INDIAN INSTITUTE OF INFORMATION TECHNOLOGY

Submitted by : ***Harshit Kumar Vishwakarma***

Branch: **CSE**

Group: **A**

Semester: **Vth**

Course: **DBMS**

Course Code: **CS 3001**

Submitted to : ***Dr. Nidhi Kushwaha***

LAB ASSIGNMENT 1:

Table – Employee

EMLPOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
001	Monika	Arora	100000	2014-02-20 09:00:00	HR
002	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
003	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
004	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
005	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
006	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
007	Satish	Kumar	75000	2014-01-20 09:00:00	Account
008	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

Creating table Employee:

```
CREATE TABLE Employee (
   EMPLOYEE_ID NUMBER,
   FIRST_NAME CHAR(50),
   LAST_NAME CHAR(50),
   SALARY NUMBER,
   JOINING_DATE CHAR(50),
   DEPARTMENT CHAR(50)
);
```

```
insert into Employee values (001, 'Monika', 'Arora', 100000, '2014-02-20 09:00:00','HR');
```

```
insert into Employee values (002, 'Niharika', 'Verma', 80000, '2014-06-11 09:00:00','Admin');
```

```
insert into Employee values (003, 'Vishal', 'Singhal', 300000, '2014-02-20 09:00:00', 'HR');
```

```
insert into Employee values (004, 'Amitabh', 'Singh', 500000, '2014-02-20 09:00:00', 'Admin');
```

```
insert into Employee values (005, 'Vivek', 'Bhati', 500000, '2014-06-11 09:00:00', 'Admin');
```

```
insert into Employee values (006, 'Vipul', 'Diwan', 200000, '2014-06-11 09:00:00', 'Account');
```

insert into Employee values (007, 'Satish', 'Kumar', 75000, '2014-01-20 09:00:00', 'Account');

insert into Employee values (008, 'Geetika', 'Chauhan', 90000, '2014-04-11 09:00:00', 'Admin');

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

8 rows returned in 0.00 seconds CSV Export

Table – Profit

EMPLOYEE_REF_ID	PROFIT_DATE	PROFIT_AMOUNT
1	2016-02-20 00:00:00	5000
2	2016-06-11 00:00:00	3000
3	2016-02-20 00:00:00	4000
1	2016-02-20 00:00:00	4500
2	2016-06-11 00:00:00	3500

Creating table Profit:

```
CREATE TABLE Profit (
   EMPLOYEE_REF_ID NUMBER,
   PROFIT_DATE DATE,
   PROFIT_AMOUNT NUMBER
);
```

INSERT INTO Profit VALUES (1, TO_DATE('2016-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 5000);

INSERT INTO Profit VALUES (2, TO_DATE('2016-06-11 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 3000);

INSERT INTO Profit VALUES (3, TO_DATE('2016-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 4000);

INSERT INTO Profit VALUES (1, TO_DATE('2016-02-20 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 4500);

INSERT INTO Profit VALUES (2, TO_DATE('2016-06-11 00:00:00', 'YYYY-MM-DD HH24:MI:SS'), 3500);

Results Explain Describe Saved SQL History

EMPLOYEE_REF_ID	PROFIT_DATE	PROFIT_AMOUNT
1	2016-02-20 00:00:00	5000
2	2016-06-11 00:00:00	3000
3	2016-02-20 00:00:00	4000
1	2016-02-20 00:00:00	5000
2	2016-06-11 00:00:00	3500

5 rows returned in 0.01 seconds CSV Export

Table – Category

EMPLOYEE_REF_ID	EMPLOYEE_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00

EMPLOYEE_REF_ID	EMPLOYEE_TITLE	AFFECTED_FROM
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

Creating table Category:

create table Category(

Employee_Ref_Id number(3),

Employee_Title char(20),

Affected_From char(30));

insert into Category values(1, 'Manager', '2016-02-20 00:00:00');

```
insert into Category values(2, 'Executive', '2016-06-11 00:00:00');
insert into Category values(8, 'Executive', '2016-06-11 00:00:00');
insert into Category values(5, 'Manager', '2016-06-11 00:00:00');
insert into Category values(4, 'Asst. Manager', '2016-06-11 00:00:00');
insert into Category values(7, 'Executive', '2016-06-11 00:00:00');
insert into Category values(6, 'Lead', '2016-06-11 00:00:00');
insert into Category values(3, 'Lead', '2016-06-11 00:00:00');
```

Results Explain Describe Saved SQL History

EMPLOYEE_REF_ID	EMPLOYEE_TITLE	AFFECTED_FROM
1	Manager	2016-02-20 00:00:00
2	Executive	2016-06-11 00:00:00
8	Executive	2016-06-11 00:00:00
5	Manager	2016-06-11 00:00:00
4	Asst. Manager	2016-06-11 00:00:00
7	Executive	2016-06-11 00:00:00
6	Lead	2016-06-11 00:00:00
3	Lead	2016-06-11 00:00:00

8 rows returned in 0.00 seconds CSV Export

For the tables created above, write the SQL queries for the following:

1. **Write an SQL query to determine the nth (say n=5) highest salary from a table.**

```
SELECT DISTINCT SALARY
FROM Employee E1
WHERE 5 = (
   SELECT COUNT(DISTINCT SALARY)
   FROM Employee E2
   WHERE E1.SALARY <= E2.SALARY
);
```

Results Explain Describe Saved SQL History

SALARY
90000

1 rows returned in 0.00 seconds CSV Export

2. **Write an SQL query to determine the 5th highest salary without using the TOP or limit method.**

```
SELECT DISTINCT SALARY
```

```
FROM Employee E1
WHERE 5 = (
   SELECT COUNT(DISTINCT SALARY)
   FROM Employee E2
   WHERE E1.SALARY <= E2.SALARY
);
```

Results Explain Describe Saved SQL History

SALARY
90000

1 rows returned in 0.00 seconds CSV Export

3. **Write an SQL query to fetch the list of employees with the same salary.**

```
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY IN (SELECT SALARY FROM Employee
   GROUP BY SALARY HAVING COUNT(*) > 1);
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY
4	Amitabh	Singh	500000
5	Vivek	Bhati	500000

2 rows returned in 0.01 seconds CSV Export

4. **Write an SQL query to show the second-highest salary from a table.**

```
SELECT MAX(SALARY) AS SecondHighestSalary
FROM Employee
WHERE SALARY < (
   SELECT MAX(SALARY)
   FROM Employee);
```

Results Explain Describe Saved SQL History

SECONDHIGHESTSALARY
300000

1 rows returned in 0.00 seconds CSV Export

5. **Write an SQL query to show one row twice in the results from a table.**

```
with sort_tab as(
SELECT * FROM Employee
UNION ALL SELECT *
FROM Employee)
SELECT * FROM sort_tab
WHERE First_Name = 'Vishal'
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR

2 rows returned in 0.02 seconds CSV Export

6. **Write an SQL query to fetch intersecting records of two tables.**

```
SELECT Employee_ID,
First_Name,
Last_name
Department FROM Employee
intersect select Employee_ID,
First_Name, Last_name
FROM Employee;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	DEPARTMENT
1	Monika	Arora
2	Niharika	Verma
3	Vishal	Singhal
4	Amitabh	Singh
5	Vivek	Bhati
6	Vipul	Diwan
7	Satish	Kumar
8	Geetika	Chauhan

8 rows returned in 0.00 seconds CSV Export

7. **Write an SQL query to fetch the first 50% of records from a table.**

```
SELECT *
FROM Employee
where Employee_ID <= (
SELECT COUNT(Employee_ID)/2
FROM Employee);
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin

4 rows returned in 0.00 seconds CSV Export

8. **Write an SQL query to fetch the departments that have less than five people in them.**

```
SELECT DEPARTMENT
FROM Employee
GROUP BY DEPARTMENT
HAVING COUNT(*) < 5;
```

Results | Explain | Describe | Saved SQL | History

DEPARTMENT
Account
HR
Admin

3 rows returned in 0.02 seconds CSV Export

9. **Write an SQL query to show the last record from a table.**

```
SELECT * FROM (
SELECT * FROM
 Employee ORDER BY
Employee_ID DESC)
fetch WHERE ROWNUM=1;
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

1 rows returned in 0.00 seconds CSV Export

10. **Write an SQL query to fetch the first row of a table.**

```
SELECT * FROM
Employee fetch
WHERE ROWNUM=1;
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR

1 rows returned in 0.00 seconds CSV Export

11. **Write an SQL query to fetch the last five records from a table.**

```
SELECT * FROM (
SELECT * FROM Employee
order by Employee_ID desc)
fetch WHERE ROWNUM<=5;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin

5 rows returned in 0.02 seconds CSV Export

12. Write an SQL query to print the name of employees having the highest salary in each department.

```
SELECT E.First_Name,
E.Department FROM Employee
E WHERE E.Salary in (
SELECT max(Salary)
FROM Employee GROUP
BY Department);
```

Results Explain Describe Saved SQL History

FIRST_NAME	DEPARTMENT
Vishal	HR
Amitabh	Admin
Vivek	Admin
Vipul	Account

4 rows returned in 0.02 seconds CSV Export

13. Write an SQL query to fetch three max salaries from a table.

```
SELECT * FROM (
SELECT * FROM Employee
ORDER BY Salary DESC)
fetch WHERE ROWNUM<=3;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR

3 rows returned in 0.00 seconds CSV Export

14. Write an SQL query to fetch three min salaries from a table.

```
SELECT * FROM (
SELECT * FROM Employee
ORDER BY Salary DESC)
```

fetch WHERE ROWNUM<=3;

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

3 rows returned in 0.00 seconds CSV Export

15. Write an SQL query to fetch nth max salaries from a table.

SELECT * FROM (
SELECT * FROM Employee
ORDER BY Salary DESC)
fetch WHERE ROWNUM<=n;

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR

3 rows returned in 0.00 seconds CSV Export

16. Write an SQL query to fetch departments along with the total salaries paid for each of them.

SELECT DEPARTMENT, SUM(SALARY) AS TotalSalary
FROM Employee
GROUP BY DEPARTMENT;

Results Explain Describe Saved SQL History

DEPARTMENT	TOTALSALARY
Account	275000
HR	400000
Admin	1170000

3 rows returned in 0.02 seconds CSV Export

17. Write an SQL query to fetch the names of employees who earn the highest salary.

SELECT FIRST_NAME, LAST_NAME
FROM Employee
WHERE SALARY = (SELECT MAX(SALARY) FROM Employee);

Results Explain Describe Saved SQL History

FIRST_NAME	LAST_NAME
Amitabh	Singh
Vivek	Bhati

2 rows returned in 0.00 seconds CSV Export

18. Write an SQL query to show the top n (say 10) records of a table.

```
select * from
employee where
ROWNUM<=10;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

8 rows returned in 0.00 seconds CSV Export

19. Write an SQL query to clone a new table from another table.

```
CREATE TABLE Employee_Clone AS
SELECT *
FROM Employee;
```

Results Explain Describe Saved SQL History

```
Table created.
```

0.06 seconds

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

8 rows returned in 0.00 seconds CSV Export

20. Write an SQL query to show only even rows from a table.

```
SELECT *
FROM Employee
WHERE MOD(EMPLOYEE_ID, 2) = 0;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

4 rows returned in 0.00 seconds CSV Export

21. Write an SQL query to show only odd rows from a table.

```
SELECT *
FROM Employee
WHERE MOD(EMPLOYEE_ID, 2) <> 0;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account

4 rows returned in 0.00 seconds CSV Export

22. Write an SQL query to fetch “FIRST_NAME” from the EMPLOYEE table using the alias name.

```
SELECT FIRST_NAME AS EMPLOYEE_NAME FROM Employee;
```

Results Explain Describe Saved SQL History

EMPLOYEE_NAME
Monika
Niharika
Vishal
Amitabh
Vivek
Vipul
Satish
Geetika

8 rows returned in 0.00 seconds CSV Export

23. Write an SQL query to fetch "FIRST_NAME" from the Employee table in upper case.

SELECT UPPER(FIRST_NAME) AS UPPER_CASE_FIRST_NAME
FROM Employee;

Results Explain Describe Saved SQL History

UPPER_CASE_FIRST_NAME
MONIKA
NIHARIKA
VISHAL
AMITABH
VIVEK
VIPUL
SATISH
GEETIKA

8 rows returned in 0.00 seconds CSV Export

24. Write an SQL query to fetch unique values of DEPARTMENT from the Employee table.

SELECT DISTINCT DEPARTMENT
FROM Employee;

Results Explain Describe Saved SQL History

DEPARTMENT
Account
HR
Admin

3 rows returned in 0.00 seconds CSV Export

25. Write an SQL query to print the first three characters of FIRST_NAME from the Employee table.

```
SELECT SUBSTR(FIRST_NAME, 1, 3) AS FIRST_THREE_CHARACTERS
FROM Employee;
```

Results Explain Describe Saved SQL History

FIRST_THREE_CHARACTERS
Mon
Nih
Vis
Ami
Viv
Vip
Sat
Gee

8 rows returned in 0.00 seconds CSV Export

26. Write an SQL query to find the position of the alphabet (‘a’) in the first name column ‘Amitabh’ from the Employee table.

```
SELECT INSTR(FIRST_NAME, 'a') AS
POSITION_OF_A FROM Employee
WHERE FIRST_NAME = 'Amitabh';
```

Results Explain Describe Saved SQL History

POSITION_OF_A
5

1 rows returned in 0.00 seconds CSV Export

27. Write an SQL query that fetches the unique values of DEPARTMENT from the Employee table and prints its length.

```
SELECT DEPARTMENT, LENGTHB(
TRIM(DEPARTMENT)) AS DEPARTMENT_LENGTH
FROM Employee
```

Results Explain Describe Saved SQL History

DEPARTMENT	DEPARTMENT_LENGTH
HR	2
Admin	5
HR	2
Admin	5
Admin	5
Account	7
Account	7
Admin	5

8 rows returned in 0.02 seconds CSV Export

28. **Write an SQL query to print the FIRST_NAME from the Employee table after replacing ‘a’ with ‘A’.**

```
SELECT REPLACE(FIRST_NAME, 'a', 'A') AS UPDATED_FIRST_NAME
FROM Employee;
```

Results | Explain | Describe | Saved SQL | History

UPDATED_FIRST_NAME
MonikA
NihArikA
VishAl
AmitAbh
Vivek
Vipul
SAtish
GeetikA

8 rows returned in 0.00 seconds CSV Export

29. **Write an SQL query to print the FIRST_NAME and LAST_NAME from the Employee table into a single column COMPLETE_NAME. A space char should separate them.**

```
SELECT FIRST_NAME || ' ' || LAST_NAME
AS COMPLETE_NAME
FROM Employee;
```

Results | Explain | Describe | Saved SQL | History

COMPLETE_NAME
Monika Arora
Niharika Verma
Vishal Singhal
Amitabh Singh
Vivek Bhati
Vipul Diwan
Satish Kumar
Geetika Chauhan

8 rows returned in 0.02 seconds CSV Export

30. **Write an SQL query to print all Employee details from the Employee table order by FIRST_NAME Ascending and DEPARTMENT Descending.**

```
SELECT *
FROM Employee
ORDER BY FIRST_NAME ASC, DEPARTMENT DESC;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin

8 rows returned in 0.00 seconds CSV Export

31. Write an SQL query to print details of employee excluding first names, "Vipul" and "Satish" from the Employee table.

```
SELECT *
FROM Employee
WHERE FIRST_NAME NOT IN ('Vipul', 'Satish');
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

6 rows returned in 0.00 seconds CSV Export

32. Write an SQL query to print details of Employee with DEPARTMENT name as "Admin".

```
SELECT *
FROM Employee
WHERE DEPARTMENT = 'Admin';
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
2	Niharika	Verma	80000	2014-06-11 09:00:00	Admin
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin

4 rows returned in 0.00 seconds CSV Export

33. **Write an SQL query to print details of the Employees whose FIRST_NAME contains ‘a’.**

```
SELECT *
FROM Employee
WHERE FIRST_NAME LIKE '%a%’;
```

Results Explain Describe Saved SQL History

FIRST_NAME
Monika
Niharika
Vishal
Amitabh
Satish
Geetika

6 rows returned in 0.00 seconds CSV Export

34. **Write an SQL query to print details of the Workers Employees whose FIRST_NAME ends with ‘h’ and contains six alphabets.**

```
SELECT *
FROM Employee
WHERE FIRST_NAME LIKE '_____h%';
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin

1 rows returned in 0.00 seconds CSV Export

35. **Write an SQL query to print details of the Employees whose SALARY lies between 100000 and 500000.**

```
SELECT *
FROM Employee
WHERE SALARY BETWEEN 100000 AND 500000;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account

5 rows returned in 0.00 seconds CSV Export

36. **Write an SQL query to print details of the Employees who joined in Feb’2014.**

```
SELECT *
```

```
FROM Employee
WHERE JOINING_DATE BETWEEN ('2014-02-01') AND ('2014-02-28');
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
3	Vishal	Singhal	300000	2014-02-20 09:00:00	HR
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin

3 rows returned in 0.02 seconds CSV Export

37. Write an SQL query to fetch worker names with salaries >= 50000 and <= 100000.

```
SELECT FIRST_NAME, LAST_NAME, SALARY
FROM Employee
WHERE SALARY BETWEEN 50000 AND 100000;
```

Results | Explain | Describe | Saved SQL | History

FIRST_NAME	LAST_NAME	SALARY
Monika	Arora	100000
Niharika	Verma	80000
Satish	Kumar	75000
Geetika	Chauhan	90000

4 rows returned in 0.00 seconds CSV Export

38. Write an SQL query to print details of the Employees who are also Managers.

```
SELECT e.*
FROM Employee e
JOIN Category c ON e.EMPLOYEE_ID = c.EMPLOYEE_REF_ID
WHERE c.EMPLOYEE_TITLE = 'Manager';
```

Results | Explain | Describe | Saved SQL | History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
1	Monika	Arora	100000	2014-02-20 09:00:00	HR
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin

2 rows returned in 0.02 seconds CSV Export

39. Write an SQL query to show records from one table that another table does not have.

```
SELECT e.*
FROM Employee e
LEFT JOIN Profit p ON e.EMPLOYEE_ID = p.EMPLOYEE_REF_ID
WHERE p.EMPLOYEE_REF_ID IS NULL;
```

Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SALARY	JOINING_DATE	DEPARTMENT
5	Vivek	Bhati	500000	2014-06-11 09:00:00	Admin
8	Geetika	Chauhan	90000	2014-04-11 09:00:00	Admin
6	Vipul	Diwan	200000	2014-06-11 09:00:00	Account
7	Satish	Kumar	75000	2014-01-20 09:00:00	Account
4	Amitabh	Singh	500000	2014-02-20 09:00:00	Admin

5 rows returned in 0.00 seconds CSV Export

40. Write an SQL query to fetch the count of employees working in the department 'Admin'

```
SELECT COUNT(*)
FROM Employee
WHERE DEPARTMENT = 'Admin';
```

Results Explain Describe Saved SQL History

COUNT(*)
4

1 rows returned in 0.00 seconds CSV Export

DBMS LAB ASSIGNMENT 2

Sailors Table

SID	SNAME	AGE	RATING
22	Gagan	7	45
29	Magan	1	33
31	Kite	8	55
32	Andy	8	25.5
58	Rawan	10	35
64	Julia	7	35
71	Zorba	10	40
74	Recordo	9	40
85	Messi	3	25.5
95	Kohli	3	63.5

Creating Table Sailors:

```
CREATE TABLE Sailors (
   SID NUMBER,
   SNAME VARCHAR2(50),
   AGE NUMBER,
   RATING NUMBER
);
INSERT INTO Sailors VALUES (22, 'Gagan', 7, 45);
INSERT INTO Sailors VALUES (29, 'Magan', 1, 33);
INSERT INTO Sailors VALUES (31, 'Kite', 8, 55);
INSERT INTO Sailors VALUES (32, 'Andy', 8, 25.5);
INSERT INTO Sailors VALUES (58, 'Rawan', 10, 35);
INSERT INTO Sailors VALUES (64, 'Julia', 7, 35);
INSERT INTO Sailors VALUES (71, 'Zorba', 10, 40);
INSERT INTO Sailors VALUES (74, 'Recordo', 9, 40);
INSERT INTO Sailors VALUES (85, 'Messi', 3, 25.5);
INSERT INTO Sailors VALUES (95, 'Kohli', 3, 63.5);
```

Results | Explain | Describe | Saved SQL | History

SID	SNAME	AGE	RATING
22	Gagan	7	45
29	Magan	1	33
31	Kite	8	55
32	Andy	8	25.5
58	Rawan	10	35
64	Julia	7	35
71	Zorba	10	40
74	Recordo	9	40
85	Messi	3	25.5
95	Kohli	3	63.5

10 rows returned in 0.00 seconds CSV Export

Reserves Table

SID	BID	DAY
22	101	10-OCT-22
29	102	10-OCT-22
31	103	10-AUG-22
32	104	10-JUL-23
58	102	11-OCT-22
64	103	11-JUN-23
71	104	11-DEC-22
74	101	09-MAY-23
85	102	09-AUG-23
95	104	09-AUG-23

Creating Table Reserves:

```
CREATE TABLE Reserves (
   SID NUMBER,
   BID NUMBER,
   DAY DATE
);
INSERT INTO Reserves VALUES (22, 101, TO_DATE('10-OCT-22', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (29, 102, TO_DATE('10-OCT-22', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (31, 103, TO_DATE('10-AUG-22', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (32, 104, TO_DATE('10-JUL-23', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (58, 102, TO_DATE('11-OCT-22', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (64, 103, TO_DATE('11-JUN-23', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (71, 104, TO_DATE('11-DEC-22', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (74, 101, TO_DATE('09-MAY-23', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (85, 102, TO_DATE('09-AUG-23', 'DD-MON-YY'));
INSERT INTO Reserves VALUES (95, 104, TO_DATE('09-AUG-23', 'DD-MON-YY'));
```

Results Explain Describe Saved SQL History

SID	BID	DAY
22	101	10-OCT-22
29	102	10-OCT-22
31	103	10-AUG-22
32	104	10-JUL-23
58	102	11-OCT-22
64	103	11-JUN-23
71	104	11-DEC-22
74	101	09-MAY-23
85	102	09-AUG-23
95	104	09-AUG-23

10 rows returned in 0.02 seconds CSV Export

Boat Table (BID, BName, Color)

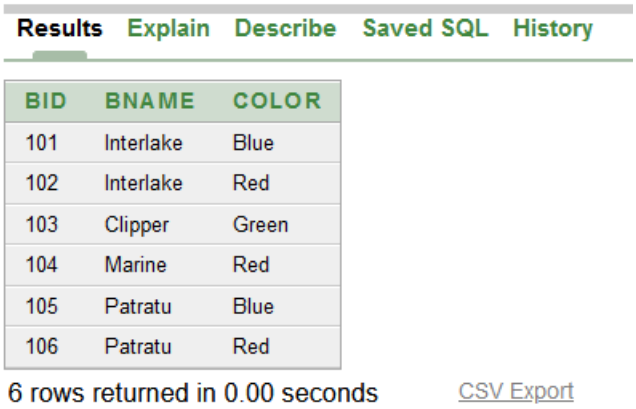
BID	BNAME	COLOR
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red
105	Patratu	Blue
106	Patratu	Red

Creating Table Boat:

```
CREATE TABLE Boat (
   BID NUMBER,
   BNAME VARCHAR2(50),
   COLOR VARCHAR2(50)
);
INSERT INTO Boat VALUES (101, 'Interlake', 'Blue');
INSERT INTO Boat VALUES (102, 'Interlake', 'Red');
INSERT INTO Boat VALUES (103, 'Clipper', 'Green');
INSERT INTO Boat VALUES (104, 'Marine', 'Red');
```

INSERT INTO Boat VALUES (105, 'Patratu', 'Blue');

INSERT INTO Boat VALUES (106, 'Patratu', 'Red');



Results Explain Describe Saved SQL History

BID	BNAME	COLOR
101	Interlake	Blue
102	Interlake	Red
103	Clipper	Green
104	Marine	Red
105	Patratu	Blue
106	Patratu	Red

6 rows returned in 0.00 seconds CSV Export

Perform the following queries on the above tables.

1. **Write an SQL syntax for joining SAILORS, RESERVES, and BOAT tables.**
   a. **First sailors and reserves then the result should be join with boat**

```
SELECT *
FROM Sailors
JOIN Reserves ON Sailors.SID = Reserves.SID
JOIN Boat ON Reserves.BID = Boat.BID;
```

Results Explain Describe Saved SQL History

SID	SNAME	AGE	RATING	SID	BID	DAY	BID	BNAME	COLOR
22	Gagan	7	45	22	101	10-OCT-22	101	Interlake	Blue
29	Magan	1	33	29	102	10-OCT-22	102	Interlake	Red
31	Kite	8	55	31	103	10-AUG-22	103	Clipper	Green
32	Andy	8	25.5	32	104	10-JUL-23	104	Marine	Red
58	Rawan	10	35	58	102	11-OCT-22	102	Interlake	Red
64	Julia	7	35	64	103	11-JUN-23	103	Clipper	Green
71	Zorba	10	40	71	104	11-DEC-22	104	Marine	Red
74	Recordo	9	40	74	101	09-MAY-23	101	Interlake	Blue
85	Messi	3	25.5	85	102	09-AUG-23	102	Interlake	Red
95	Kohli	3	63.5	95	104	09-AUG-23	104	Marine	Red

10 rows returned in 0.00 seconds CSV Export

   b. **First reserves and boat then the result should be join with sailors Use the following concept SELECT * FROM Table1 JOIN Table2 ON Table1.fielda=Table2.fielda JOIN Table3 ON Table2.fieldb = Table3.fieldb**

```
SELECT *
FROM Reserves
JOIN Boat ON Reserves.BID = Boat.BID
JOIN Sailors ON Sailors.SID = Reserves.SID;
```

Results Explain Describe Saved SQL History

SID	BID	DAY	BID	BNAME	COLOR	SID	SNAME	AGE	RATING
22	101	10-OCT-22	101	Interlake	Blue	22	Gagan	7	45
29	102	10-OCT-22	102	Interlake	Red	29	Magan	1	33
31	103	10-AUG-22	103	Clipper	Green	31	Kite	8	55
32	104	10-JUL-23	104	Marine	Red	32	Andy	8	25.5
58	102	11-OCT-22	102	Interlake	Red	58	Rawan	10	35
64	103	11-JUN-23	103	Clipper	Green	64	Julia	7	35
71	104	11-DEC-22	104	Marine	Red	71	Zorba	10	40
74	101	09-MAY-23	101	Interlake	Blue	74	Recordo	9	40
85	102	09-AUG-23	102	Interlake	Red	85	Messi	3	25.5
95	104	09-AUG-23	104	Marine	Red	95	Kohli	3	63.5

10 rows returned in 0.02 seconds CSV Export

2. **Select BNAME which is assigned to at least two BID.**

```
SELECT BNAME
FROM Boat
GROUP BY BNAME
HAVING COUNT(BID) >= 2;
```

Results | Explain | Describe | Saved SQL | History

BNAME
Patratu
Interlake

2 rows returned in 0.00 seconds CSV Export

3. **Apply SQL RIGHT JOIN two tables, the RESERVES table and the BOAT table.**

```
SELECT *
FROM Reserves
RIGHT JOIN Boat ON Reserves.BID = Boat.BID;
```

Results | Explain | Describe | Saved SQL | History

SID	BID	DAY	BID	BNAME	COLOR
22	101	10-OCT-22	101	Interlake	Blue
29	102	10-OCT-22	102	Interlake	Red
31	103	10-AUG-22	103	Clipper	Green
32	104	10-JUL-23	104	Marine	Red
58	102	11-OCT-22	102	Interlake	Red
64	103	11-JUN-23	103	Clipper	Green
71	104	11-DEC-22	104	Marine	Red
74	101	09-MAY-23	101	Interlake	Blue
85	102	09-AUG-23	102	Interlake	Red
95	104	09-AUG-23	104	Marine	Red
-	-	-	106	Patratu	Red
-	-	-	105	Patratu	Blue

12 rows returned in 0.02 seconds CSV Export

4. **Apply SQL LEFT JOIN two tables, the SAILORS and the RESERVES table.**

```
SELECT *
FROM Sailors
LEFT JOIN Reserves ON Sailors.SID = Reserves.SID;
```

Results Explain Describe Saved SQL History

SID	SNAME	AGE	RATING	SID	BID	DAY
22	Gagan	7	45	22	101	10-OCT-22
29	Magan	1	33	29	102	10-OCT-22
31	Kite	8	55	31	103	10-AUG-22
32	Andy	8	25.5	32	104	10-JUL-23
58	Rawan	10	35	58	102	11-OCT-22
64	Julia	7	35	64	103	11-JUN-23
71	Zorba	10	40	71	104	11-DEC-22
74	Recordo	9	40	74	101	09-MAY-23
85	Messi	3	25.5	85	102	09-AUG-23
95	Kohli	3	63.5	95	104	09-AUG-23

10 rows returned in 0.00 seconds CSV Export

5. **Get the complete record (SNAME, DAY) from both tables([SAILORS], [RESERVERS]), if no match is found in any table, then show NULL.**

```
SELECT SNAME, DAY
FROM Sailors
FULL OUTER JOIN Reserves ON Sailors.SID = Reserves.SID;
```

Results Explain Describe Saved SQL History

SNAME	DAY
Gagan	10-OCT-22
Magan	10-OCT-22
Kite	10-AUG-22
Andy	10-JUL-23
Rawan	11-OCT-22
Julia	11-JUN-23
Zorba	11-DEC-22
Recordo	09-MAY-23
Messi	09-AUG-23
Kohli	09-AUG-23

10 rows returned in 0.00 seconds CSV Export

6. **Write a query to find out the SNAME whose BID is 101, and display it.**

```
SELECT SNAME FROM Sailors
JOIN Reserves ON Sailors.SID = Reserves.SID
WHERE Reserves.BID = 101;
```

Results Explain Describe Saved SQL History

SNAME
Gagan
Recordo

2 rows returned in 0.00 seconds CSV Export

7. **Write down the query to fetch BID which is assigned to the Sailors whose age is less than 5 years.**

```
SELECT BID
FROM Sailors
JOIN Reserves ON Sailors.SID = Reserves.SID
WHERE Sailors.AGE < 5;
```

Results | Explain | Describe | Saved SQL | History

BID
102
102
104

3 rows returned in 0.02 seconds CSV Export

8. **Apply SQL FULL JOIN on two tables, the RESERVES table and the BOAT table.**

```
SELECT *
FROM Reserves
FULL OUTER JOIN Boat ON Reserves.BID = Boat.BID;
```

Results | Explain | Describe | Saved SQL | History

SID	BID	DAY	BID	BNAME	COLOR
74	101	09-MAY-23	101	Interlake	Blue
22	101	10-OCT-22	101	Interlake	Blue
85	102	09-AUG-23	102	Interlake	Red
58	102	11-OCT-22	102	Interlake	Red
29	102	10-OCT-22	102	Interlake	Red
64	103	11-JUN-23	103	Clipper	Green
31	103	10-AUG-22	103	Clipper	Green
95	104	09-AUG-23	104	Marine	Red
71	104	11-DEC-22	104	Marine	Red
32	104	10-JUL-23	104	Marine	Red
-	-	-	106	Patratu	Red
-	-	-	105	Patratu	Blue

12 rows returned in 0.02 seconds CSV Export

9. **Write down the query to fetch SID & COLOR who have the same color boat.**

```
SELECT Sailors.SID, Boat.COLOR
FROM Sailors
JOIN Reserves ON Sailors.SID = Reserves.SID
JOIN Boat ON Reserves.BID = Boat.BID
GROUP BY Sailors.SID, Boat.COLOR
HAVING COUNT(DISTINCT Boat.BID) > 1;
```

Results Explain Describe Saved SQL History

SID	COLOR
74	Blue
32	Red
71	Red
95	Red
31	Green
85	Red
58	Red
64	Green
22	Blue
29	Red

10 rows returned in 0.02 seconds CSV Export

10. Find the name and age of sailor who is having red and blue color boat.

```
SELECT SNAME, AGE
FROM Sailors
JOIN Reserves ON Sailors.SID = Reserves.SID
JOIN Boat ON Reserves.BID = Boat.BID
WHERE Boat.COLOR IN ('Red', 'Blue')
GROUP BY SNAME, AGE
HAVING COUNT(DISTINCT Boat.COLOR) = 2;
```

Results Explain Describe Saved SQL History

SNAME	AGE
Gagan	7
Magan	1
Andy	8
Rawan	10
Zorba	10
Recordo	9
Messi	3
Kohli	3

8 rows returned in 0.00 seconds CSV Export

11. Find the name of boat whose rating is greater than 40.

```
SELECT BName
FROM Boat
WHERE BID IN (
   SELECT BID
   FROM Sailors
   WHERE RATING > 40
);
```

Results Explain Describe Saved SQL History

BNAME
Interlake
Interlake
Clipper
Marine
Patratu
Patratu

6 rows returned in 0.00 seconds CSV Export

12. Perform the natural join between sailors and reserves

```
SELECT *
FROM Sailors
NATURAL JOIN Reserves;
```

Results Explain Describe Saved SQL History

SID	SNAME	AGE	RATING	BID	DAY
22	Gagan	7	45	101	10-OCT-22
29	Magan	1	33	102	10-OCT-22
31	Kite	8	55	103	10-AUG-22
32	Andy	8	25.5	104	10-JUL-23
58	Rawan	10	35	102	11-OCT-22
64	Julia	7	35	103	11-JUN-23
71	Zorba	10	40	104	11-DEC-22
74	Recordo	9	40	101	09-MAY-23
85	Messi	3	25.5	102	09-AUG-23
95	Kohli	3	63.5	104	09-AUG-23

10 rows returned in 0.00 seconds CSV Export

13. Perform the natural join between boat and reserves.

```
SELECT * FROM Boat
NATURAL JOIN Reserves;
```

Results Explain Describe Saved SQL History

BID	BNAME	COLOR	SID	DAY
101	Interlake	Blue	22	10-OCT-22
102	Interlake	Red	29	10-OCT-22
103	Clipper	Green	31	10-AUG-22
104	Marine	Red	32	10-JUL-23
102	Interlake	Red	58	11-OCT-22
103	Clipper	Green	64	11-JUN-23
104	Marine	Red	71	11-DEC-22
101	Interlake	Blue	74	09-MAY-23
102	Interlake	Red	85	09-AUG-23
104	Marine	Red	95	09-AUG-23

10 rows returned in 0.00 seconds CSV Export

14. **Find the boat name and color which was booked by Gagan.**

```
SELECT B.BName, B.Color
FROM Sailors S
JOIN Reserves R ON S.SID = R.SID
JOIN Boat B ON R.BID = B.BID
WHERE S.SNAME = 'Gagan';
```

Results Explain Describe Saved SQL History

BNAME	COLOR
Interlake	Blue

1 rows returned in 0.00 seconds CSV Export

15. **Find the name of boat that was booked in 2023.**

```
SELECT B.BName
FROM Reserves R
JOIN Boat B ON R.BID = B.BID
WHERE TO_CHAR(R.DAY, 'YYYY') = '2023';
```

Results Explain Describe Saved SQL History

BNAME
Interlake
Interlake
Clipper
Marine
Marine

5 rows returned in 0.00 seconds CSV Export

16. **Apply CHECK constraint on rating attribute that check the ratings should be greater than or equal to 20.**

```
ALTER TABLE Sailors
ADD CONSTRAINT chk_rating CHECK (RATING >= 20);
```

Results Explain Describe Saved SQL History

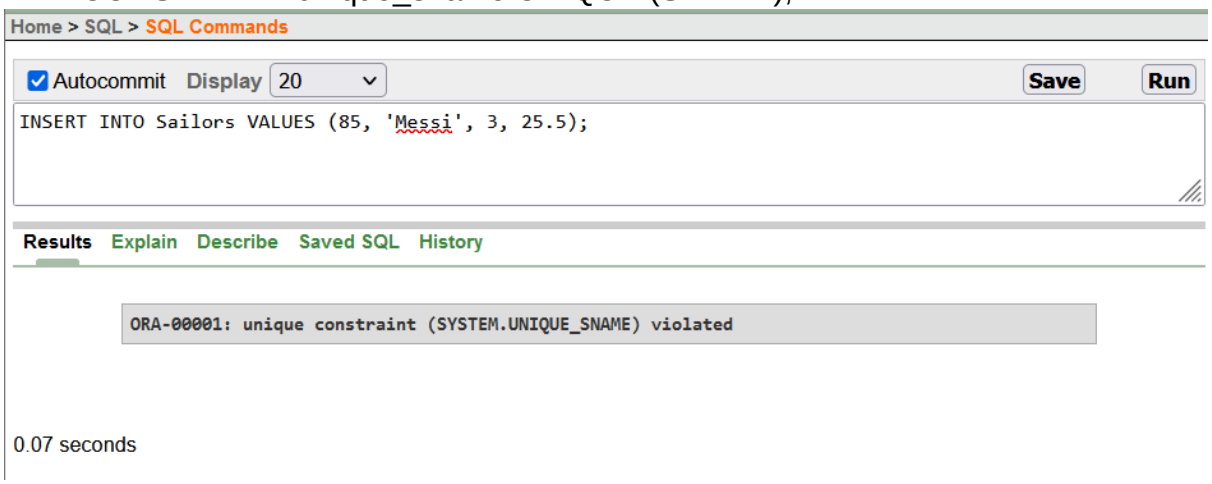
```
Table altered.
```

0.09 seconds

17. **Apply UNIQUE, NOT NULL constraints on SName and verify it by inserting NULL value in it.**

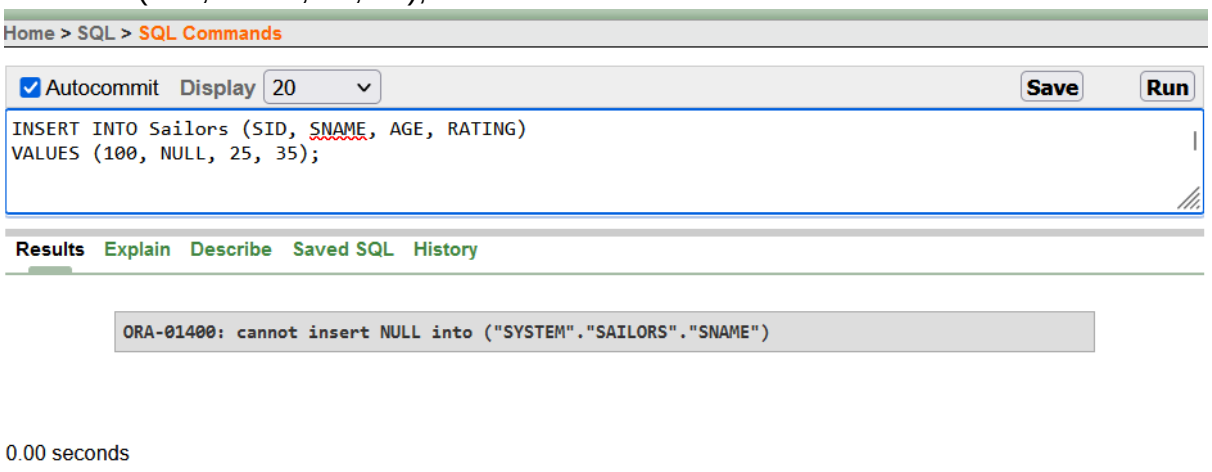
```
-- UNIQUE constraint
ALTER TABLE Sailors
```

ADD CONSTRAINT unique_sname UNIQUE (SNAME);



--NOT NULL constraint
ALTER TABLE Sailors
MODIFY (SNAME NOT NULL);

-- Try to insert NULL value to test NOT NULL constraint
INSERT INTO Sailors (SID, SNAME, AGE, RATING)
VALUES (100, NULL, 25, 35);



18. Show the working of followings:

• **ON DELETE CASCADE:** if a row of the referenced table is deleted, then all matching rows in the referencing table are deleted.

• **ON DELETE SET NULL:** if a row of the referenced table is deleted, then all referencing columns in all matching rows of the referencing table to be set to null.

• **ON DELETE SET DEFAULT:** if a row of the referenced table is deleted, then all referencing columns in all matching rows of the referencing table to be set to the column's default value.

• **ON DELETE RESTRICT:** it is prohibited to delete a row of the referenced table if that row has any matching rows in the referencing table.

• **ON DELETE NO ACTION (the default):** there is no referential delete action; the referential constraint only specifies a constraint check.

Syntax: create table ChildTable
(attr1 int not Null,
attr2_id int not Null, foreign key(attr1)
references ParentTable(attr3) on delete cascade
);

1. **ON DELETE CASCADE:**

```
CREATE TABLE ParentTable (
    attr3 INT PRIMARY KEY
);

CREATE TABLE ChildTable (
    attr1 INT NOT NULL,
    attr2_id INT NOT NULL,
    FOREIGN KEY (attr1) REFERENCES ParentTable(attr3) ON DELETE
CASCADE
);
```

In this example, if a row in ParentTable is deleted, all matching rows in ChildTable with the corresponding attr1 values will also be automatically deleted.

2. **ON DELETE SET NULL**

```
CREATE TABLE ParentTable (
    attr3 INT PRIMARY KEY
);

CREATE TABLE ChildTable (
    attr1 INT,
    attr2_id INT,
    FOREIGN KEY (attr1) REFERENCES ParentTable(attr3) ON DELETE
SET NULL
);
```

If a row in ParentTable is deleted, all corresponding attr1 values in ChildTable will be set to NULL.

3. **ON DELETE SET DEFAULT**

```
CREATE TABLE ParentTable (
```

```
    attr3 INT PRIMARY KEY
);

CREATE TABLE ChildTable (
    attr1 INT DEFAULT 0,
    attr2_id INT,
    FOREIGN KEY (attr1) REFERENCES ParentTable(attr3) ON DELETE
SET DEFAULT);
```

If a row in ParentTable is deleted, all corresponding attr1 values in ChildTable will be set to the default value of 0 (or any other default value specified).

4. **ON DELETE RESTRICT**

```
CREATE TABLE ParentTable (
    attr3 INT PRIMARY KEY
);

CREATE TABLE ChildTable (
    attr1 INT,
    attr2_id INT,
    FOREIGN KEY (attr1) REFERENCES ParentTable(attr3) ON DELETE
RESTRICT);
```

This option restricts the deletion of a row in ParentTable if there are any matching rows in ChildTable.

5. **ON DELETE NO ACTION**

```
CREATE TABLE ParentTable (
    attr3 INT PRIMARY KEY
);

CREATE TABLE ChildTable (
    attr1 INT,
    attr2_id INT,
    FOREIGN KEY (attr1) REFERENCES ParentTable(attr3) ON DELETE
NO ACTION);
```

This is the default option. It specifies that no action should be taken if a row in ParentTable is deleted, and it is mainly used for checking referential integrity without any automatic action.

DBMS LAB ASSIGNMENT 3

Employee_Id	First_Name	Last_Name	Email	Phone_No	Hire_Date	Job_Id	Salary	Manager_Id	Department_Id
100	Ritik	Raj	RITRAJ	3456286544	1987-06-17	AD_PRES	24000.00	0	90
101	Yash	Kumar	YASKUM	2748922645	1987-06-18	AD_VP	17000.00	100	90
102	David	Roy	DAVROY	2745987567	1987-06-19	AD_VP	17000.00	100	60
103	Suraj	Singh	SURSIN	4328769801	1987-06-20	IT_PROG	9000.00	102	60
104	Bruce	Jha	BRUJHA	7634981746	1987-06-21	IT_PROG	6000.00	103	60
105	Nancy	Rai	NANRAI	9918387658	1987-06-22	IT_PROG	4800.00	103	60
106	Yug	Patel	YUGPAT	1264839275	1987-06-23	IT_PROG	4800.00	103	60
107	Nayan	Kumar	NAYKUM	7498275689	1987-06-24	IT_PROG	4200.00	103	100
108	Harshit	Austin	HARAUS	8746018731	1987-06-25	FI_MGR	12000.00	101	100
109	Jiya	Kumari	JIYKUM	3198376549	1987-06-26	FI_ACCOUNT	9000.00	108	100

Creating Table Employee

```
CREATE TABLE Employee (
    Employee_Id NUMBER(5) PRIMARY KEY,
    First_Name VARCHAR2(50),
    Last_Name VARCHAR2(50),
    Email VARCHAR2(50),
    Phone_No VARCHAR2(15),
    Hire_Date DATE,
    Job_Id VARCHAR2(20),
    Salary NUMBER(10,2),
    Manager_Id NUMBER(5),
    Department_Id NUMBER(5)
);
```

```
INSERT INTO Employee VALUES (100, 'Ritik', 'Raj', 'RITRAJ', '3456286544',
TO_DATE('1987-06-17', 'YYYY-MM-DD'), 'AD_PRES', 24000.00, 0, 90);
```

```
INSERT INTO Employee VALUES (101, 'Yash Kumar', 'YASKUM', '2748922645',
TO_DATE('1987-06-18', 'YYYY-MM-DD'), 'AD_VP', 17000.00, 100, 90);
```

INSERT INTO Employee VALUES (102, 'David', 'Roy', 'DAVROY', '2745987567', TO_DATE('1987-06-19', 'YYYY-MM-DD'), 'AD_VP', 17000.00, 100, 60);

INSERT INTO Employee VALUES (103, 'Suraj Singh', 'SURSIN', '4328769801', TO_DATE('1987-06-20', 'YYYY-MM-DD'), 'IT_PROG', 9000.00, 102, 60);

INSERT INTO Employee VALUES (104, 'Bruce', 'Jha', 'BRUJHA', '7634981746', TO_DATE('1987-06-21', 'YYYY-MM-DD'), 'IT_PROG', 6000.00, 103, 60);

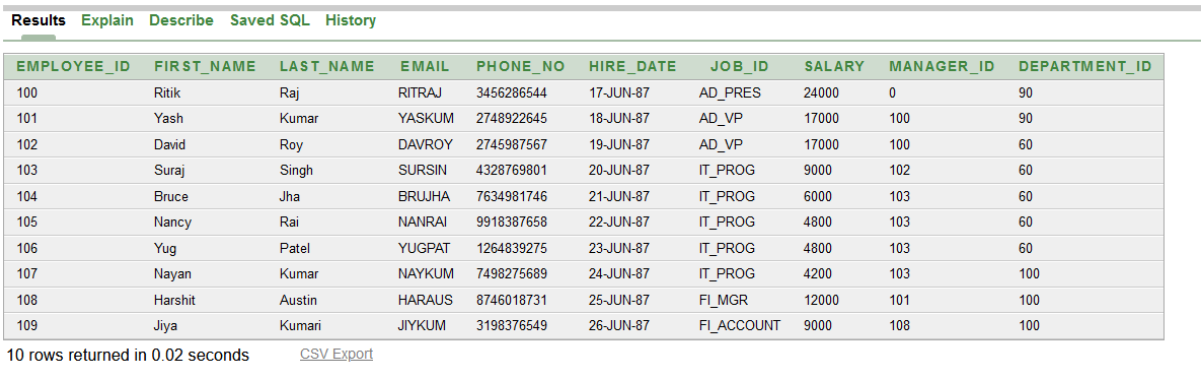
INSERT INTO Employee VALUES (105, 'Nancy', 'Rai', 'NANRAI', '9918387658', TO_DATE('1987-06-22', 'YYYY-MM-DD'), 'IT_PROG', 4800.00, 103, 60);

INSERT INTO Employee VALUES (106, 'Yug', 'Patel', 'YUGPAT', '1264839275', TO_DATE('1987-06-23', 'YYYY-MM-DD'), 'IT_PROG', 4800.00, 103, 60);

INSERT INTO Employee VALUES (107, 'Nayan Kumar', 'NAYKUM', '7498275689', TO_DATE('1987-06-24', 'YYYY-MM-DD'), 'IT_PROG', 4200.00, 103, 100);

INSERT INTO Employee VALUES (108, 'Harshit Austin', 'HARAUS', '8746018731', TO_DATE('1987-06-25', 'YYYY-MM-DD'), 'FI_MGR', 12000.00, 101, 100),

INSERT INTO Employee VALUES (109, 'Jiya Kumari', 'JIYKUM', '3198376549', TO_DATE('1987-06-26', 'YYYY-MM-DD'), 'FI_ACCOUNT', 9000.00, 108, 100);



Results Explain Describe Saved SQL History

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	EMAIL	PHONE_NO	HIRE_DATE	JOB_ID	SALARY	MANAGER_ID	DEPARTMENT_ID
100	Ritik	Raj	RITRAJ	3456286544	17-JUN-87	AD_PRES	24000	0	90
101	Yash	Kumar	YASKUM	2748922645	18-JUN-87	AD_VP	17000	100	90
102	David	Roy	DAVROY	2745987567	19-JUN-87	AD_VP	17000	100	60
103	Suraj	Singh	SURSIN	4328769801	20-JUN-87	IT_PROG	9000	102	60
104	Bruce	Jha	BRUJHA	7634981746	21-JUN-87	IT_PROG	6000	103	60
105	Nancy	Rai	NANRAI	9918387658	22-JUN-87	IT_PROG	4800	103	60
106	Yug	Patel	YUGPAT	1264839275	23-JUN-87	IT_PROG	4800	103	60
107	Nayan	Kumar	NAYKUM	7498275689	24-JUN-87	IT_PROG	4200	103	100
108	Harshit	Austin	HARAUS	8746018731	25-JUN-87	FI_MGR	12000	101	100
109	Jiya	Kumari	JIYKUM	3198376549	26-JUN-87	FI_ACCOUNT	9000	108	100

10 rows returned in 0.02 seconds CSV Export

1. **Write a PL/SQL block to calculate the incentive of an employee whose ID is 100.**

```
DECLARE
   v_employee_id NUMBER := 100;
   v_salary NUMBER;
   v_incentive NUMBER;
BEGIN
   -- Get the salary of the employee
   SELECT Salary INTO v_salary FROM Employee WHERE Employee_Id =
v_employee_id;

   -- Calculate incentive (you can customize the incentive calculation logic)
   v_incentive := v_salary * 0.1; -- 10% of salary as incentive
```

```
    -- Display or use the incentive value as needed
    DBMS_OUTPUT.PUT_LINE('Incentive for Employee ' || v_employee_id || ': '
|| v_incentive);
END;
/
```


```
Incentive for Employee 100: 2400

Statement processed.


0.01 seconds
```

2. **Write a code in PL/SQL TO create a trigger that automatically updates a 'last_modified' timestamp whenever a row in a specific table is updated.**

```
Alter TABLE employee Add
last_modified TIMESTAMP;

CREATE OR REPLACE TRIGGER update_last_modified
BEFORE UPDATE ON employee
FOR EACH ROW
BEGIN
:NEW.last_modified := SYSTIMESTAMP;
END;
/
```


```
Trigger created.


0.16 seconds
```

3. **Write a code in PL/SQL to create a trigger that checks for different values in a specific column and raises an exception if found.**

```
CREATE OR REPLACE TRIGGER check_last_name
BEFORE INSERT OR UPDATE ON Employee
FOR EACH ROW
BEGIN
    -- Check for specific values in the 'last_name' column
    IF :NEW.last_name= 'INVALID' THEN
        -- Raise an exception if 'INVALID' values are found
        RAISE_APPLICATION_ERROR(-20001, 'Invalid value in the last_name
column: ' || :NEW.last_name);
    END IF;
```

```
END;
```


```
Trigger created.
```

0.05 seconds

4. **Write a code in PL/SQL to create a trigger that checks for duplicate values in a specific column and raises an exception if found.**

```
CREATE OR REPLACE TRIGGER prevent_duplicates
BEFORE INSERT ON Employee
FOR EACH ROW
DECLARE
v_count NUMBER;
BEGIN
   -- Check if the new first_name already exists
   SELECT COUNT(*) INTO v_count FROM Employee WHERE first_name =
:NEW.first_name;
      -- If duplicate value found, raise an error
   IF v_count> 0 THEN
      RAISE_APPLICATION_ERROR(-20001, 'first name already exists.');
   END IF;
END;
/
```


```
Trigger created.
```

0.02 seconds

5. **Create a trigger that checks if "value" is greater than 1000 for any new row inserted to "Test" table, then insert a row to Audit table?**

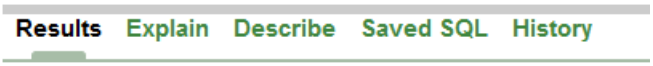
```
CREATE TABLE Test
(
test_id INTEGER,
value NUMBER,
test_date DATE
);

CREATE TABLE Test_Audit
(
```

```
test_id INTEGER,
value NUMBER,
test_date DATE
);

CREATE OR REPLACE TRIGGER check_and_insert_audit
BEFORE INSERT ON Test
FOR EACH ROW
BEGIN
   -- Check if the value is greater than 1000
   IF :NEW.value > 1000 THEN
      -- Insert a corresponding row into Test_audit
      INSERT INTO Test_audit (test_id, value, test_date)
      VALUES (:NEW.test_id, :NEW.value, :NEW.test_date);
   END IF;
END;
/
```


```
Trigger created.

0.03 seconds
```

6. **Write a PL/SQL function to return the maximum bit length of the Last_Names in the Employees table.**

```
DECLARE
v_bit_length NUMBER;
v_max_bit_length NUMBER := 0;
BEGIN
  FOR emp IN (SELECT Last_Name FROM Employee) LOOP
v_bit_length := LENGTHB(trim(emp.Last_Name)) * 8;
   IF v_bit_length > v_max_bit_length THEN
         v_max_bit_length := v_bit_length;
   END IF;
  END LOOP;
  DBMS_OUTPUT.PUT_LINE('Maximum Bit Length: ' || v_max_bit_length);
END;
/
```




```
Maximum Bit Length: 48

Statement processed.


0.03 seconds
```

7. **Write a PL/SQL block to calculate the bit length of the employee's first name in the employees table for all records.**

```
DECLARE
v_bit_length NUMBER;
BEGIN
  FOR emp IN (SELECT Last_Name FROM Employee) LOOP
v_bit_length := LENGTHB(trim(emp.Last_Name)) * 8;
    DBMS_OUTPUT.PUT_LINE('Bit length of ' || emp.Last_Name || ': ' ||
v_bit_length);
  END LOOP;
END;
/
```


```
Bit length of Raj: 24
Bit length of Kumar: 40
Bit length of Roy: 24
Bit length of Singh: 40
Bit length of Jha: 24
Bit length of Rai: 24
Bit length of Patel: 40
Bit length of Kumar: 40
Bit length of Austin: 48
Bit length of Kumari: 48

Statement processed.


0.02 seconds
```

8. **Write a PL/SQL block to display the character representation of each ASCII value in the range of 65 to 90.**

```
DECLARE
   v_ascii_value NUMBER;
   v_character CHAR(1);
BEGIN
   FOR v_ascii_value IN 65..90 LOOP
      -- Convert ASCII value to character
      v_character := CHR(v_ascii_value);
```

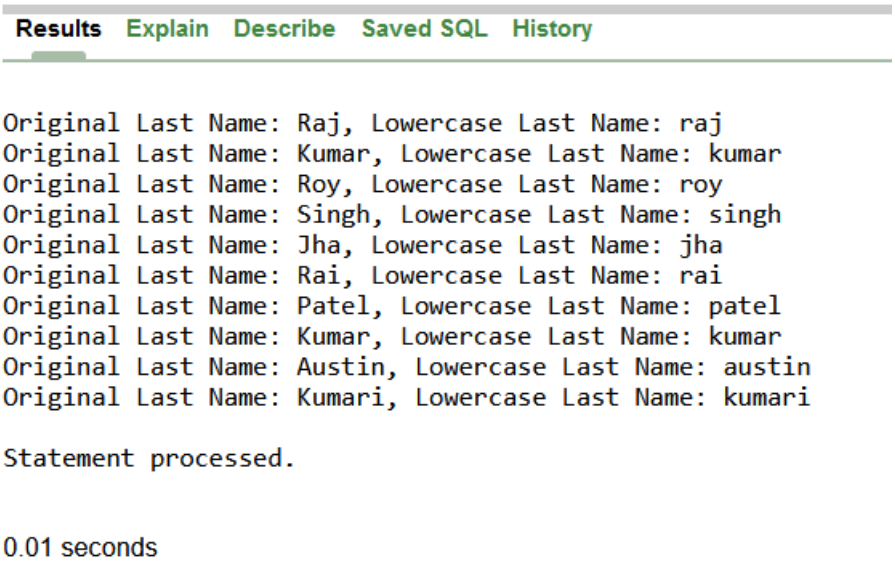
```
        -- Display ASCII value and its character representation
        DBMS_OUTPUT.PUT_LINE('ASCII Value: ' || v_ascii_value || ',
Character: ' || v_character);
    END LOOP;
END;
/
```


```
ASCII Value: 65, Character: A
ASCII Value: 66, Character: B
ASCII Value: 67, Character: C
ASCII Value: 68, Character: D
ASCII Value: 69, Character: E
ASCII Value: 70, Character: F
ASCII Value: 71, Character: G
ASCII Value: 72, Character: H
ASCII Value: 73, Character: I
ASCII Value: 74, Character: J
ASCII Value: 75, Character: K
ASCII Value: 76, Character: L
ASCII Value: 77, Character: M
ASCII Value: 78, Character: N
ASCII Value: 79, Character: O
ASCII Value: 80, Character: P
ASCII Value: 81, Character: Q
ASCII Value: 82, Character: R
ASCII Value: 83, Character: S
ASCII Value: 84, Character: T
ASCII Value: 85, Character: U
ASCII Value: 86, Character: V
ASCII Value: 87, Character: W
ASCII Value: 88, Character: X
ASCII Value: 89, Character: Y
ASCII Value: 90, Character: Z

Statement processed.


0.00 seconds
```

9. **Write a PL/SQL block to convert the last name of each employee in the employees table to lowercase and display the result.**

```
DECLARE
    v_lower_last_name VARCHAR2(50);
BEGIN
    FOR emp IN (SELECT Last_Name FROM Employee) LOOP
        -- Convert the last name to lowercase
        v_lower_last_name := LOWER(emp.Last_Name);

        -- Display the original and lowercase last names
        DBMS_OUTPUT.PUT_LINE('Original Last Name: ' || emp.Last_Name || ',
Lowercase Last Name: ' || v_lower_last_name);
```

```
    END LOOP;
END;
/
```



10. **Write a PL/SQL block that replaces all occurrences of the substring 'IT_PROG' with 'IT PROGRAMMER' in the job titles of employees in the employees table. Display the updated job titles.**

```
DECLARE
    v_job_title Employee.Job_Id%TYPE;
BEGIN
    FOR emp IN (SELECT Job_Id FROM Employee) LOOP
        -- Check if the current job_title is 'IT_PROG'
        IF emp.Job_Id = 'IT_PROG' THEN
            -- Update job_title to 'IT_PROGRAMMER'
            v_job_title := 'IT_PROGRAMMER';
        ELSE
            -- Keep the existing job_title if not 'IT_PROG'
            v_job_title := emp.Job_Id;
        END IF;

        -- Display the updated job_title
        DBMS_OUTPUT.PUT_LINE('Employee Job Title: ' || emp.Job_Id || ',
Updated Job Title: ' || v_job_title);
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

```
Employee Job Title: AD_PRES, Updated Job Title: AD_PRES
Employee Job Title: AD_VP, Updated Job Title: AD_VP
Employee Job Title: AD_VP, Updated Job Title: AD_VP
Employee Job Title: IT_PROG, Updated Job Title: IT_PROGRAMMER
Employee Job Title: IT_PROG, Updated Job Title: IT_PROGRAMMER
Employee Job Title: IT_PROG, Updated Job Title: IT_PROGRAMMER
Employee Job Title: IT_PROG, Updated Job Title: IT_PROGRAMMER
Employee Job Title: IT_PROG, Updated Job Title: IT_PROGRAMMER
Employee Job Title: FI_MGR, Updated Job Title: FI_MGR
Employee Job Title: FI_ACCOUNT, Updated Job Title: FI_ACCOUNT

1 row(s) updated.


0.02 seconds
```

11. **Write a program in PL/SQL to find the number of rows effected by the use of SQL%ROWCOUNT attributes of an implicit cursor.**

```
CREATE TABLE emp_temp AS
  SELECT Employee_Id, First_Name, Last_Name,Email
  FROM Employee;

BEGIN
   UPDATE emp_temp
   SET    email = 'not available'
   WHERE  first_name LIKE 'B%';

   dbms_output.Put_line('Number of record updated: '
                        ||To_char(SQL%rowcount));
END;
/
```

Results Explain Describe Saved SQL History

```
Number of record updated: 1

1 row(s) updated.


0.03 seconds
```

12. **Write a program in PL/SQL to create a table-based record using the %ROWTYPE attribute.**

```
DECLARE
   v_employee_record Employee_Data%ROWTYPE;
BEGIN
   -- Initialize the record with values
```

```
    v_employee_record.Employee_Id := 101;
    v_employee_record.First_Name := 'John';
    v_employee_record.Last_Name := 'Doe';
    -- Assign values to other columns as needed

    -- Display the values in the record
    DBMS_OUTPUT.PUT_LINE('Employee ID: ' ||
v_employee_record.Employee_Id);
    DBMS_OUTPUT.PUT_LINE('First Name: ' ||
v_employee_record.First_Name);
    DBMS_OUTPUT.PUT_LINE('Last Name: ' ||
v_employee_record.Last_Name);
    -- Display other columns as needed
END;
/
```


```
Employee ID: 101
First Name: John
Last Name: Doe

Statement processed.


0.02 seconds
```

13. Write a program in PL/SQL to create an implicit cursor with for loop.

```
DECLARE
    v_employee_salary NUMBER;
BEGIN
    -- Implicit cursor using a FOR loop
    FOR emp IN (SELECT Salary FROM Employee) LOOP
        v_employee_salary := emp.Salary;
        -- You can perform operations on v_employee_salary or display it as
needed
        DBMS_OUTPUT.PUT_LINE('Employee Salary: ' || v_employee_salary);
    END LOOP;
END;
/
```

Results Explain Describe Saved SQL History

```
Employee Salary: 24000
Employee Salary: 17000
Employee Salary: 17000
Employee Salary: 9000
Employee Salary: 6000
Employee Salary: 4800
Employee Salary: 4800
Employee Salary: 4200
Employee Salary: 12000
Employee Salary: 9000

Statement processed.


0.02 seconds
```

14. **Write a program to find the minimum of two values. Here, the procedure takes two numbers using the IN mode and returns their minimum using the OUT parameters.**

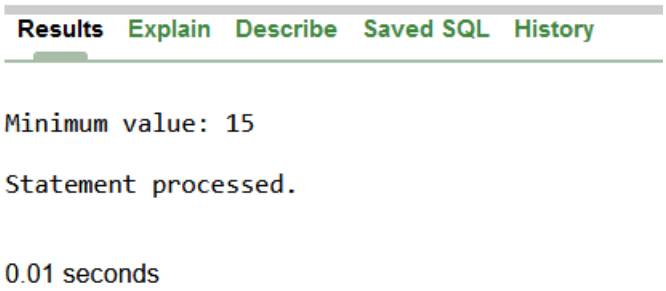
```
-- Create a PL/SQL procedure to find the minimum of two values
CREATE OR REPLACE PROCEDURE find_minimum(
    p_number1 IN NUMBER,
    p_number2 IN NUMBER,
    p_minimum OUT NUMBER
) AS
BEGIN
    -- Check for the minimum of the two values
    IF p_number1 < p_number2 THEN
        p_minimum := p_number1;
    ELSE
        p_minimum := p_number2;
    END IF;
END;
/
```

Results Explain Describe Saved SQL History

```
Procedure created.


0.08 seconds
```

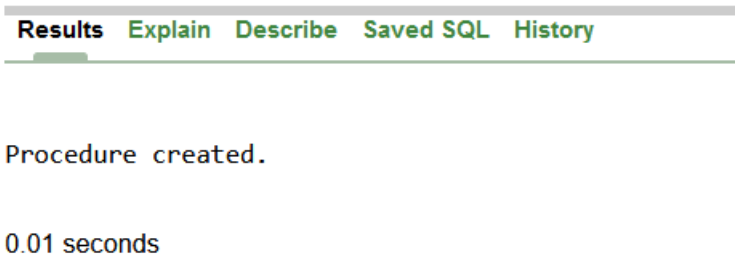
```
-- Example of calling the procedure
DECLARE
    v_result NUMBER;
BEGIN
```

```
    -- Call the procedure with two values
    find_minimum(15, 25, v_result);

    -- Display the result
    DBMS_OUTPUT.PUT_LINE('Minimum value: ' || v_result);
END;
/
```



15. **Write a program in which procedure computes the square of value of a passed value. This example shows how we can use the same parameter to accept a value and then return another result.**

```
-- Create a PL/SQL procedure to compute the square of a value
CREATE OR REPLACE PROCEDURE compute_square(
    p_value IN OUT NUMBER
) AS
BEGIN
    -- Compute the square of the passed value
    p_value := p_value * p_value;
END;
/
```




```
-- Example of calling the procedure
DECLARE
    v_number NUMBER := 5; -- Initial value
BEGIN
    -- Call the procedure with the initial value
    DBMS_OUTPUT.PUT_LINE('Original value: ' || v_number);

    -- Call the procedure to compute the square (passing and returning the
same parameter)
```

```
    compute_square(v_number);

    -- Display the squared result
    DBMS_OUTPUT.PUT_LINE('Squared value: ' || v_number);
END;
/
```

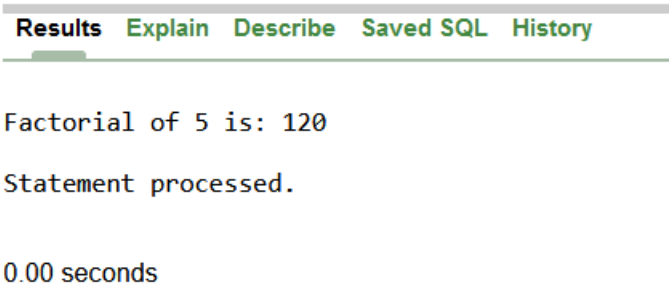

```
Original value: 5
Squared value: 25

Statement processed.


0.00 seconds
```

16. Write a program in PL/SQL using function to calculate the factorial of a number

```
-- Create a PL/SQL function to calculate the factorial of a number
CREATE OR REPLACE FUNCTION calculate_factorial(
    p_number IN NUMBER
) RETURN NUMBER IS
    v_result NUMBER := 1;
BEGIN
    -- Check if the input number is non-negative
    IF p_number < 0 THEN
        -- Return -1 for invalid input (factorial is not defined for negative
numbers)
        RETURN -1;
    END IF;

    -- Calculate the factorial
    FOR i IN 1..p_number LOOP
        v_result := v_result * i;
    END LOOP;

    -- Return the factorial result
    RETURN v_result;
END;
/
```


```
Function created.


0.02 seconds
```

```
-- Example of calling the function
DECLARE
    v_input_number NUMBER := 5; -- Change this to the desired number
    v_factorial_result NUMBER;
BEGIN
    -- Call the function to calculate the factorial
    v_factorial_result := calculate_factorial(v_input_number);

    -- Display the result
    IF v_factorial_result = -1 THEN
        DBMS_OUTPUT.PUT_LINE('Factorial is not defined for negative
numbers.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('Factorial of ' || v_input_number || ' is: ' ||
v_factorial_result);
    END IF;
END;
/
```



17. Create the CUSTOMER table.

```
CREATE TABLE Customer (
    ID NUMBER PRIMARY KEY,
    NAME VARCHAR2(50),
    AGE NUMBER,
    ADDRESS VARCHAR2(100),
    SALARY NUMBER
);
INSERT INTO Customer VALUES (1, 'Ram', 23, 'Allahabad', 12000);
INSERT INTO Customer VALUES (2, 'Suresh', 22, 'Kanpur', 12200);
INSERT INTO Customer VALUES (3, 'Mahi', 24, 'Ghaziabad', 24000);
INSERT INTO Customer VALUES (4, 'Chandan', 25, 'Noida', 26000);
INSERT INTO Customer VALUES (5, 'Aakash', 21, 'Paris', 28000);
INSERT INTO Customer VALUES (6, 'Sulekha', 20, 'Delhi', 30000);
```

Results Explain Describe Saved SQL History

ID	NAME	AGE	ADDRESS	SALARY
1	Ram	23	Allahabad	12000
2	Suresh	22	Kanpur	12200
3	Mahi	24	Ghaziabad	24000
4	Chandan	25	Noida	26000
5	Aakash	21	Paris	28000
6	Sulekha	20	Delhi	30000

6 rows returned in 0.00 seconds CSV Export

18. **Write a program to update the CUSTOMER table and increase the the salary of each customer by 1000. Here, SQL%ROWCOUNT attribute is used to determine the number of rows affected.**

```
DECLARE
   v_updated_rows NUMBER;

BEGIN
   -- Update the salary for each customer
   UPDATE Customer
   SET SALARY = SALARY + 1000;

   -- Get the number of rows affected
   v_updated_rows := SQL%ROWCOUNT;

   -- Display the number of rows affected
   DBMS_OUTPUT.PUT_LINE('Number of rows updated: ' ||
v_updated_rows);
END;
/
```

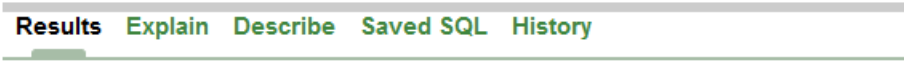
Results Explain Describe Saved SQL History

```
Number of rows updated: 6

1 row(s) updated.


0.00 seconds
```

19. **Using explicit cursor write a program to retrieve the customer name and address from CUSTOMER table.**

```
DECLARE
   -- Declare variables to store customer data
   v_customer_name Customer.NAME%TYPE;
   v_customer_address Customer.ADDRESS%TYPE;
```

```
    -- Declare an explicit cursor
    CURSOR customer_cursor IS
        SELECT NAME, ADDRESS
        FROM Customer;

BEGIN
    -- Open the cursor
    OPEN customer_cursor;

    -- Fetch data from the cursor
    LOOP
        FETCH customer_cursor INTO v_customer_name, v_customer_address;

        -- Exit the loop if no more rows
        EXIT WHEN customer_cursor%NOTFOUND;

        -- Display customer name and address
        DBMS_OUTPUT.PUT_LINE('Customer Name: ' || v_customer_name || ',
Address: ' || v_customer_address);
    END LOOP;

    -- Close the cursor
    CLOSE customer_cursor;
END;
/
```




```
Customer Name: Ram, Address: Allahabad
Customer Name: Suresh, Address: Kanpur
Customer Name: Mahi, Address: Ghaziabad
Customer Name: Chandan, Address: Noida
Customer Name: Aakash, Address: Paris
Customer Name: Sulekha, Address: Delhi

Statement processed.


0.02 seconds
```