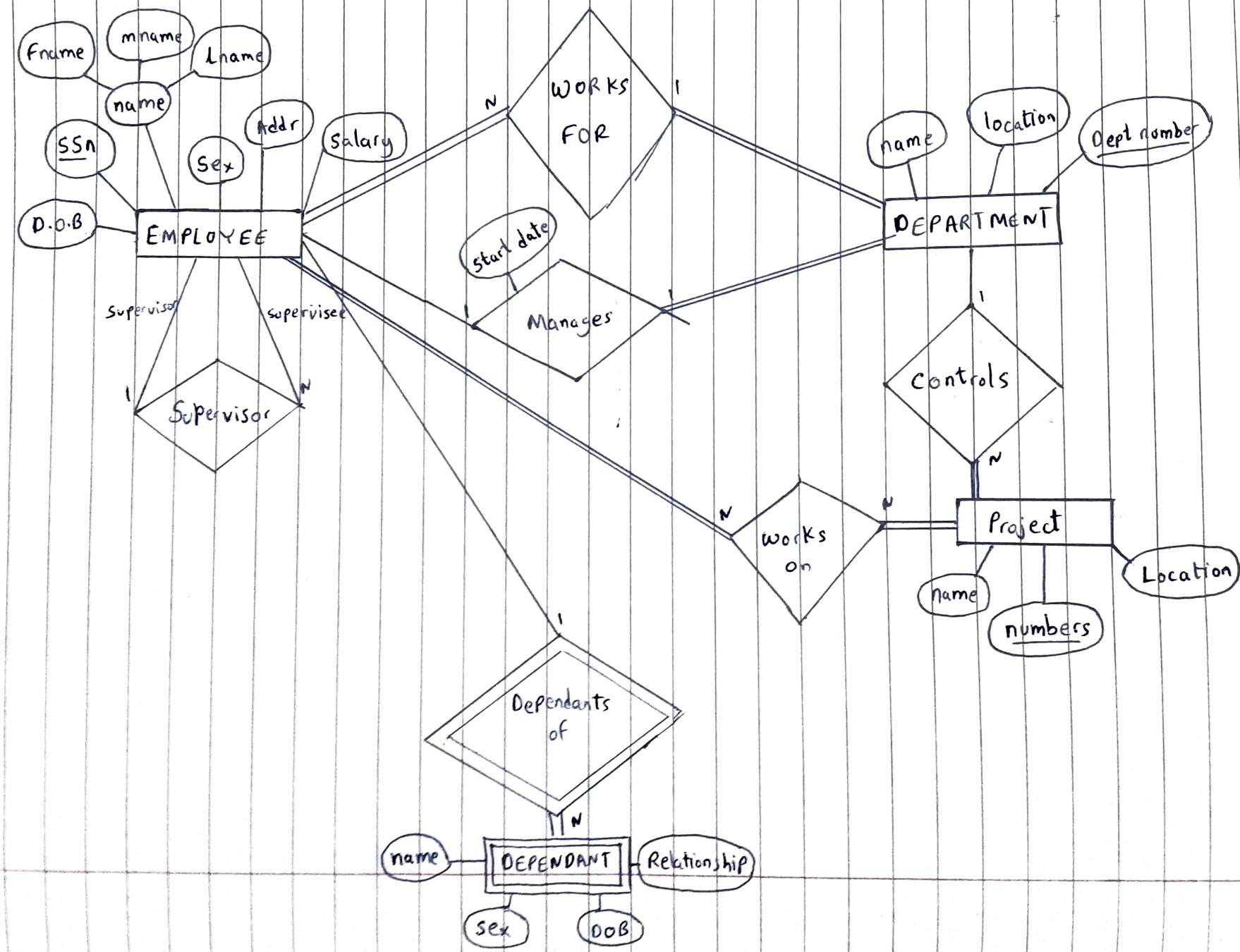
Aim:- Identify realworld problems and develop the problem statement (company database), and design an ER diagram for company database

Problem statement:-

Requirement of company are as follows

1) The company is orginathised into departments. Each department has a name, number, and an employe who manages the department. We Keep track of start date of department manager.
2) Each department controls a number of projects. Each project has a name, number and is located at a single location
3) We store each employess ssn, addr, salary, sex, and is located at a single location, salary, sex, DOB. Each employer works for one department but may work on several projects, we keep track of the hours per week that employee currently works on each project.

ER diagram of company database



Conclusion :-

Hence, we have identified the real world problem of a company and developed the problem statement, and also designed ER diagram for the sa

SDS Page No
Date

Aim :- To map ER/EER to Relational schema model for company Database

Theory :- Rules of ER to relational mapping

1) Mapping of regular entity types
For each strong entity type E in ER schema, create a relation R that includes all the simple attributes of E. Form a primary key from the attributes

2) Mapping of weak entity types
For each weak entity type W in ER schema with owner entity type E create a relation R and include all simple attributes of W as attribute of R

3) Mapping of binary 1-1 relation.
For each binary 1:1 relationship R in ER, identify the relations S and T that corresponds to the entity types participating in R. There are three possible approaches.
1) Foreign key approach
2) Merged relation operation
3) Cross-refferance of relationship relation option

4) Mapping of binary 1:N relation
For each regular binary 1:N relationship type R, identify relation S that represent the participating entity type at the N-side of relationship type

5) Mapping of binary M:N condition
For each regular binary M:N type R, create a new relation S to represent R Include as foreign key attribute in S the primary key of relation that represents participating entity types

EMPLOYEE

Fname	mname	lname	ssn	DOB	Salary	Addr	Sex	Super ssn	Dept no.

DEPARTMENT

Dept name	Dept no.	Mgr ssn	Mgr start date

DEPARTMENT LOCATION

Dept no	Dept location

PROJECT

Pname	Pnumber	P location	Dept no

WORKS ON

Essn	Pnumber	Hours

Dependant

Essn	Dependant name	Sex	D.O.B	Relationship

6) Mapping multivalued attributes

Multivalued attribute becomes another table in the relational schema. The primary key its parent table will become a part of composite primary key comprising itself and primary tabeles attribute

7) Mapping of composite attributes

Only the leaf / composite / child attributes of a composite attribute will become a collumn of respective table

Conclusion

Hence, we have successfully mapped the relational schema model for company database

DBMS
Lab Assignment number 3

Name: Aamir Ansari **Batch** A **Roll no.** 01

Aim: Experiment to study Data Definition Language Commands and Integrity constraints.

Theory:

The Data Definition Language (DDL) is used to create and destroy databases and database objects. These commands will primarily be used by database administrators during the setup and removal phases of a database project.
Let's take a look at the structure and usage of four basic DDL commands:

CREATE:
Create command is used to create different database objects like table,view etc.
Syntax: create table <table name>(<column name> <datatype>[size],….);

ALTER:
Once you've created a table within a database, you may wish to modify the definition of it.
The ALTER command allows you to make changes to the structure of a table without deleting and recreating it. Take a look at the following command:

DROP
The final command of the Data Definition Language, DROP, allows us to remove entire database objects from our DBMS. For example, if we want to permanently remove the personal_info table that we created, we'd use the following command:

Syntax: Drop table tablename;
Integrity constraints.

Integrity constraints:

1. NOT NULL Constraint: When column is defined as a NOT NULL then that column becomes mendatory column i.e value must be entered in that column for each row.
Syntax: columnname datatype(size) NOT NULL

2. Primary key constraint: This constraint ensures that
a)Data entered in the table column is unique across the entire table.
b)None of the cells belonging to the column are left emty.
Syntax: columnname datatype(size) Primary key

3. Unique key constraint: This constraint ensures that
a) Data entered in the table column is unique across the entire table.
b) Column can have NULL value.
Syntax: columnname datatype(size) Unique

4. Foreign key constraint: Foreign key constraint creates relationship between Records. Often we wish to ensure that a value appearing in a relation for a given set of attributes also appears for another set of attributes in another relation. This is called referential integrity This constraint ensures that
a) Record cannot be inserted into a detail if corresponding records in

the master table do not exist.

b) Records of the master table cannot be deleted if corresponding records in detail table exist.
Syntax: columnname datatype(size) references tablename(columnname)
On delete cascade

5. Check Constraint: Business rule validations can be applied column by using check constraint. Check constraint must be specified as a logical expression that evaluates either true or false.
Syntax: columnname datatype(size) CHECK(logical expression)

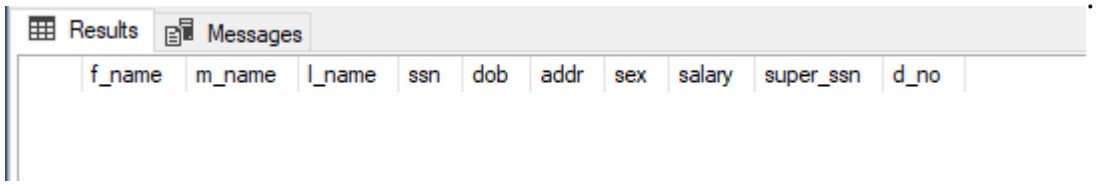
Defining Integrity Constraints with the CREATE TABLE Command

Example:

```
--Execution of different types of DDL Commands(create, Alter, Delete)
--create command
        create table customer(cid int, cname varchar(20), address char(10));
--Alter command(add | alter column | drop column)
--adding new column in existing table
        alter table customer add phno numeric(10)
--Modifying existing column
        alter table customer alter column address varchar(20)
--dropping column from existing table
        alter table customer drop column address
--Execution of different types of Integrity constraints
--Table Employee
        create table employee(ssn int primary key check(ssn like &#39;__&#39;),ename varchar(20)
        not null, salary money, superssn int foreign key references employee(ssn),
        dno int default 5)
--Table Department
        create table dept(dno int primary key, dname varchar(20), mgrssn int references
        employee, startdate datetime)
-- applying foreign key constarint on existing table
        alter table employee add constraint fk_dno foreign key(dno) references dept(dno)
--Table deptlocation
        create table deptloc(dno int,dloc varchar(20),primary key(dno,dloc),foreign key(dno)
        references dept(dno))
--Table Project
        create table project(pno int primary key,pname varchar(20),dno int references, dept(dno))
--Table workson
        create table workson(ssn int, pno int, noofhrs int, primary key(ssn,pno),
        foreign key(ssn) references employee(ssn) on delete cascade on update cascade,
        foreign key(pno) references project(pno) on delete cascade on update cascade)
--Table Dependent
        create table dependent(ssn int,depname varchar(20),relation varchar(20),
        primary key(ssn,depname),foreign key(ssn) references employee(ssn))
--droping primary key constraint from existing table
        alter table dependent drop constraint PK__dependent__3B0BC30C
--Applying constraint on existing table
        alter table dependent add constraint pk_ssn primary key(ssn,depname)
```

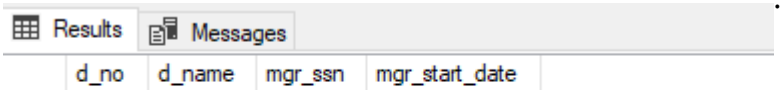
Code:

```
CREATE TABLE Employee (
        f_name VARCHAR(30) NOT NULL,
        m_name VARCHAR(30),
        l_name VARCHAR(30),
        ssn BIGINT NOT NULL,
        dob DATE,
        addr VARCHAR(50),
        sex CHAR(1),
        salary MONEY,
        super_ssn BIGINT NOT NULL,
        d_no INT,

        PRIMARY KEY (ssn),
        FOREIGN KEY (super_ssn) REFERENCES Employee(ssn)
);

SELECT * FROM Employee;
```



***

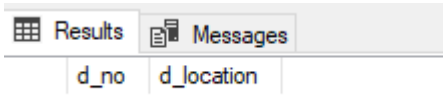
```
CREATE TABLE Department (
        d_no INT NOT NULL,
        d_name VARCHAR(20) NOT NULL,
        mgr_ssn BIGINT,
        mgr_start_date DATE,

        PRIMARY KEY (d_no),
        FOREIGN KEY (mgr_ssn) REFERENCES Employee(ssn)
);

SELECT * FROM Department;
```



***

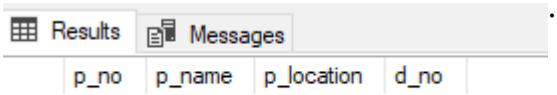
```
ALTER TABLE Employee
ADD FOREIGN KEY (d_no) REFERENCES Department(d_no);
```

```
CREATE TABLE Department_location (
        d_no INT NOT NULL,
        d_location VARCHAR(20) NOT NULL,

        PRIMARY KEY (d_no, d_location),
        FOREIGN KEY (d_no) REFERENCES Department (d_no)
);
SELECT * FROM Department_location;
```



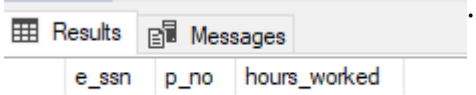
***

```
CREATE TABLE Project (
        p_no INT NOT NULL,
        p_name VARCHAR(20) NOT NULL,
        p_location VARCHAR(20) NOT NULL,
        d_no INT NOT NULL,

        PRIMARY KEY (p_no),
        FOREIGN KEY (d_no) REFERENCES Department (d_no)
);

SELECT * FROM Project;
```



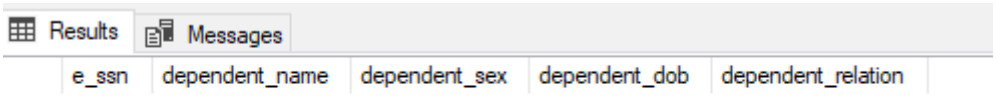
***

```
CREATE TABLE Works_on (
        e_ssn BIGINT NOT NULL,
        p_no INT NOT NULL,
        hours_worked FLOAT,

        PRIMARY KEY (e_ssn, p_no),
        FOREIGN KEY (e_ssn) REFERENCES Employee (ssn),
        FOREIGN KEY (p_no) REFERENCES Project (p_no)
);

SELECT * FROM Works_on;
```



***

```
CREATE TABLE Dependant (
        e_ssn BIGINT NOT NULL,
        dependent_name VARCHAR(30) NOT NULL,
        dependent_sex CHAR(1),
        dependent_dob DATE,
        dependent_relation VARCHAR(20),

        PRIMARY KEY (e_ssn, dependent_name),
        FOREIGN KEY (e_ssn) REFERENCES Employee (ssn)
);
SELECT * FROM Dependant;
```



***

Conclusion: Thus we have implemented different DDL Commands and Integrity constraints successfully.

DBMS LAB
Lab Experiment number 04

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Experiment to study Data Manipulation Language Commands.

Theory:

The Data Manipulation Language (DML) is used to retrieve, insert and modify database information. These commands will be used by all database users during the routine operation of the database. Let's take a brief look at the basic DML commands:

INSERT

The INSERT command in SQL is used to add records to an existing table. Returning to the personal_info example from the previous section, let's imagine that our HR department needs
to add a new employee to their database. They could use a command similar to the one shown
below:
Syntax: insert into table tablename values(values);
Example:

INSERT INTO employee
values('bart', 'simpson', 12345, $45000)
These correspond to the table attributes in the order they were defined: first_name, last_name, employee_id, and salary.

SELECT

The SELECT command is the most commonly used command in SQL. It allows database users to retrieve the specific information they desire from an operational database. Let's take
a look at a few examples, again using the personal_info table from our employees database.
Syntax: select <attribute list> from <list if tables> where predicate;
Example:

SELECT * FROM employee
Alternatively, users may want to limit the attributes that are retrieved from the database. For example, the Human Resources department may require a list of the last names of all employees in the company. The following SQL command would retrieve only that information.
SELECT last_name FROM employee
Finally, the WHERE clause can be used to limit the records that are retrieved to those that meet specified criteria.

SELECT *FROM employee WHERE salary > 50000

UPDATE

The UPDATE command can be used to modify information contained within a table, either in bulk or individually.

Syntax: update tablename set predicate;

Example:

UPDATE employee SET salary = salary * 1.03

On the other hand, our new employee Bart Simpson has demonstrated performance above and beyond the call of duty. Management wishes to recognize his stellar accomplishments with a $5,000 raise. The WHERE clause could be used to single out Bart for this raise:

UPDATE employee SET salary = salary + $5000
WHERE employee_id = 12345

Give 10 % raise in salary who are working on railway project and working for IT department

Update employee set salary=salary*1.1 where dno=(select dno from dept where dname=”it”)
and ssn=(select ssn from workson where pno=(select pno from project where pname=”railway”))

DELETE

The DELETE command with a WHERE clause can be used to remove his record from the employee table:

Syntax: delete from tablename where predicate;

Example:

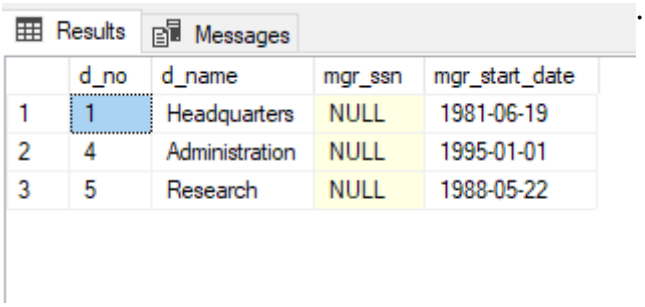
DELETE FROM employee WHERE employee_id = 12345

Delete employees working for IT department

Delete from employee where dno=(select dno from dept where dname=”IT”)

Code:

```
-- Insert value in department table
INSERT INTO Department (d_no, d_name, mgr_ssn, mgr_start_date)
VALUES (
        5,
        'Research',
        NULL,
        '1988-05-22'
),
(
        4,
        'Administration',
        NULL,
        '1995-01-01'
),
(
        1,
        'Headquarters',
        NULL,
        '1981-06-19'
)
SELECT * FROM Department;
```


Results Messages

	d_no	d_name	mgr_ssn	mgr_start_date
1	1	Headquarters	NULL	1981-06-19
2	4	Administration	NULL	1995-01-01
3	5	Research	NULL	1988-05-22

```
-- Insert values in employee table
INSERT INTO Employee (f_name, m_name, l_name, ssn, dob, addr, sex, salary, super_ssn, d_no)
VALUES (
        'John', 'B', 'Smith',
        123456789,
        '1965-01-09',
        '731 Fondren, Houston, TX',
        'M',
        30000,
        NULL,
        5
),
(
```

```
        'Franklin', 'T', 'Wong',
        33344555,
        '1955-12-08',
        '638 Voss, Houston, TX',
        'M',
        40000,
        NULL,
        5
),
(
        'Alicia', 'J', 'Zelaya',
        999887777,
        '1968-01-19',
        '3321 Castle, Spring, TX',
        'F',
        25000,
        NULL,
        4
),
(
        'Jennifer', 'S', 'Wallace',
        987654321,
        '1941-06-20',
        '291 Berrym, Bellaire, TX',
        'F',
        43000,
        NULL,
        4
),
(
        'Ramesh', 'K', 'Narayan',
        666884444,
        '1962-09-15',
        '975 Fire Oak, Humble, TX',
        'M',
        38000,
        NULL,
        5
),
(
        'Joyce', 'A', 'English',
        453453453,
        '1972-07-31',
        '5361 Rice, Houston, TX',
        'F',
        25000,
```

```
        NULL,
        5
),
(
        'Ahmad', 'V', 'Jabbar',
        987987987,
        '1969-03-29',
        '980 Dallas, Houston, TX',
        'M',
        25000,
        NULL,
        4
),
(
        'James', 'E', 'Borg',
        888665555,
        '1937-11-10',
        '450 Stone, Houston, TX',
        'M',
        55000,
        NULL,
        1
)
SELECT * FROM Employee;
```

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	NULL	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	NULL	5
3	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	NULL	5
4	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	NULL	5
5	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
6	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	NULL	4
7	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	NULL	4
8	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	NULL	4

.

```
-- Insert foreign key super_ssn in Emplyee table
UPDATE Employee SET super_ssn = 333445555
WHERE ssn = 123456789;

UPDATE Employee SET super_ssn = 888665555
WHERE ssn = 333445555;

UPDATE Employee SET super_ssn = 987654321
WHERE ssn = 999887777;
```

```
UPDATE Employee SET super_ssn = 888665555
WHERE ssn = 987654321;

UPDATE Employee SET super_ssn = 333445555
WHERE ssn = 666884444;

UPDATE Employee SET super_ssn = 333445555
WHERE ssn = 453453453;

UPDATE Employee SET super_ssn = 987654321
WHERE ssn = 987987987;

SELECT * FROM Employee
```

Results | Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	333445555	5
4	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
5	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
6	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4
7	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4
8	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

.

```
-- Insert foreign key mgs_ssn in Department Table
UPDATE Department SET mgr_ssn = 333445555
WHERE d_no = 5;

UPDATE Department SET mgr_ssn = 987654321
WHERE d_no = 4;

UPDATE Department SET mgr_ssn = 888665555
WHERE d_no = 1;

SELECT * FROM Department
```

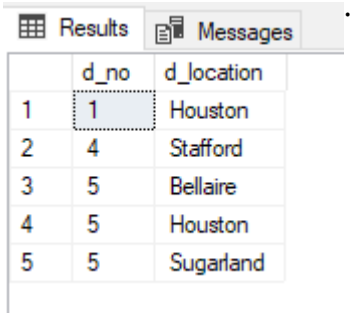
.

Results | Messages

	d_no	d_name	mgr_ssn	mgr_start_date
1	1	Headquarters	888665555	1981-06-19
2	4	Administration	987654321	1995-01-01
3	5	Research	333445555	1988-05-22

```
-- Insert values in Department_location table
```

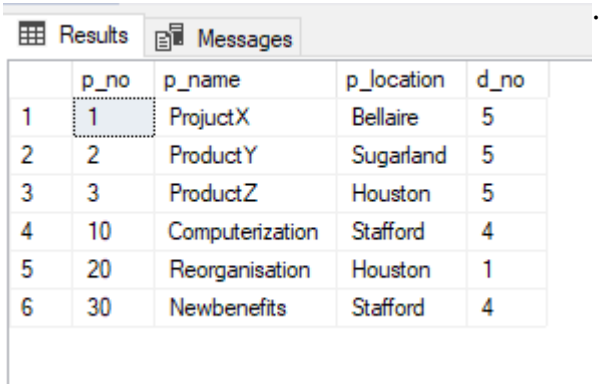
```
INSERT INTO Department_location(d_no, d_location)
VALUES (
        1, 'Houston'
),
(
        4, 'Stafford'
),
(
        5, 'Bellaire'
),
(
        5, 'Sugarland'
),
(
        5, 'Houston'
)
SELCT * FROM  Department_location;
```



Results | Messages

	d_no	d_location
1	1	Houston
2	4	Stafford
3	5	Bellaire
4	5	Houston
5	5	Sugarland

```
-- Insert values in Project table
INSERT INTO Project(p_no, p_name, p_location, d_no)
VALUES (
        1, 'ProjuctX', 'Bellaire', 5
),
(
        2, 'ProductY', 'Sugarland', 5
),
(
        3, 'ProductZ', 'Houston', 5
),
(
        10, 'Computerization', 'Stafford', 4
),
(
        20, 'Reorganisation', 'Houston', 1
),
(
        30, 'Newbenefits', 'Stafford', 4 )
SELECT * FROM Project
```



Results | Messages

	p_no	p_name	p_location	d_no
1	1	ProjuctX	Bellaire	5
2	2	ProductY	Sugarland	5
3	3	ProductZ	Houston	5
4	10	Computerization	Stafford	4
5	20	Reorganisation	Houston	1
6	30	Newbenefits	Stafford	4

```
-- Insert values in Works_on table
INSERT INTO Works_on (e_ssn, p_no, hours_worked)
VALUES (
        123456789, 1, 32.5
),
(
        123456789, 2, 7.5
),
(
        666884444, 3, 40.0
),
(
        453453453, 1, 20.0
),
(
        453453453, 2, 10.0
),
(
        333445555, 2, 10.0
),
(
        333445555, 3, 10.0
),
(
        333445555, 10, 10.0
),
(
        333445555, 20, 10.0
),
(
        999887777, 30, 30.0
),
(
        999887777, 10, 10.0
),
```

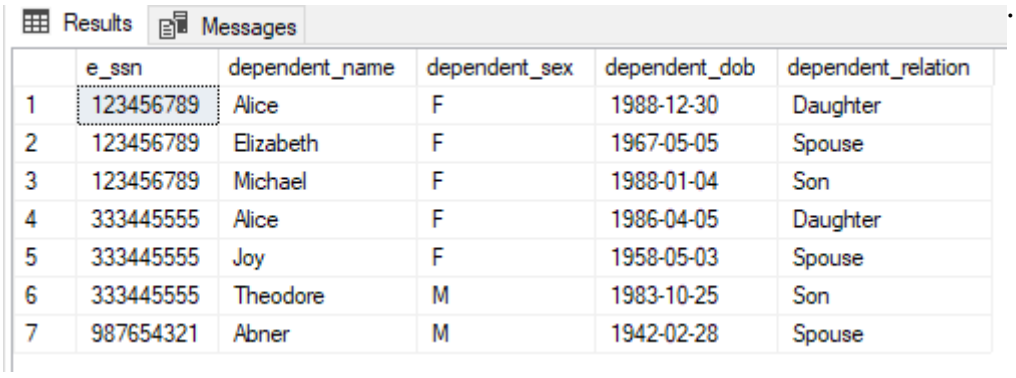
```
(
        987987987, 10, 35.0
),
(
        987987987, 30, 3.0
),
(
        987654321, 30, 20.0
),
(
        987654321, 20, 15.0
),
(
        888665555, 20, NULL
)
SELECT * FROM Works_on
```

Results | Messages

	e_ssn	p_no	hours_worked
1	123456789	1	32.5
2	123456789	2	7.5
3	333445555	2	10
4	333445555	3	10
5	333445555	10	10
6	333445555	20	10
7	453453453	1	20
8	453453453	2	10
9	666884444	3	40
10	888665555	20	NULL
11	987654321	20	15
12	987654321	30	20
13	987987987	10	35
14	987987987	30	3
15	999887777	10	10
16	999887777	30	30

```
--Insert values into Dependant table
INSERT INTO Dependant(e_ssn, dependent_name, dependent_sex, dependent_dob,
dependent_relation)
VALUES (
        333445555, 'Alice',
        'F',
        '1986-04-05',
        'Daughter'
),
(
        333445555, 'Theodore',
        'M',
```

```
        '1983-10-25',
        'Son'
),
(
        333445555, 'Joy',
        'F',
        '1958-05-03',
        'Spouse'
),
(
        987654321, 'Abner',
        'M',
        '1942-02-28',
        'Spouse'
),
(
        123456789, 'Michael',
        'F',
        '1988-01-04',
        'Son'
),
(
        123456789, 'Alice',
        'F',
        '1988-12-30',
        'Daughter'
),
(
        123456789, 'Elizabeth',
        'F',
        '1967-05-05',
        'Spouse'
)
```



Results | Messages

	e_ssn	dependent_name	dependent_sex	dependent_dob	dependent_relation
1	123456789	Alice	F	1988-12-30	Daughter
2	123456789	Elizabeth	F	1967-05-05	Spouse
3	123456789	Michael	F	1988-01-04	Son
4	333445555	Alice	F	1986-04-05	Daughter
5	333445555	Joy	F	1958-05-03	Spouse
6	333445555	Theodore	M	1983-10-25	Son
7	987654321	Abner	M	1942-02-28	Spouse

Conclusion: Thus we have implemented different DML commands successfully.

DBMS LAB

Lab Assignment number 05

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Experiment to study Simple and nested queries.

Theory:

Query is nothing but question asked to the database.

Select statement:

Select command is used to retrieve data from database. The result is stored in the temporary table called result set.

General Syntax:

Select A1, A2, A3-----An from R1, R2------Rn where Predicate

Eg. Consider table Employee (eid, ename, salary, dno)

Select ename , salary from employee

Selecting all columns:-

Select *from ‘tablename&gt’

Select *from employee

Choosing selected columns:-

Select A1, A2, A3-----An from tablename

Select eid,salary from employee

Computed values in select List:-

Select eid,salary*1.2 as increase_sal from employee

Filtering Rows from a table:-

Where clause:-

Where clause is used to extract only those records that fulfill specified condition.

Where clause is used to filter records.

Select ‘columnname&gt’ from ‘tablename&gt’ where ‘condition&gt’

Select empid from employee where salary>20000

Distinct clause:- Eliminating duplicates

Distinct keyword is used to eliminate duplicate rows from the resultset.

Select distinct dno from employee

By default – all

Select all dno from employee

Top clause:- Top clause limits the number of rows returned in the resultset.

Syntax:- Select top n ‘columname&gt’ from ‘tablename&gt’

-- Display first three records from resultset(employee)

Select top 3 * from emp

-- Display salaries of first two records from resultset

Select top 2 salary from emp

Operators in SQL:

Relational operators:- ,> ,’ ,’=, >=,’>(!=)

Logical Operators:- and , or ,not

Range searching Operators(between ,not between)

Between operator :- is used to specify range of values.

Syntax:- select columname from tablename where columnname between lowerlimit and upperlimit

Eg. Display emp details from employees where salary is within range 10000 and 20000.

Select *from employee where salary between 10000 and 20000

Select *from employee where salary =10000 and salary =20000

Display emp details from employee where salary is not Within range between 10000 and 20000

Select *from employee where salary not between 10000 nad 20000

Pattern matching operator:- Like ,Not Like

Like operator:

Like operator is used in a where clause to search for a specified pattern in a column.

Pattern is specified as a string in single quotes.

General Syntax:-

Select colnames from tablename where colname like ‘pattern’

SQL wild characters:-

SQL wild characters can substitute for one or more characters

When searching for data in database.

SQL wild characters must be used with Like and not like operator.

Wild character Description

% A substitute for 0 or more characters.

_ A substitute for exactly one characters

[range] A single character within range [A-D]

[^range] A single character not within range

Eg. Select *from emp where ename like ‘A%’

Select *from emp where ename like ‘%A%’

Select *from emp where eid like ‘__’

Select *from emp where eid like ‘__%’

Select *from emp where ename like ‘_[e-f]%’

Set Membership operators:- In and Not In

In operator:-

In operator is used to check membership of value in given set

Not in :-

Not In operator is used to check absence of membership of value in given set

Syntax:

Select colname from tablename where colname in(value1,value2…..)

Eg. Select *from emp where dno in(20,10,25)

Select *from emp where dno not in(20,10,25)

Nested Queries:-

Nested query is a query that has another query embedded within it.The embedded query is called a subquery.

Some queries requires that data to be fetched from database and then used in comparison

Condition.

When we use Select----from----where block within where clause of another query such type of query is called Nested queries.

General syntax:

Outer query(parent query)

Select ‘columnname> from ‘tablename> where col= (select ------ from ‘tablename> where ‘condition>)

Subquery typically appears within a search condition as a part of the where or having clause of a select, update, delete.

Set comparision operators(all, some|any):-

All operator :- All operator is used to compare value of a column with all values from given set.

Some / Any :- Some / Any operator is used to compare value of a column with some of the values from given set.

--Set comparision operators are always used with relational operators.

Eg. >all,>=all, <all, <=all, <>all, <some, =some, <>some, <=some, >some----------etc.

=some equivalent to in, <>some equivalent to not in.

Eg. Retrive employee details of employees those are working in
It department.

empid	ename	salary	dno
1	John	45000.0000	101
5	smit	35000.0000	101
15	Nisha	40000.0000	103
10	Neha	25000.0000	102
2	smita	42000.0000	101

dno	dname
101	comp
102	it

select *from emp where dno =(select dno from dept where dname ='it')
(102)

empid	ename	salary	dno
10	Neha	25000.0000	102

Eg. Retrive employee details of employees those are working It
department or comp department.

select *from emp where dno in(select dno from dept where dname in ('it', 'comp')
(101,102)

List employee details of employees earning salary more than salary of emp with empid 5.

select *from emp where salary >(select salary from emp where empid=5)
35000

empid	ename	salary	dno
1	John	45000.0000	101
15	Nisha	40000.0000	103
2	smita	42000.0000	101

List employee details of employees earning salary more than salary of every emp working for dept no 101.
select *from emp where salary >=all(select salary from emp where dno=101)
(45000,35000,42000)

empid	ename	salary	dno
1	John	45000.0000	101

select *from emp where salary >=(select salary from emp where dno=101) --error

List employee details of employees earning salary more than salary of some of emps working for dept no 101.

select *from emp where salary >=some(select salary from emp where dno=101)

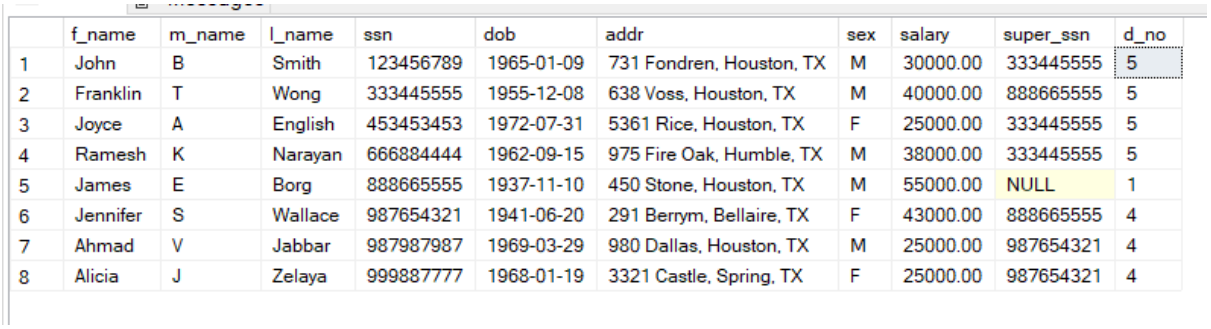
empid	ename	salary	dno
1	John	45000.0000	101
5	smit	35000.0000	101
15	Nisha	40000.0000	103
2	smita	42000.0000	101

--List employee details of employees earning salary more than salary of some of emps working for IT department.
select *from emp where salary >=some(select salary from emp where dno=(select dno from dept where dname='it'))

--List employee details of employees earning salary more than salary of some of emps working for IT department or comp dept.
select *from emp where salary >=some(select salary from emp where dno in (select dno from dept where dname in ('it',’comp’)))

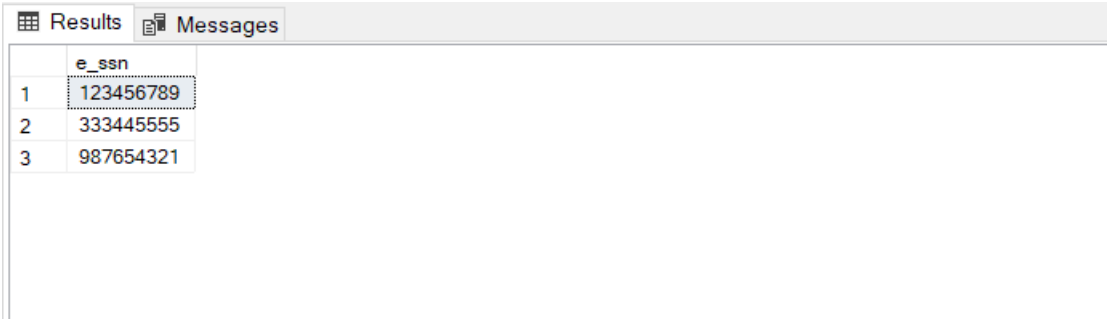
Code:

--1 Salary greater than 20,000
SELECT * FROM Employee WHERE salary > 20000;



	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	333445555	5
4	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
5	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
6	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4
7	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4
8	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

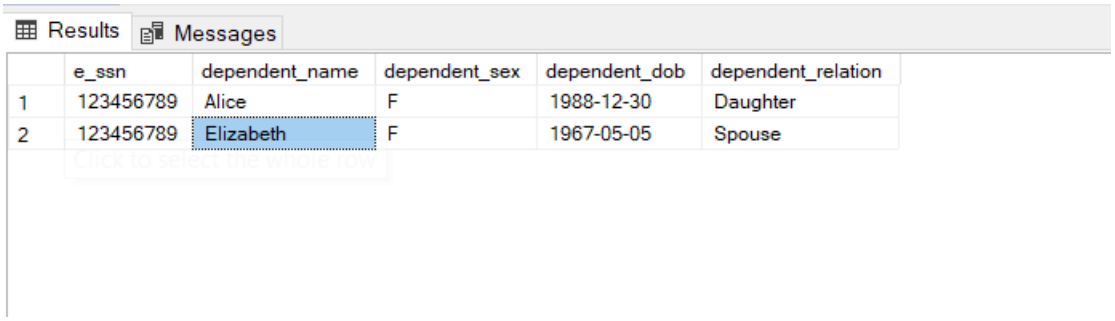
--2 Distinct ssn for employee who have dependant
SELECT DISTINCT e_ssn FROM Dependant;



Results Messages

	e_ssn
1	123456789
2	333445555
3	987654321

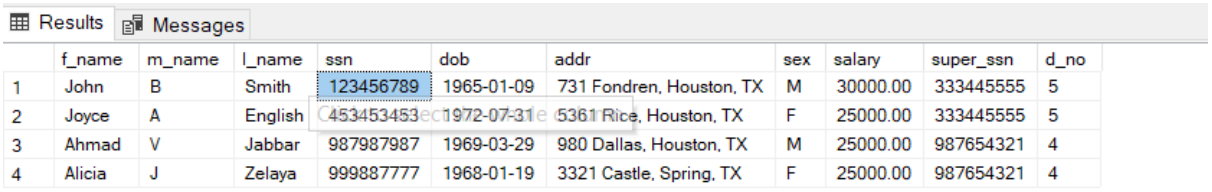
--3 First two dependants
SELECT TOP 2 * FROM Dependant;



Results Messages

	e_ssn	dependent_name	dependent_sex	dependent_dob	dependent_relation
1	123456789	Alice	F	1988-12-30	Daughter
2	123456789	Elizabeth	F	1967-05-05	Spouse

--4 Employee with salary between 10,000 and 30,000
SELECT * FROM Employee WHERE salary BETWEEN 10000 AND 30000;



Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
2	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	333445555	5
3	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4
4	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

--5 Employee with salary between 10,000 and 30,000
SELECT * FROM Employee WHERE salary NOT BETWEEN 10000 AND 30000;

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
3	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
4	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

--6A Employee with name starting with A
SELECT * FROM Employee WHERE f_name LIKE 'A%';

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4
2	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

--6B Employee with name ending with A
SELECT * FROM Employee WHERE f_name LIKE '%a';

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

--6C Employee with name ending with A
SELECT * FROM Employee WHERE f_name like '%[f,e]';

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	333445555	5

--7 Employee in department 1 and 4
SELECT * FROM Employee WHERE d_no = 1 OR d_no = 4;

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
2	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4
3	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4
4	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

--8 Employee working in research
SELECT * FROM Employee WHERE d_no = (SELECT d_no FROM Department WHERE d_name = 'Research');

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	333445555	5
4	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5

--9 Employee with more salary than a particular employee
SELECT * FROM Employee WHERE salary > (SELECT salary FROM Employee WHERE ssn = 123456789);

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
3	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
4	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

--10 Employee with more salary than max of department 5
SELECT * FROM Employee WHERE salary > (SELECT MAX(salary) FROM Employee WHERE d_no = 5);

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
2	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

--11 Employee with more salary than min of Administration department
SELECT * FROM Employee WHERE salary > (SELECT MIN(salary) FROM Employee WHERE d_no = (SELECT d_no FROM Department WHERE d_name = 'Administration'));

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
4	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
5	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

--12 Employee with more salary than min of Administration or Research
SELECT * FROM Employee WHERE salary > (SELECT MIN(salary) FROM Employee WHERE d_no IN (SELECT d_no FROM Department WHERE d_name IN ('Administration', 'Research')));

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000.00	333445555	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
4	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
5	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

Conclusion: We have successfully studied and implemented Simple and nested queries.

DBMS LAB

Lab Assignment number 06

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Experiment to study co-related queries and complex queries.

Theory:

Aggregate functions: -
Which takes set of values as input and returns single value as output.
Aggregate function generates summary value.

Sum():-The sum function returns the sum of values from given set.
Syntax:-
Select sum(distinct|all colname) from tablename
Default-all
Eg. Select sum(distinct salary) as totalsal from employee

Avg():-The Avg() function returns the average of values from given set.
Syntax:-
Select Avg(distinct|all colname) from tablename
Default-all

Eg. Select avg(salary) as Avgsal from employee

count():-The count() function returns number of values from given set.
Syntax:-
Select count(distinct|all colname) from tablename
Select count(*) from employee

↓
Number of records

Min():-The min() function returns minimum(smallest) value from given set.
Max():-The max() function returns maximum(largest) value from given set.

--All aggregate functions by default ignores null value.
Min(),Max(),count() can be used with numeric, character,date/time columns.
Group by clause:-
Group by clause is used to form group the records based on some criteria.
Group by clause applies aggregate function on subgroup of tuples from a relation where subgroups are formed on some column value.
Eg. group by colname
--Column name is called grouping column.

empid	ename	salary	dno
1	John	45000.0000	101
5	smit	35000.0000	101
15	Nisha	40000.0000	103
10	Neha	25000.0000	102
2	smita	42000.0000	101

--after grouping

empid	ename	salary	dno
1	John	45000.0000	101
5	smit	35000.0000	101
2	smita	42000.0000	101
15	Nisha	40000.0000	103
10	Neha	25000.0000	102

Select sum(salary) as totalsal from emp group by dno

totalsal
122000.0000
25000.0000
40000.0000

select dno,sum(salary) as totalsal from emp group by dno

	dno	totalsal
1	101	122000.0000
2	102	25000.0000
3	103	40000.0000

Select dno,sum(salary) as totalsal, count(*) as no_of_emp from emp group by dno

	dno	totalsal	no_of_emp
1	101	122000.0000	3
2	102	25000.0000	1
3	103	40000.0000	1

--There are restrictions on columns that we can specify in the
select list.The only columns allowed in the select list are
Grouping columns(column that we included after group by clause.
Aggregate function.

Select dno, salary as totalsal from emp group by dno --error

↓

(not aggregate function | not grouping column)

Eg. Display dno and number of employees for every
Department those are earning salary more than 40000.

Select dno,count(empid) from emp where salary>40000
group by dno

dno	totalsal	no_of_emp
101	87000.0000	2

Having clause:-
Having clause is used to apply condition on group rather than tuple.
Where clause can not apply condition on group of value.

Eg. Display dno and average salary for every dept.
select dno,avg(salary) as avg_sal from emp group by dno

	dno	avg_sal
1	101	40666.6666
2	102	25000.0000
3	103	40000.0000

Eg. Display dno and average salary of departments where average salary of department is >=40000.
select dno,avg(salary) as avg_sal from emp group by dno having avg(salary)>=40000

dno	avg_sal
101	40666.6666
103	40000.0000

Eg. Display dno and no_of_emp of those depts where no_of_emp those are earning salary within range 30000 to 40000 are at least 2.

select dno,count(empid) as no_of_emp from emp where salary between 30000 and 45000
group by dno having count(empid)>=2

dno	no_of_emp
101	3

Where Group By Having

Co-related queries:-
Whenever a condition in the where clause of a nested query references to some attributes of a relation declared in the outer query then two queries are said to be co-related.
Correlated subquery runs once for each row selected by the outer query. It contains a reference to a value from the row selected by the outer query.

Eg. Display employee details of employees who are earning salary more than average salary of department for which they are working.

empid	ename	salary	dno
1	John	45000.0000	101
5	smit	35000.0000	101
15	Nisha	40000.0000	103
10	Neha	25000.0000	102
2	smita	42000.0000	101

select dno,avg(salary) from emp group by dno

dno	(No column name)
101	40666.6666
102	25000.0000
103	40000.0000

select *from emp as e1 where e1.salary>

(select avg(salary) from emp as e2 where e1.dno=e2.dno)

empid	ename	salary	dno
2	smita	42000.0000	101
1	John	45000.0000	101

Nested subquery runs only once for the entire nesting (outer) query. It does not contain any reference to the outer query row.
Correlated subquery runs once for each row selected by the outer query

select *from emp as e1 where e1.salary>=all
(select salary from emp as e2 where e1.dno=e2.dno)

empid	ename	salary	dno
1	John	45000.0000	101
15	Nisha	40000.0000	103
10	Neha	25000.0000	102

select *from emp where salary
in (select max(salary) from emp group by dno)

ename	salary	dno
John	45000.0000	101
Nisha	40000.0000	103
Neha	25000.0000	102

Exists and Not Exists:-
Exists and Not Exists operator is used to check where the result of co-related nested query is empty or not.
Exists:-
Exists operator return true if there is at least one tuple in the output of subquery.
Not Exists: -
Not Exists operator return true if there are no tuple in the output of subquery.(empty)
select *from dept as d where exists(select *from emp as e where e.dno=d.dno)
select *from dept as d where not exists(select *from emp as e where e.dno=d.dno)
Derived Relations
SQL allows a subquery expression to be used in the from clause.
If such an expression is used, the result relation must be given a name, and the attributes can be renamed.
Eg.
select dno from (select dno,count(*) from emp group by dno) as deptinfo(dno,noofemp) where noofemp>1

dno
101

Complex Queries

-- Calculate avg,min,max,total salary and no of emp in each dept.
SELECT d_no, AVG(salary) as avg_salary, MAX(salary) as min_salary, MIN(salary) as min_salary, SUM(salary) as total_salary, COUNT(*) as no_of_employees
FROM Employee
GROUP BY d_no;

Results Messages

	d_no	avg_salary	min_salary	min_salary	total_salary	no_of_employees
1	1	55000.00	55000.00	55000.00	55000.00	1
2	4	31000.00	43000.00	25000.00	93000.00	3
3	5	33250.00	40000.00	25000.00	133000.00	4

-- Display dept no and number of emp for every dept those are earning salary more than 40K.
SELECT d_no, COUNT(*) as no_of_employees
FROM Employee
WHERE salary > 40000
GROUP BY d_no;

Results Messages

	d_no	no_of_employees
1	1	1
2	4	1

-- Display d_no, avg salary of department whose average salary is greater 40000
SELECT d_no, AVG(salary) as avg_salary
FROM Employee
GROUP BY d_no
HAVING (AVG(salary) > 40000);

Results Messages

	d_no	avg_salary
1	1	55000.00

-- Display dno and no of emp of those dept where no of emp those are earning salary with-in the range 30k and 40k are atleast 2.
SELECT d_no, COUNT(*) as no_of_employee
FROM Employee
WHERE salary BETWEEN 30000 AND 40000
GROUP BY d_no
HAVING(COUNT(*) >= 2);

Results Messages

	d_no	no_of_employee
1	5	3

Co-related Query and nested

```
-- Display emp details of emp with Maximum salary in each dept
SELECT *
FROM Employee
WHERE salary IN (
                    SELECT MAX(salary)
                    FROM Employee
                    GROUP BY d_no
                );
```

Results | Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
3	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

Co-related Queries

```
-- Display emp details of emp earning salary more than avg salary of dept for which they are working.
SELECT *
FROM Employee AS e1
WHERE e1.salary > (
                    SELECT AVG(salary)
                    FROM Employee AS e2
                    WHERE e1.d_no = e2.d_no
                );
```

Results | Messages

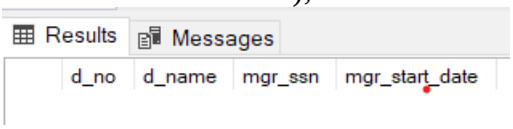
	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5

```
-- Display the details of those dept which have emp working for them.
SELECT *
FROM Department AS d
WHERE EXISTS (
                    SELECT *
                    FROM Employee AS e
                    WHERE e.d_no = d.d_no
                );
```

Results | Messages

	d_no	d_name	mgr_ssn	mgr_start_date
1	1	Headquarters	888665555	1981-06-19
2	4	Administration	987654321	1995-01-01
3	5	Research	333445555	1988-05-22

```
-- Display the details of those dept which have no emp working for them.
SELECT *
FROM Department AS d
WHERE NOT EXISTS (
                        SELECT *
                        FROM Employee AS e
                        WHERE e.d_no = d.d_no
                );
```



Conclusion: We have successfully studied and implemented co-related queries and complex queries.

DBMS LAB

Lab Assignment number 07

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Experiment to study views.

Theory:

View:-

In SQL view is a table that is derived from other existing base table or existing views.

Unlike ordinary tables (base tables) in a relational datatbase a view does not form part of the physical schema

A view does not exists in physical form it is considered as a virtual table.

Views are nothing but saved SQL statements, and are sometimes referred as "Virtual Tables"

Specification of view in SQL:-

1. CREATE VIEW View_Name AS query statement
2. CREATE VIEW View_Name(list of columns) AS query

Statemen

```
create view deptavg as select dno,avg(salary) as avg_sal
from emp group by dno
```

or

```
create view deptavg(dno,avgsal) as select dno,avg(salary)
from emp group by dno
```

select *from deptavg

dno	avg_sal
101	40666.6666
102	25000.0000
103	40000.0000

--Retrieve employee details of employees those are earning
Salary more than average salary of dno 101.

-- By using view

```
select *From emp where salary>(select avg_sal from deptavg
where dno=101)
```

40666.6666

-- Without using view

```
select *From emp where salary>(select avg(salary) from emp where  dno=101)
```

Eg. create view empinfo as select ssn,dno from emp

```
select *From empinfo
Insert into empinfo(20,103)
```

ssn	dno
1	101
2	101
10	102
5	101
15	103

ssn	ename	salary	dno	superssn
10	Neha	25000.0000	102	NULL
2	smita	42000.0000	101	10
1	John	45000.0000	101	10
5	smit	35000.0000	101	15
15	Nisha	40000.0000	103	10
5	NULL	NULL	105	NULL

Delete from empinfo where ssn=10

--Create view deptinfo which includes columnsdno,dname,

No_of_emp,avgsal of every dept.

create view deptinfo as select e.dno,dname,avg(salary) as avgsal,count(*) as noofemp from emp as e,dept as d where e.dno=d.dno group by e.dno,dname

select *from deptinfo

dno	dname	avgsal	noofemp
101	comp	40666.6666	3
102	it	25000.0000	1

Retrieve employee details of employees those are earning

Salary more than average salary of IT department.

--By using view

select *From emp where salary>(select avgSal from deptinfo where dname='it')

25000.0000

ssn	ename	salary	dno	superssn
1	John	45000.0000	101	10
2	smita	42000.0000	101	10
5	smit	35000.0000	101	15
15	Nisha	40000.0000	103	10

--without using view

select *From emp where salary>(select avg(salary) from emp where dno=(Select dno from dept where dname='it'))

Advantages of view:-

View can provide extra security.

View can provide abstraction so database users can create abstraction by using view

Views can hide the complexity of data.

Views take very little space to store; the database contains only the definition of a view, not a copy of all the data it presents(view saves memory)

Views can act as aggregated tables, where the database engine aggregates data (sum, average, etc) and presents the calculated results as part of the data(views simplifies certain queries)

How views are stored?

When we define a view the database system stores the definition of view itself rather than the result of the evaluation of query expression.

Whenever a view definition appears in a query it is replaced by stored query expression and gets recomputed.

Types of views.

1.Updatable Views

2. Read Only Views

Updatable View:-

]If the database system can determine the reverse mapping from the view schema to the schema of the underlying base tables, then the view is updatable.

INSERT, UPDATE and DELETE operations can be performed on updatable views

Read-only view:-

Read-only views do not support such operations because the DBMS cannot map the changes to the underlying base tables.

Updation on view:-

A view with single defining table is updatable if the view attribute contains primary key of base relation as well as attributes with not null constraints that do not have default value.

A view defined on more than one table is not generally updatable.

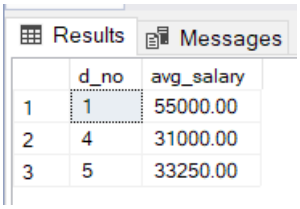
If it is insertion delete operation then it is not possible on more than one table.

Update operation is possible on the view defined on more than one table provided that it does not violates integrity constraints defined on the table.

If view contains aggregate functions then also view is not updatable.

// CODE

```
-- (1) Write a query to create a view to display average salary of every department
CREATE VIEW average(d_no, avg_salary) AS
SELECT d_no, AVG(salary)
FROM Employee
GROUP BY d_no;
SELECT * FROM average;
```



Results | Messages

	d_no	avg_salary
1	1	55000.00
2	4	31000.00
3	5	33250.00

```
-- (2a) Write a query to retrieve employee details of employees those are earning salary more than the average salary of department no. 5 without using view
SELECT *
FROM Employee
WHERE salary > (SELECT AVG(salary)
                FROM Employee
                Where d_no = 5);
```

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
3	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
4	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

.

-- (2b) Write a query to retrieve employee details of employees those are earning salary more than the average salary of department no. 5 using view

```
CREATE VIEW salary_greater_avg_5 AS
SELECT *
FROM Employee
WHERE salary > (SELECT AVG(salary)
                FROM Employee
                Where d_no = 5);
SELECT * FROM salary_greater_avg_5;
```

Results Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
3	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
4	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

-- (3) Write a query to create a view of employee info which shows the ssn and department no. of every employee

```
CREATE VIEW ssn_and_dno AS
SELECT ssn, d_no
FROM Employee
SELECT * FROM ssn_and_dno;
```

Results Messages

	ssn	d_no
1	123456789	5
2	333445555	5
3	453453453	5
4	666884444	5
5	888665555	1
6	987654321	4
7	987987987	4
8	999887777	4

-- (4) Write a query to create a view of department info which shows dept name, dept id, average salary and no. of employees

```
CREATE VIEW department_info(d_name, d_no, avg_salary, no_employee) AS
SELECT d_name, d.d_no, AVG(salary), COUNT(*)
FROM Employee e JOIN Department d ON e.d_no = d.d_no
GROUP BY d.d_name, d.d_no;
SELECT * FROM department_info;
```

Results | Messages

	d_name	d_no	avg_salary	no_employee
1	Headquarters	1	55000.00	1
2	Administration	4	31000.00	3
3	Research	5	33250.00	4

-- (5a) Write a query to retrieve employee details of employee those are earning salary more than the average salary of Research dept without using view

```
SELECT *
FROM Employee
WHERE salary > (SELECT AVG(salary)
                FROM Employee
                WHERE d_no = (SELECT d_no
                              FROM Department
                              WHERE d_name = 'Research'
                              )
                );
```

Results | Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
3	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
4	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

-- (5b) Write a query to retrieve employee details of employee those are earning salary more than the average salary of Research dept using view

```
CREATE VIEW salary_greater_avg_research AS
SELECT *
FROM Employee
WHERE salary > (SELECT AVG(salary)
                FROM Employee
                WHERE d_no = (SELECT d_no
                              FROM Department
                              WHERE d_name = 'Re          search'
                              )
                );
SELECT * FROM salary_greater_avg_research;
```

Results | Messages

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
2	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
3	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
4	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4

Conclusion : Hence we have successfully studied and implemented Views in DBMS.

DBMS LAB
Lab Assignment number 08

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim: Experiment to study and implement Join in SQL.

Theory:

Join:-

Join operation is used to retrive data from two or more than two tables based on some logical relationship between tables.
Join condition is specified on common column between two tables.
Join condition can be specified either from clause or where clause
There are two methods of specifying join condition
By using join key word
Without using join keyword.

Without using Join keyword:-
General syntax:-
Select <list of columns> from table1 as t1,table2 as t2 where t1.colname=t2.colname
Eg. Retrive empname and dname of dept for which emp is
working.
Select ename,dname from emp as e,dept as d where e.dno=d.dno

ename	dname
John	comp
smit	comp
Neha	it
smita	comp

Select ename,dno,dname from emp as e,dept as d where e.dno=d.dno -- Error (Ambiguous column name dno)

Select ename,e.dno,dname from emp as e,dept as d where e.dno=d.dno

ename	dno	dname
John	101	comp
smit	101	comp
Neha	102	it
smita	101	comp

By using join key word:-
Select e.ename,e.dno,dname from(emp as e join dept as d on e.dno=d.dno)

Types of Joins:-

1.Inner Join -

Inner Join keyword returns rows when there is at least one match in both table.
An inner join requires each record in the two joined tables to have a matching record.
An inner join essentially combines the records from two tables (A and B) based on a given join-predicate.
The result of the join can be defined as the outcome of first taking the Cartesian product (or cross-join) of all records in the tables then return all records which satisfy the join predicate

Select e.ename,e.dno,dname from(emp as e inner join dept as d on e.dno=d.dno)
Two types:-1) Self join
2) Equijoin

A. Self-join:-

A self-join is joining a table to itself
Eg. Display empname ,supervisor name of supervisor
who supervises that employee.
Select e.ename as empname , s.ename as supervisorname from emp as e,emp as s where e.superssn=s.ssn

ssn	ename	salary	dno	superssn
1	John	45000.0000	101	10
2	smita	42000.0000	101	10
10	Neha	25000.0000	102	NULL
5	smit	35000.0000	101	15
15	Nisha	40000.0000	103	10

Employee Table

empname	supervisorname
John	Neha
smita	Neha
smit	Nisha
Nisha	Neha

Output

select *from emp as e,emp as s

ssn	ename	salary	dno	superssn	ssn	ename	salary	dno	superssn
1	John	45000.0000	101	10	1	John	45000.0000	101	10
2	smita	42000.0000	101	10	1	John	45000.0000	101	10
10	Neha	25000.0000	102	NULL	1	John	45000.0000	101	10
5	smit	35000.0000	101	15	1	John	45000.0000	101	10
15	Nisha	40000.0000	103	10	1	John	45000.0000	101	10
1	John	45000.0000	101	10	2	smita	42000.0000	101	10
2	smita	42000.0000	101	10	2	smita	42000.0000	101	10
10	Neha	25000.0000	102	NULL	2	smita	42000.0000	101	10
5	smit	35000.0000	101	15	2	smita	42000.0000	101	10
15	Nisha	40000.0000	103	10	2	smita	42000.0000	101	10
1	John	45000.0000	101	10	10	Neha	25000.0000	102	NULL
2	smita	42000.0000	101	10	10	Neha	25000.0000	102	NULL

Eg. Display empname ,supervisor name of supervisor
who supervises that employee where supervisor salary is greater than salary of supervisor.

Select e.ename , s.ename from emp as e,emp as s where e.superssn=s.ssn and e.salary>s.salary

ename	ename
John	Neha
smita	Neha
Nisha	Neha

B. Equi join:-

Only rows satisfying selection criteria from both joined tables are selected. Selection criteria is based on equality condition on common column.

join condition uses only equality predicate "="

Eg. Display ename and dname of department for which employee is working.

select ename,dname from emp as e,dept as d where e.dno=d.dno

2. Outer Join:- Result of innerjoin + non matching records

LEFT OUTER JOIN - Rows satisfying selection criteria from both joined tables are selected as well as all remaining rows from left joined table are being kept along with null those are not having matching records in right joined table .

Select e.ename,e.dno,dname from(emp as e left outer join dept as d on e.dno=d.dno)

ename	dno	dname
John	101	comp
smita	101	comp
Neha	102	it
smit	101	comp
Nisha	103	NULL

RIGHT OUTER JOIN - Rows satisfying selection criteria from both joined tables are selected as well as all remaining rows from right joined table are being kept along with Nulls those are not having matching records in left joined table values.

Select e.ename,e.dno,dname from(emp as e right outer join dept as d on e.dno=d.dno)

ename	dno	dname
John	101	comp
smita	101	comp
smit	101	comp
Neha	102	it
NULL	NULL	extc

FULL OUTER JOIN - rows satisfying selection criteria from both joined tables are selected as well as all remaining rows both from left joined table and right joined table are those are not having matching records.

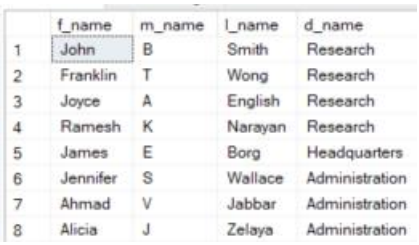
Select e.ename,e.dno,dname from(emp as e full outer join dept as d on e.dno=d.dno)

ename	dno	dname
John	101	comp
smita	101	comp
Neha	102	it
smit	101	comp
Nisha	103	NULL
NULL	NULL	extc

Output:
Display ename and dname of the department for which the employee is working using Joins.

1. Perform Left Outer Join

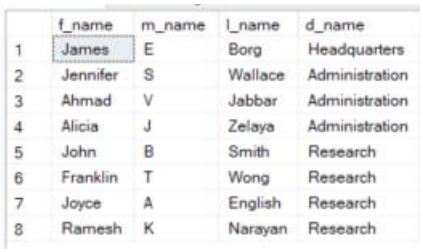
   select Fname, Mname, Lname, Dname from (Employee as e left outer join Department as d on e.Dno=d.Dnumber);



	f_name	m_name	l_name	d_name
1	John	B	Smith	Research
2	Franklin	T	Wong	Research
3	Joyce	A	English	Research
4	Ramesh	K	Narayan	Research
5	James	E	Borg	Headquarters
6	Jennifer	S	Wallace	Administration
7	Ahmad	V	Jabbar	Administration
8	Alicia	J	Zelaya	Administration

2. Perform Right Outer Join

   select Fname, Mname, Lname, Dname from (Employee as e right outer join Department as d on e.Dno=d.Dnumber);



	f_name	m_name	l_name	d_name
1	James	E	Borg	Headquarters
2	Jennifer	S	Wallace	Administration
3	Ahmad	V	Jabbar	Administration
4	Alicia	J	Zelaya	Administration
5	John	B	Smith	Research
6	Franklin	T	Wong	Research
7	Joyce	A	English	Research
8	Ramesh	K	Narayan	Research

3. Perform Full Outer Join

   select Fname, Mname, Lname, Dname from (Employee as e full outer join Department as d on e.Dno=d.Dnumber);

	f_name	m_name	l_name	d_name
1	John	B	Smith	Research
2	Franklin	T	Wong	Research
3	Joyce	A	English	Research
4	Ramesh	K	Narayan	Research
5	James	E	Borg	Headquarters
6	Jennifer	S	Wallace	Administration
7	Ahmad	V	Jabbar	Administration
8	Alicia	J	Zelaya	Administration

Conclusion: Hence we successfully studied and implemented all types of Joins in DBMS.

DBMS LAB
Lab Assignment number 09

Name: Aamir Ansari **Batch:** A **Roll no.** 01

Aim:- Experiment to study Stored Procedures and functions.

Theory:-
PL/SQL:- (Procedural SQL)
PL/SQL is extension of SQL.
SQL with Programming Capability.

Variable:- Is an object that can hold data value.
Local variable:-
Local variable name should begin with @ sign in SQL Server.
Local variables are used for passing values to SQL statement.
Local variables are used to store value temporary
eg. @empid

System Global variable:-
1. Global variable names begins with @@ sign.
2. eg. @@version

Declaring variable:-
General syntax:
Declare vname as datatype
Or
Declare vname datatype
Eg. Declare @eid as integer
Every variable should be defined for its datatype separately.
Begin……..end: -
Block of code is identified by using begin….end keyword.
Scope of the variable is identified by using begin and end keyword.
Eg. Begin
Declare @a as float
End
Assigning Value to the variable:-
Set or select statement is used to assign value to the variable.
Set keyword:-
Set @vname=10
Select @vname=columnname

Eg.

```
Begin
Declare @s as integer,@name as varchar(20)
Set @s=101
Select @name=ename from emp where ssn=@s
   End
```

```
select @@version          --Global Variable
--output
Microsoft SQL Server  2000
```

Control structures:-

-- If Control structure

```
begin
declare @a int,@b int,@c int General Syntax:-
set @a=10 if (condition)
set @b=20 set @c=15  begin
if(@a>@b)and(@a>@c) stmts
   print @a end
else if(@b>@c) else
     print @b begin
else print @c stmts
end end
--output 20
```

-- While loop

```
 begin
 declare @i int   General syntax:-
 set @i=1 while(cond)
 while(@i<10) begin
 begin stmts
    print 'i='+cast(@i as varchar(20))            end
--print @i
set @i=@i+1
 end
end
--output
i=1
i=2
i=3
i=4
```

i=5
i=6
i=7
i=8
i=9

-- Case Control Structures

```
begin
declare @t as varchar(20)
declare @s varchar(20) General Syntax:-
set @t=’o’ case
set @s=case when condn then exp
      when @t='o' then 'one' when condn then exp
       when @t=’t’ then ’two’ else exp
      else ‘greater than two’ end
end
end
```

Stored Procedures:-

Stored procedures are precompiled database queries that improves the security, efficiency and usability of code.

Stored procedures are extremely similar to the constructs seen in other programming languages.

They accept data in the form of input parameters that are specified at execution time. These input parameters (if implemented) are utilized in the execution of a series of statements that produce some result.

This result is returned to the calling environment through the use of a recordset, output parameters and a return code.

Stored procedures can have upto 1024 parameter.

General syntax:-

```
Create procedure procname (@v as datatype in|out)
As
Begin
   Stmts
 End
```

Calling Procedure:-

Execute procname value

```
create procedure getmonth
as
begin
select month(getdate())
end
```

```
execute getmonth
```

--Procedure without output parameter

```
-- create procedure which displays salary of emp when we pass  ssn as parameter to the procedure
create procedure listsal(@s int)
as
begin
   declare @sal int
   select @sal=emp_sal from empl1 where emp_id=@s
   print @sal
end
```

--Executing Procedure without parameter

```
execute listsal 1
```

--Procedure with output parameter

```
create procedure listsal1(@e int,@sal int output)
as
begin
      select @sal=emp_sal from empl1 where emp_id=@e
end
--Execution
begin
declare @s int
execute listsal1 2,@s output
print @s
end
```

```
--Nesting of procedure
create procedure grosssal(@e int,@d int,@hra int)
as
begin
  declare @g int,@s int
  execute listsal1 @e, @s output
   set @g=@s+(@s*(@d/100))+(@s*(@hra/100))
print 'gross sal'+ cast(@g as varchar(20))
end
```

```
execute grosssal 1,50,20
```

Advantages of Stored procedure:-

Precompiled execution. SQL Server compiles each stored procedure once and then reutilizes the execution plan. Execution speed increases.

Reduced client/server traffic.

Efficient reuse of code and programming abstraction. Stored procedures can be used by multiple users and client programs.

Enhanced security controls. You can grant users permission to execute a stored procedure independently of underlying table permissions.

Function:-

Function is precompiled set of statements which returns value explicitly to the caller of function.

```
Create funcname fname (@v as datatype) returns datatype
As
Begin
   Stmts
Return value
 End
```

Execution :-

```
Print username.functionname(value)
create function avgsal1(@d int)
returns int as
begin
  declare @avgsal int
  select @avgsal=avg(emp_sal) from empl1 where do=@d
  return @avgsal
end
```

--Execution

```
print jayshree.avgsal1(1)
select *from empl1 where emp_sal>jayshree.avgsal1(1)
```

Procedures

1. Printing numbers from 1 to 10

```
begin
        declare @a as int;
        set @a=1;
        while(@a<10)
        begin
                print 'a='+cast(@a as varchar(20))
                set @a=@a+1;
        end
end
```

```
a=1
a=2
a=3
a=4
a=5
a=6
a=7
a=8
a=9
```

2. Print the greatest of 3 numbers

```
begin
        declare @a as int,@b as int,@c as int;
        set @a=15;
        Set @b=20
        set @c=10;
        print 'The largest number is'
        if(@a>@b)and(@a>@c)
                print @a
        else if(@b>@c)
                print @b
        else
                print @c
end
```

```
The largest number is
20
```

3. Printing the table of 5

```
begin
```

```
declare @i as int;
set @i=1
declare @m as int;
set @m=5
while(@i<=10)
begin
    print cast(@m as varchar(20))+'x'+cast(@i as varchar(20))+'='+cast(@m*@i as
    varchar(20))
    set @i=@i+1;
end
end
```

```
5x1=5
5x2=10
5x3=15
5x4=20
5x5=25
5x6=30
5x7=35
5x8=40
5x9=45
5x10=50
```

Stored Procedures:

1. Without Parameters: Create a procedure which displays details of all employees

```
CREATE PROCEDURE EmployeeSelect
AS
SELECT * FROM Employee
GO;
EXEC EmployeeSelect;
```

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	15000.00	333445555	5
2	Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000.00	888665555	5
3	Joyce	A	English	453453453	1972-07-31	5361 Rice, Houston, TX	F	25000.00	333445555	5
4	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5
5	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1
6	Jennifer	S	Wallace	987654321	1941-06-20	291 Berrym, Bellaire, TX	F	43000.00	888665555	4
7	Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000.00	987654321	4
8	Alicia	J	Zelaya	999887777	1968-01-19	3321 Castle, Spring, TX	F	25000.00	987654321	4

2. With Parameters

a. A procedure which displays details of Employee with name 'James'

```
CREATE PROCEDURE EmpSelect @Fname varchar(30)
AS
SELECT * FROM Employee WHERE Fname=@Fname
```

```
GO
EXEC EmpSelect @Fname='James';
```

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000.00	NULL	1

b. A procedure which displays details of Employee with name 'Ramesh' and salary=38000

```
CREATE PROCEDURE selectCondition @Fname varchar(30), @Salary money
AS
SELECT * FROM Employee WHERE Fname=@Fname AND Salary=@Salary
GO
EXEC selectCondition @Fname='Ramesh',@Salary=38000;
```

	f_name	m_name	l_name	ssn	dob	addr	sex	salary	super_ssn	d_no
1	Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000.00	333445555	5

Conclusion: Hence we have successfully studied and implemented stored Procedures in DBMS