Advanced Database Management Systems
Experiment-5
Use of different SQL clauses and join

Aryan Mohan
500092142
Batch- 2

```
CREATE DATABASE LabExperiment5;

use LabExperiment5;

CREATE TABLE DEPARTMENT ( DEPTNO INTEGER NOT NULL,
DNAME VARCHAR(15) NOT NULL, LOC VARCHAR(30), PRIMARY
KEY (DEPTNO) );


CREATE TABLE EMPLOYEE ( EMPNO INTEGER NOT NULL, ENAME
VARCHAR(30) NOT NULL, JOB VARCHAR(20) NOT NULL, MGR
INTEGER, HIREDATE DATE, SAL INTEGER, COMM INTEGER,
DEPTNO INTEGER PRIMARY KEY (EMPNO), FOREIGN KEY (DEPTNO)
REFERENCES DEPARTMENT(DEPTNO) );


insert into DEPARTMENT values(10, 'ACCOUNTING','NEW YORK');
insert into DEPARTMENT values(20, 'RESEARCH','DALLAS');
insert into DEPARTMENT values( 30, 'SALES','CHICAGO');
insert into DEPARTMENT values(40, 'OPERATIONS','BOSTON');


insert into EMPLOYEE values(7369,'SMITH','CLERK', 7902 , '17-DEC-
80',500,800,20);
insert into EMPLOYEE values(7499, 'ALLEN','SALESMAN', 7698,'20-FEB-
81',1600,300,30);
insert into EMPLOYEE values(7521, 'WARD', 'SALESMAN',7698, '22-FEB-
81', 1250,500,30);
insert into EMPLOYEE values(7566, 'JONES','MANAGER', 7839,'02-APR-
81',2975,0,20);
insert into EMPLOYEE values(7654, 'MARTIN', 'SALESMAN',7698, '28-SEP-
81',1250,1400,30);
insert into EMPLOYEE values(7698, 'BLAKE', 'MANAGER',7839,'01-MAY-
81',2850,0,30);
```

```
insert into EMPLOYEE values(7782, 'CLARK', 'MANAGER',7839,'09-JUN-81',2450,0,10);
insert into EMPLOYEE values(7788, 'SCOTT', 'ANALYST',7566,'09-DEC-82',3000,0,20);
insert into EMPLOYEE values(7839, 'KING', 'PRESIDENT',7599,'17-NOV-81',5000,0,10);
insert into EMPLOYEE values(7844,'TURNER', 'SALESMAN', 7698,'08-SEP-81',1500,0,30);
insert into EMPLOYEE values(7876,'ADAMS', 'CLERK', 7788,'12-JAN-83',1100,0,20);
insert into EMPLOYEE values(7900,'JAMES', 'CLERK',7698,'03-DEC-81',950,0,30);
insert into EMPLOYEE values(7902,'FORD', 'ANALYST',7566,'03-DEC-81',3000,0,20);
insert into EMPLOYEE values(7934,'MILLER', 'CLERK',7782,'23-JAN-82',1300,0,10);

select * from EMPLOYEE;

select * from DEPARTMENT;


-----Experimen 5
--1. List the Deptno where there are no emps.
select deptno, count(*) from employee group by deptno having count(*)=0;
```

Output:

deptno	(No column name)

```
--2. List the No.of emp's and Avg salary within each department for each job.
select count(*),avg(sal),deptno,job from employee group by deptno,job;
```

Output:

	(No column name)	(No column name)	deptno	job
1	2	3000	20	ANALYST
2	1	1300	10	CLERK
3	2	800	20	CLERK
4	1	950	30	CLERK
5	1	2450	10	MANAGER
6	1	2975	20	MANAGER
7	1	2850	30	MANAGER
8	1	5000	10	PRESIDENT
9	4	1400	30	SALESMAN

--3. Find the maximum average salary drawn for each job except for 'President'.

```
select avg(SAL),job from EMPLOYEE where JOB!= 'PRESIDENT' group by JOB;
```

Output:

	(No column name)	job
1	3000	ANALYST
2	962	CLERK
3	2758	MANAGER
4	1400	SALESMAN

--4. List the department details where at least two emps are working.

```
select deptno ,count(*) from EMPLOYEE group by deptno having count(*)>=2
```

Output:

	deptno	(No column name)
1	10	3
2	20	5
3	30	6

--5. List the no. of emps in each department where the no. is more than 3.

```
select deptno,count(*) from EMPLOYEE group by deptno having count(*)>3;
```

Output:

	deptno	(No column name)
1	20	5
2	30	6

--6. List the names of the emps who are getting the highest sal dept wise.

```
select E.ename,E.deptno, E.SAL from EMPLOYEE E where E.sal in (select max(sal)from EMPLOYEE group by deptno);
```

Output:

	ename	deptno	SAL
1	BLAKE	30	2850
2	SCOTT	20	3000
3	KING	10	5000
4	FORD	20	3000

--7. List the Deptno and their average salaries for dept with the average salary less than the averages for all departments.

```
select deptno,avg(sal) from EMPLOYEE group by deptno having
avg(sal)<(select avg(Sal) from EMPLOYEE);
```

Output:

	deptno	(No column name)
1	30	1566

---JOIN OPERATIONS

---Equi join

```
SELECT EMPLOYEE.EMPNO, EMPLOYEE.ENAME,
EMPLOYEE.DEPTNO, DEPARTMENT.DEPTNO, DEPARTMENT.LOC
FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.DEPTNO =
DEPARTMENT.DEPTNO;
```

Output:

	EMPNO	ENAME	DEPTNO	DEPTNO	LOC
1	7369	SMITH	20	20	DALLAS
2	7499	ALLEN	30	30	CHICAGO
3	7521	WARD	30	30	CHICAGO
4	7566	JONES	20	20	DALLAS
5	7654	MARTIN	30	30	CHICAGO
6	7698	BLAKE	30	30	CHICAGO
7	7782	CLARK	10	10	NEW YORK
8	7788	SCOTT	20	20	DALLAS
9	7839	KING	10	10	NEW YORK
10	7844	TURNER	30	30	CHICAGO
11	7876	ADAMS	20	20	DALLAS
12	7900	JAMES	30	30	CHICAGO
13	7902	FORD	20	20	DALLAS
14	7934	MILLER	10	10	NEW YORK

----Outer Joins

```
SELECT e.ENAME, e.DEPTNO, d.DNAME FROM EMPLOYEE e,
DEPARTMENT d WHERE e.DEPTNO = d.DEPTNO;
```

Output:

	ENAME	DEPTNO	DNAME
1	SMITH	20	RESEARCH
2	ALLEN	30	SALES
3	WARD	30	SALES
4	JONES	20	RESEARCH
5	MARTIN	30	SALES
6	BLAKE	30	SALES
7	CLARK	10	ACCOUNTING
8	SCOTT	20	RESEARCH
9	KING	10	ACCOUNTING
10	TURNER	30	SALES
11	ADAMS	20	RESEARCH
12	JAMES	30	SALES
13	FORD	20	RESEARCH
14	MILLER	10	ACCOUNTING

```
SELECT e.ename, d.DEPTNO, d.dname FROM EMPLOYEE e,
DEPARTMENT d WHERE e.deptno =d.deptno ORDER BY e.deptno;
```

Output:

	ename	DEPTNO	dname
1	CLARK	10	ACCOUNTING
2	KING	10	ACCOUNTING
3	MILLER	10	ACCOUNTING
4	FORD	20	RESEARCH
5	ADAMS	20	RESEARCH
6	SCOTT	20	RESEARCH
7	SMITH	20	RESEARCH
8	JONES	20	RESEARCH
9	MARTIN	30	SALES
10	BLAKE	30	SALES
11	ALLEN	30	SALES
12	WARD	30	SALES
13	TURNER	30	SALES
14	JAMES	30	SALES

```
SELECT e.ename, d.DEPTNO, d.dname FROM EMPLOYEE e INNER JOIN
DEPARTMENT AS d ON e.deptno =d.deptno ORDER BY e.deptno;
```

Output:

	ename	DEPTNO	dname
1	CLARK	10	ACCOUNTING
2	KING	10	ACCOUNTING
3	MILLER	10	ACCOUNTING
4	FORD	20	RESEARCH
5	ADAMS	20	RESEARCH
6	SCOTT	20	RESEARCH
7	SMITH	20	RESEARCH
8	JONES	20	RESEARCH
9	MARTIN	30	SALES
10	BLAKE	30	SALES
11	ALLEN	30	SALES
12	WARD	30	SALES
13	TURNER	30	SALES
14	JAMES	30	SALES

```
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE
e INNER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;
```

Output:

	ename	EMPNO	SAL	deptno	dname
1	SMITH	7369	500	20	RESEARCH
2	ALLEN	7499	1600	30	SALES
3	WARD	7521	1250	30	SALES
4	JONES	7566	2975	20	RESEARCH
5	MARTIN	7654	1250	30	SALES
6	BLAKE	7698	2850	30	SALES
7	CLARK	7782	2450	10	ACCOUNTING
8	SCOTT	7788	3000	20	RESEARCH
9	KING	7839	5000	10	ACCOUNTING
10	TURNER	7844	1500	30	SALES
11	ADAMS	7876	1100	20	RESEARCH
12	JAMES	7900	950	30	SALES
13	FORD	7902	3000	20	RESEARCH
14	MILLER	7934	1300	10	ACCOUNTING

```
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE
e LEFT OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;
```

Output:

	ename	EMPNO	SAL	deptno	dname
1	SMITH	7369	500	20	RESEARCH
2	ALLEN	7499	1600	30	SALES
3	WARD	7521	1250	30	SALES
4	JONES	7566	2975	20	RESEARCH
5	MARTIN	7654	1250	30	SALES
6	BLAKE	7698	2850	30	SALES
7	CLARK	7782	2450	10	ACCOUNTING
8	SCOTT	7788	3000	20	RESEARCH
9	KING	7839	5000	10	ACCOUNTING
10	TURNER	7844	1500	30	SALES
11	ADAMS	7876	1100	20	RESEARCH
12	JAMES	7900	950	30	SALES
13	FORD	7902	3000	20	RESEARCH
14	MILLER	7934	1300	10	ACCOUNTING

```
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE e RIGHT OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;
```

Output:

	ename	EMPNO	SAL	deptno	dname
1	CLARK	7782	2450	10	ACCOUNTING
2	KING	7839	5000	10	ACCOUNTING
3	MILLER	7934	1300	10	ACCOUNTING
4	SMITH	7369	500	20	RESEARCH
5	JONES	7566	2975	20	RESEARCH
6	SCOTT	7788	3000	20	RESEARCH
7	ADAMS	7876	1100	20	RESEARCH
8	FORD	7902	3000	20	RESEARCH
9	ALLEN	7499	1600	30	SALES
10	WARD	7521	1250	30	SALES
11	MARTIN	7654	1250	30	SALES
12	BLAKE	7698	2850	30	SALES
13	TURNER	7844	1500	30	SALES
14	JAMES	7900	950	30	SALES
15	NULL	NULL	NULL	40	OPERATIONS

```
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE e FULL OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;
```

Output:

	ename	EMPNO	SAL	deptno	dname
1	SMITH	7369	500	20	RESEARCH
2	ALLEN	7499	1600	30	SALES
3	WARD	7521	1250	30	SALES
4	JONES	7566	2975	20	RESEARCH
5	MARTIN	7654	1250	30	SALES
6	BLAKE	7698	2850	30	SALES
7	CLARK	7782	2450	10	ACCOUNTING
8	SCOTT	7788	3000	20	RESEARCH
9	KING	7839	5000	10	ACCOUNTING
10	TURNER	7844	1500	30	SALES
11	ADAMS	7876	1100	20	RESEARCH
12	JAMES	7900	950	30	SALES
13	FORD	7902	3000	20	RESEARCH
14	MILLER	7934	1300	10	ACCOUNTING
15	NULL	NULL	NULL	40	OPERATIONS

```
SELECT * FROM EMPLOYEE e FULL OUTER JOIN DEPARTMENT AS d
ON e.deptno =d.deptno;
```

Output:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980-12-17	500	800	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	1981-04-02	2975	0	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	1981-06-09	2450	0	10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	1982-12-09	3000	0	20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT	7599	1981-11-17	5000	0	10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	1983-01-12	1100	0	20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	1981-12-03	950	0	30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	1981-12-03	3000	0	20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	1982-01-23	1300	0	10	10	ACCOUNTING	NEW YORK
15	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	40	OPERATIONS	BOSTON

```
SELECT * FROM EMPLOYEE e RIGHT OUTER JOIN DEPARTMENT AS
d ON e.deptno =d.deptno;
```

Output:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
1	7782	CLARK	MANAGER	7839	1981-06-09	2450	0	10	10	ACCOUNTING	NEW YORK
2	7839	KING	PRESIDENT	7599	1981-11-17	5000	0	10	10	ACCOUNTING	NEW YORK
3	7934	MILLER	CLERK	7782	1982-01-23	1300	0	10	10	ACCOUNTING	NEW YORK
4	7369	SMITH	CLERK	7902	1980-12-17	500	800	20	20	RESEARCH	DALLAS
5	7566	JONES	MANAGER	7839	1981-04-02	2975	0	20	20	RESEARCH	DALLAS
6	7788	SCOTT	ANALYST	7566	1982-12-09	3000	0	20	20	RESEARCH	DALLAS
7	7876	ADAMS	CLERK	7788	1983-01-12	1100	0	20	20	RESEARCH	DALLAS
8	7902	FORD	ANALYST	7566	1981-12-03	3000	0	20	20	RESEARCH	DALLAS
9	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30	30	SALES	CHICAGO
10	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30	30	SALES	CHICAGO
11	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30	30	SALES	CHICAGO
12	7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30	30	SALES	CHICAGO
13	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30	30	SALES	CHICAGO
14	7900	JAMES	CLERK	7698	1981-12-03	950	0	30	30	SALES	CHICAGO
15	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	40	OPERATIONS	BOSTON

```
SELECT * FROM EMPLOYEE e LEFT OUTER JOIN DEPARTMENT AS d
ON e.deptno =d.deptno;
```

Output:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980-12-17	500	800	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	1981-04-02	2975	0	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	1981-06-09	2450	0	10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	1982-12-09	3000	0	20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT	7599	1981-11-17	5000	0	10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	1983-01-12	1100	0	20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	1981-12-03	950	0	30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	1981-12-03	3000	0	20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	1982-01-23	1300	0	10	10	ACCOUNTING	NEW YORK

```
SELECT * FROM EMPLOYEE e INNER JOIN DEPARTMENT AS d ON
e.deptno =d.deptno;
```

Output:

	EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO	DEPTNO	DNAME	LOC
1	7369	SMITH	CLERK	7902	1980-12-17	500	800	20	20	RESEARCH	DALLAS
2	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30	30	SALES	CHICAGO
3	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30	30	SALES	CHICAGO
4	7566	JONES	MANAGER	7839	1981-04-02	2975	0	20	20	RESEARCH	DALLAS
5	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30	30	SALES	CHICAGO
6	7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30	30	SALES	CHICAGO
7	7782	CLARK	MANAGER	7839	1981-06-09	2450	0	10	10	ACCOUNTING	NEW YORK
8	7788	SCOTT	ANALYST	7566	1982-12-09	3000	0	20	20	RESEARCH	DALLAS
9	7839	KING	PRESIDENT	7599	1981-11-17	5000	0	10	10	ACCOUNTING	NEW YORK
10	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30	30	SALES	CHICAGO
11	7876	ADAMS	CLERK	7788	1983-01-12	1100	0	20	20	RESEARCH	DALLAS
12	7900	JAMES	CLERK	7698	1981-12-03	950	0	30	30	SALES	CHICAGO
13	7902	FORD	ANALYST	7566	1981-12-03	3000	0	20	20	RESEARCH	DALLAS
14	7934	MILLER	CLERK	7782	1982-01-23	1300	0	10	10	ACCOUNTING	NEW YORK

Advance Database Management Systems Lab

Experiment- 6

To understand the concepts of Views

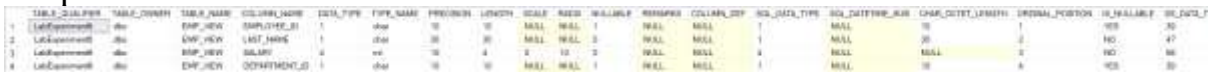
Aryan Mohan

500092142

Batch- 2


```
CREATE DATABASE LabExperiment6;

USE LabExperiment6;

CREATE TABLE EMPLOYEES( Employee_id VARCHAR(10) NOT NULL
PRIMARY KEY, First_Name VARCHAR(30) NOT NULL, Last_Name
VARCHAR(30) NOT NULL, DOB Date, salary DECIMAL(25,0) NOT NULL,
Department_id VARCHAR(10) );

--1 Create View of name emp_view and the column would be Employee_id,
Last_Name, salary and department_id only.
CREATE VIEW emp_view(Employee_id, Last_Name, salary, Department_id)
AS SELECT Employee_id, Last_Name, salary, Department_id FROM
EMPLOYEES;

exec sp_columns emp_view
```

Output:



```
--2) Insert values into view(remove the NOT NULL constraint and then insert
values)

ALTER TABLE EMPLOYEES ALTER COLUMN First_Name varchar(30)
null;

INSERT INTO emp_view VALUES('EMP000111', 'JAIN', 50000,
'INFORMATIC');

SELECT * FROM EMPLOYEES;
```

```
INSERT INTO emp_view VALUES ('EMP000112', 'Joyce', 25000, 'ECE');
INSERT INTO emp_view VALUES ('EMP000113', 'Ramesh',38000, 'ME');
INSERT INTO emp_view VALUES ('EMP000114', 'James', 55000, 'CIVIL');
INSERT INTO emp_view VALUES('EMP000115','Jennifer',65555, 'IT');

SELECT * FROM EMPLOYEES;

SELECT * FROM emp_view;
```

Output:

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DOB	SALARY	DEPARTMENT_ID
1	EMP000111	NULL	JAIN	NULL	50000	INFORMATIC

	EMPLOYEE_ID	FIRST_NAME	LAST_NAME	DOB	SALARY	DEPARTMENT_ID
1	EMP000111	NULL	JAIN	NULL	50000	INFORMATIC
2	EMP000112	NULL	Joyce	NULL	25000	ECE
3	EMP000113	NULL	Ramesh	NULL	38000	ME
4	EMP000114	NULL	James	NULL	55000	CIVIL
5	EMP000115	NULL	Jennifer	NULL	65555	IT

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	EMP000111	JAIN	50000	INFORMATIC
2	EMP000112	Joyce	25000	ECE
3	EMP000113	Ramesh	38000	ME
4	EMP000114	James	55000	CIVIL
5	EMP000115	Jennifer	65555	IT

```
--3) Modify, delete and drop operations are performed on view.
--UPDATE
update emp_view set salary=60000 where Employee_id='EMP000113';

DELETE FROM emp_view WHERE Last_Name='Joyce';

SELECT * FROM emp_view;

CREATE VIEW ADBMS(L_Name, salary) AS SELECT Last_Name, salary
FROM EMPLOYEES;

SELECT * FROM ADBMS;

DROP VIEW ADBMS;
```

Output:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	EMP000111	JAIN	50000	INFORMATIC
2	EMP000113	Ramesh	60000	ME
3	EMP000114	James	55000	CIVIL
4	EMP000115	Jennifer	65555	IT

	L_Name	salary
1	JAIN	50000
2	Ramesh	60000
3	James	55000
4	Jennifer	65555

---4 Creates a view named salary_view. The view shows the employees in department and their annual salary.

```
SELECT * FROM emp_view;
INSERT INTO emp_view VALUES('EMP000116','GUPTA',80000, '20');

CREATE VIEW salary_viewB1 AS SELECT Employee_id, Last_Name, salary
FROM EMPLOYEES WHERE Department_id='20';

SELECT * FROM salary_viewB1;
```

Output:

	EMPLOYEE_ID	LAST_NAME	SALARY	DEPARTMENT_ID
1	EMP000111	JAIN	50000	INFORMATIC
2	EMP000113	Ramesh	60000	ME
3	EMP000114	James	55000	CIVIL
4	EMP000115	Jennifer	65555	IT

	Employee_id	Last_Name	salary
1	EMP000116	GUPTA	80000

Advance Database Management System Lab

Experiment- 7

To understand the concepts of Index

Aryan Mohan

500092142

Batch- 2

```
---1) Create an index of name employee_idx on EMPLOYEES with column
Last_Name, Department_id
CREATE DATABASE LabExperiment7;

USE LabExperiment7;

CREATE TABLE EMPLOYES( Employee_id VARCHAR(10) NOT NULL
PRIMARY KEY, First_Name VARCHAR(30) NOT NULL, Last_Name
VARCHAR(30) NOT NULL, DOB Date, salary DECIMAL(25,0) NOT NULL,
Department_id VARCHAR(10) )

insert into EMPLOYES values(7499, 'ALLEN','Narayan', '20-FEB-
81',1600,'CSE');

SELECT * FROM EMPLOYES;
insert into EMPLOYES values(7521, 'WARD', 'S', '22-FEB-81', 125000,
'AIML');
insert into EMPLOYES values(7566, 'JONES','Wong','02-APR-
81',297500,'AIML');
insert into EMPLOYES values(7654, 'MARTIN', 'SALMAN','28-SEP-
81',125000,'CIVIL');
insert into EMPLOYES values(7698, 'BLAKE', 'NAGER','01-MAY-
81',285000,'BIGDATA');
insert into EMPLOYES values(7782, 'CLARK', 'MAGER','09-JUN-81',245000,
'BIGDATA');
insert into EMPLOYES values(7788, 'SCOTT', 'ANAL','09-DEC-
82',300000,'ME');
insert into EMPLOYES values(7839, 'KING', 'PRESIDENT','17-NOV-
81',500000,'AIML');

CREATE INDEX employee_idx on EMPLOYES(Last_Name, Department_id)
```

Output:

	Employee_id	First_Name	Last_Name	DOB	salary	Department_id
1	7499	ALLEN	Narayan	1981-02-20	1600	CSE

```
--2) Find the ROWID for the above table and create a unique index on
employee_id column of the EMPLOYEES.
CREATE UNIQUE INDEX EMP_UNI ON EMPLOYES(Employee_id)

---3) Create a reverse index on employee_id column of the EMPLOYEES.
CREATE INDEX EMP_REVERSE ON EMPLOYES(First_name) REVERSE;

---4) Create a unique and composite index on employee_id and check whether
there is duplicity of tuples or not.
CREATE INDEX employee_comp on EMPLOYES(First_Name,Last_Name,
DOB,salary);

CREATE UNIQUE INDEX emp_comp on
EMPLOYES(First_Name,Last_Name, DOB,salary);

--5) Create Function-based indexes defined on the SQL functions
UPPER(column_name) or LOWER(column_name) to facilitate case-insensitive
searches(on  column Last_Name).
CREATE TABLE EMPLOYEE( Employee_id VARCHAR(10) NOT NULL
PRIMARY KEY, First_Name VARCHAR(30) NOT NULL, Last_Name
VARCHAR(30) NOT NULL, Last_Name_upper as UPPER(Last_Name),
First_name_lower as LOWER(First_Name), DOB Date, salary
DECIMAL(25,0) NOT NULL, Department_id VARCHAR(10) )

insert into EMPLOYEE values(7566, 'jones','wong','02-APR-
81',297500,'AIML');
insert into EMPLOYEE values(7788, 'scott', 'anal','09-DEC-82',300000,'ME');
insert into EMPLOYEE values(7654, 'MARTIN', 'fox','28-SEP-
81',125000,'CIVIL');

SELECT * FROM EMPLOYEE;

CREATE UNIQUE INDEX emp_fun_index ON
EMPLOYEE(Last_Name_upper);

SELECT Employee_id, First_Name, DOB,salary from EMPLOYEE where
UPPER(Last_Name)= 'WONG';
```

Output:

	Employee_id	First_Name	Last_Name	DOB	salary	Department_id
1	7499	ALLEN	Narayan	1981-02-20	1600	CSE

```
--6) Drop the function based index on column Last_Name.

DROP INDEX emp_fun_index ON EMPLOYEE;

CREATE CLUSTERED INDEX emp_clust on employes(First_Name);

CREATE TABLE DEPARTMENT ( Dname VARCHAR(15) NOT NULL,
Dnumber INT NOT NULL, Mgr_ssn CHAR(9) NOT NULL, Mgr_start_date
DATE );
CREATE CLUSTERED INDEX DEPT_clust on DEPARTMENT(Dname);
```

Advanced Database Management Systems
Experiment-8
To understand the concepts of Sequence

Aryan Mohan
500092142
Batch- 2


```
create database LabExperiment8;

USE LabExperiment8;

CREATE SCHEMA EMPLOYEE ;

CREATE SEQUENCE EMPLOYEE.EMPID_SEQ START WITH 100
INCREMENT BY 1 ;

--Write a SQL command for finding the current and the next status of
EMPID_SEQ.
SELECT NEXT VALUE FOR EMPLOYEE.EMPID_SEQ;
```

Output:

	(No column name)
1	100

	(No column name)
1	101

	(No column name)
1	102

```
--Change the Cache value of the sequence EMPID_SEQ to 20 and maxvalue to
1000.
ALTER SEQUENCE EMPLOYEE.EMPID_SEQ RESTART WITH 500
INCREMENT BY 5 MINVALUE 50 MAXVALUE 1000 CYCLE CACHE 20;
SELECT NEXT VALUE FOR EMPLOYEE.EMPID_SEQ;
```

Output:

	(No column name)
1	500

```
--4) Insert values in employees table using sequences for employee_id column.
CREATE SCHEMA TEST;

CREATE TABLE TEST.EMPLOYEE( EMPID INTEGER PRIMARY KEY,
ENAME VARCHAR(30), JOB VARCHAR(20), MGR INTEGER,
```

```
HIREDATE DATE, SAL INTEGER, COMM INTEGER, DEPTNO INTEGER
);

CREATE SEQUENCE TEST.emp_id START WITH 1000 INCREMENT BY
1 ;

INSERT TEST.EMPLOYEE (EMPID, ENAME, JOB,MGR,
HIREDATE,SAL,COMM,DEPTNO) values (NEXT VALUE FOR
TEST.emp_id, 'ALLEN','SALESMAN', 7698,'20-FEB-81',1600,300,30);

SELECT * FROM TEST.EMPLOYEE;

INSERT TEST.EMPLOYEE (EMPID, ENAME, JOB,MGR,
HIREDATE,SAL,COMM,DEPTNO) values (NEXT VALUE FOR
TEST.emp_id, 'WARD', 'SALESMAN',7698, '22-FEB-81', 1250,500,30);

SELECT * FROM TEST.EMPLOYEE;
```

Output:

	EMPID	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	1000	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30

	EMPID	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
1	1000	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
2	1001	WARD	SALESMAN	7698	1981-02-22	1250	500	30

```
--Drop sequence EMPID_SEQ.
DROP SEQUENCE EMPLOYEE.EMPID_SEQ;

--Create a sequence called REVERSE to generate numbers in the descending
order from 10000 to 1000 with a decrement of 5.
CREATE SEQUENCE TEST.REVERSE START WITH 10000 INCREMENT
BY -5 MINVALUE 1000 MAXVALUE 10000 CYCLE CACHE 3;

SELECT NEXT VALUE FOR TEST.REVERSE;
```

Output:

	(No column name)
1	10000

Advance Database Management System Lab

Experiment- 9

To understand the concepts of PL/SQL programming

Aryan Mohan

500092142

Batch- 2

--1) Write a PL/SQL code to accept the value of A, B & C display which is greater.

```
BEGIN
DECLARE @A INTEGER;
SET @A =15;
DECLARE @B INTEGER;
SET @B = 65;
DECLARE @C INTEGER;
SET @C =25;
IF @A>@B AND @A>@C
PRINT 'GREATEST IS A';
ELSE IF @B>@C AND @B>@A
PRINT 'GREATEST IS B';
ELSE
PRINT 'GREATEST IS C';
END;
```

Output:

```
GREATEST IS B

Completion time: 2023-04-26T17:55:17.8833300+05:30
```

--2) Using PL/SQL Statements create a simple loop that display message “Welcome to PL/SQL Programming” 20 times

```
DECLARE @i integer;
set @i=1;
while @i<=20
BEGIN
PRINT 'Welcome to PL/SQL Programming';
set @i=@i+1;
END
```

Output:

```
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming

Completion time: 2023-04-26T17:56:10.8984882+05:30
```

```
-- 3) Write a PL/SQL code block to find the factorial of a number.
DECLARE @fact integer, @n integer;
set @fact=1;
set @n=6;
while @n > 0
begin
set @fact=@n*@fact
set @n=@n-1
end
print @fact
```

Output:

```
720

Completion time: 2023-04-26T17:56:30.8874657+05:30
```

```
--4) Write a PL/SQL program to generate Fibonacci series.
declare @f1 INTEGER=0, @f2 INTEGER=1,@f3 INTEGER,@i
INTEGER=3,@len INTEGER;
print 'First two number'
print @f1;
print @f2;
set @len=10;
print 'fibonacci series is';
while(@i<=@len)
begin
set @f3=@f1+@f2;
```

```
print @f3
set @f1=@f2;
set @f2=@f3;
set @i=@i+1;
end;
```

Output:

```
First two number
0
1
fibonacci series is
1
2
3
5
8
13
21
34

Completion time: 2023-04-26T17:56:47.6019567+05:30
```

```
--5) Write a PL/SQL code to fund the sum of first N numbers
declare @n integer, @i integer, @sum integer = 0;
set @i = 1;
set @n=10;
while (@i <= @n)
begin
set @sum=@sum+@i
set @i=@i+1
end
print 'sum of first N=10 numbers'
print @sum
```

Output:

```
sum of first N=10 numbers
55

Completion time: 2023-04-26T17:57:05.1809556+05:30
```

Advance Database Management Systems Lab

Experiment- 10

To understand the concepts of function and procedure in PL/SQL

Aryan Mohan

500092142

Batch- 2

```
create database LabExperiment10;

use LabExperiment10;

--1) Write a procedure to accept the value of A, B & C display which is greater.
Create procedure comp_no(@A INTEGER,@B INTEGER,@C INTEGER )
as begin
BEGIN
IF @A>@B AND @A>@C
PRINT 'GREATEST IS A';
ELSE IF @B>@C AND @B>@A
PRINT 'GREATEST IS B';
ELSE
PRINT 'GREATEST IS C';
END;
END

EXECUTE comp_no 100, 200, 50;
```

Output:

```
GREATEST IS B

Completion time: 2023-04-26T18:06:04.5349247+05:30
```

```
----2) Using procedure create a simple loop that display message “Welcome to PL/SQL Programming” 20 times
create procedure display_message(@message varchar(200) )
as begin
DECLARE @i integer;
set @i=1;
while @i<=20
```

```
BEGIN
PRINT @message
set @i=@i+1;
END
END

Execute display_message 'Welcome to PL/SQL Programming';
```

Output:

```
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
Welcome to PL/SQL Programming
```

```
--3) Write procedure to find the factorial of a number.
Create procedure fact(@no int)
as begin
Declare @i int = 1,@fact_no int=1
while (@i<=@no)
Begin
Set @fact_no = @fact_no * @i
Set @i += 1
End
Select @fact_no
End

Execute fact 5;


--4) Write a procedure to generate Fibonacci series.
create procedure Fibonacci(@fibno int)
as begin
```

```
declare @f1 INTEGER=0, @f2 INTEGER=1,@f3 INTEGER,@i
INTEGER=3,@len INTEGER;
print 'First two number'
print @f1;
print @f2;
print 'fibonacci series is';
while(@i<=@fibno)
begin
set @f3=@f1+@f2;
print @f3
set @f1=@f2;
set @f2=@f3;
set @i=@i+1;
end;
END;

execute Fibonacci 5;
```

Output:

```
First two number
0
1
fibonacci series is
1
2
3
First two number
0
1
fibonacci series is
1
2
3
First two number
0
1
fibonacci series is
1
2
3
First two number
```

```
--5) Write a procedure to find the sum of first N numbers
create procedure sum_number(@n integer)
as BEGIN
declare @i integer, @sum integer = 0;
set @i = 1;
while (@i <= @n)
begin
set @sum=@sum+@i
```

```
set @i=@i+1
end
print 'sum of first @n numbers'
print @sum
END

EXECUTE sum_number 5;
```

Output:

```
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
sum of first @n numbers
15
```

Advance Database Management System Lab

Experiment- 11

To understand the concepts of implicit and explicit cursor.

Aryan Mohan

500092142

Batch- 2

```
--1. Using implicit cursor update the salary by an
increase of 10% for all the records in EMPLOYEES table,
and finally display how many records have been updated.
If no records exist display the message “No Change”.
Create database LabExperiment11;

USE LabExperiment11;

CREATE TABLE EMPLOYEE( EMPID INTEGER PRIMARY KEY, ENAME
VARCHAR(30), JOB VARCHAR(20), MGR INTEGER, HIREDATE DATE,
SALARY INTEGER, COMM INTEGER, DEPTNO INTEGER );

insert into EMPLOYEE values(7499, 'ALLEN','SALESMAN',
7698,'20-FEB-81',1600,300,30);
insert into EMPLOYEE values(7521, 'WARD', 'SALESMAN',7698,
'22-FEB-81', 1250,500,30);
insert into EMPLOYEE values(7566, 'JONES','MANAGER',
7839,'02-APR-81',2975,0,20);
insert into EMPLOYEE values(7654, 'MARTIN',
'SALESMAN',7698, '28-SEP-81',1250,1400,30);
insert into EMPLOYEE values(7698, 'BLAKE',
'MANAGER',7839,'01-MAY-81',2850,0,30);
insert into EMPLOYEE values(7782, 'CLARK',
'MANAGER',7839,'09-JUN-81',2450,0,10);
insert into EMPLOYEE values(7788, 'SCOTT',
'ANALYST',7566,'09-DEC-82',3000,0,20);
insert into EMPLOYEE values(7839, 'KING',
'PRESIDENT',7599,'17-NOV-81',5000,0,10);
insert into EMPLOYEE values(7844,'TURNER', 'SALESMAN',
7698,'08-SEP-81',1500,0,30);
```

```
CREATE TABLE EMPLOYEE_AUDIT_SAL ( EMPID INTEGER, ENAME
VARCHAR(30), JOB VARCHAR(20), HIREDATE DATE, SALARY
INTEGER, DEPTNO INTEGER );

DECLARE @emp_id integer, @emp_name VARCHAR(50), @emp_job
varchar(20),@emp_date date, @emp_salary integer,@emp_dept
integer, @row integer;
DECLARE UPDATE_EM22 CURSOR FOR SELECT EMPID, ENAME, JOB,
HIREDATE, SALARY, DEPTNO FROM EMPLOYEE
OPEN UPDATE_EM22 FETCH NEXT FROM UPDATE_EM22 INTO @emp_id,
@emp_name, @emp_job, @emp_date, @emp_salary,@emp_dept
--set @emp_salary=@emp_salary+@emp_salary*0.1
WHILE @@FETCH_STATUS = 0
BEGIN
--SELECT @emp_id AS EMPID, @emp_name AS ENAME, @emp_job AS
JOB, @emp_date AS JOINING_DATE, @emp_salary AS
SALARY,@emp_dept AS DEPT
SET @emp_salary=@emp_salary+@emp_salary*0.1
insert into
EMPLOYEE_AUDIT_SAL(EMPID,ENAME,JOB,HIREDATE,SALARY,DEPTNO)
VALUES (@emp_id, @emp_name, @emp_job, @emp_date,
@emp_salary,@emp_dept)
FETCH NEXT FROM UPDATE_EM22 INTO @emp_id, @emp_name,
@emp_job, @emp_date, @emp_salary,@emp_dept
--SET @emp_salary=@emp_salary+@emp_salary*0.1
--insert into
EMPLOYEE_AUDIT(EMPID,ENAME,JOB,HIREDATE,SALARY,DEPTNO)
VALUES (@emp_id, @emp_name, @emp_job, @emp_date,
@emp_salary,@emp_dept)
END
SET @row = (SELECT COUNT(*) FROM EMPLOYEE_AUDIT_SAL)
if @row=0
print 'No Change'
else
select * from EMPLOYEE_AUDIT_SAL
order by EMPID
CLOSE UPDATE_EM22
DEALLOCATE UPDATE_EM22

SELECT * FROM EMPLOYEE
ORDER BY EMPID
```

Output:

	EMPID	ENAME	JOB	MGR	HIREDATE	SALARY	COMM	DEPTNO
1	7499	ALLEN	SALESMAN	7698	1981-02-20	1600	300	30
2	7521	WARD	SALESMAN	7698	1981-02-22	1250	500	30
3	7566	JONES	MANAGER	7839	1981-04-02	2975	0	20
4	7654	MARTIN	SALESMAN	7698	1981-09-28	1250	1400	30
5	7698	BLAKE	MANAGER	7839	1981-05-01	2850	0	30
6	7782	CLARK	MANAGER	7839	1981-06-09	2450	0	10
7	7788	SCOTT	ANALYST	7566	1982-12-09	3000	0	20
8	7839	KING	PRESIDENT	7599	1981-11-17	5000	0	10
9	7844	TURNER	SALESMAN	7698	1981-09-08	1500	0	30

```
--2. Using explicit cursor fetch the employee name,
employee_id and salary of all the records from EMPLOYEES
table.
DECLARE @employee_id integer, @emp_name VARCHAR(50),
@emp_salary integer;
DECLARE FETCH_CURSOR CURSOR FOR
SELECT EMPID, ENAME, SALARY FROM EMPLOYEE
OPEN FETCH_CURSOR
FETCH NEXT FROM FETCH_CURSOR INTO @employee_id, @emp_name,
@emp_salary

WHILE @@FETCH_STATUS = 0
BEGIN
select @employee_id, @emp_name, @emp_salary
FETCH NEXT FROM FETCH_CURSOR INTO @employee_id, @emp_name,
@emp_salary
END
CLOSE FETCH_CURSOR
DEALLOCATE FETCH_CURSOR
```

Output:

	(No column name)	(No column name)	(No column name)
1	7499	ALLEN	1600
1	7521	WARD	1250
1	7566	JONES	2975
1	7654	MARTIN	1250
1	7698	BLAKE	2850
1	7782	CLARK	2450
1	7788	SCOTT	3000
1	7839	KING	5000
1	7844	TURNER	1500

```
--3. Using explicit cursor Insert the records from
EMPLOYEES table for the columns employee_id, Last_Name and
salary for those records whose salary exceeds 2500 into a
new table TEMP_EMP
CREATE TABLE EMPLOYEE_TEMP ( EMPID INTEGER, ENAME
VARCHAR(30), SALARY INTEGER );
DECLARE @emp_id integer, @emp_name VARCHAR(50),
@emp_salary integer;
DECLARE INSERT_CURSOR CURSOR FOR
SELECT EMPID, ENAME, SALARY FROM EMPLOYEE
OPEN INSERT_CURSOR
FETCH NEXT FROM INSERT_CURSOR INTO @emp_id, @emp_name,
@emp_salary

WHILE @@FETCH_STATUS = 0
BEGIN
--select @emp_id, @emp_name, @emp_salary
IF @emp_salary>2500
insert into EMPLOYEE_TEMP(EMPID,ENAME,SALARY) VALUES
(@emp_id, @emp_name,@emp_salary)
FETCH NEXT FROM INSERT_CURSOR INTO @emp_id, @emp_name,
@emp_salary
END
CLOSE INSERT_CURSOR
DEALLOCATE INSERT_CURSOR

SELECT * FROM EMPLOYEE_TEMP
ORDER BY EMPID
```

Advance Database Management System Lab

Experiment- 12

To understand the concepts of Trigger.

Aryan Mohan

500092142

Batch- 2

```
CREATE TABLE CUSTOMER(ID INTEGER PRIMARY KEY,NAME
VARCHAR(20), AGE INTEGER, ADDRESS VARCHAR(30), SALARY
DECIMAL(20,0));
INSERT INTO CUSTOMER VALUES(1, 'Ramesh', 32, 'Ahmedabad',
2000.00);
INSERT INTO CUSTOMER VALUES(2, 'Khilan', 25,
'Delhi',1500.00);
INSERT INTO CUSTOMER VALUES(3, 'Kaushik', 23, 'Kota',
2000.00);
INSERT INTO CUSTOMER VALUES(4, 'Chaitali', 25,
'Mumbai',6500.00);
INSERT INTO CUSTOMER VALUES(5, 'Hardik', 27,
'Bhopal',8500.00);
INSERT INTO CUSTOMER VALUES(6, 'Komal', 22, 'MP',4500.00);
CREATE TABLE CUSTOMER_AUDIT(ID INTEGER, NAME VARCHAR(20),
AGE INTEGER, ADDRESS VARCHAR(30), SALARY DECIMAL(20,0),
Audit_Action varchar(100), Audit_Timestamp datetime );
CREATE TRIGGER After_Insert ON [dbo].[CUSTOMER] FOR INSERT
AS declare @EMPID INTEGER; declare @EMPNAME VARCHAR(20);
declare @EMPAGE INTEGER; declare @EMPADDRESS VARCHAR(30);
declare @EMPSALARY DECIMAL(20,0); declare @audit_Action
varchar(100);
select @EMPID=i.ID from inserted i;
select @EMPNAME=i.NAME from inserted i;
select @EMPAGE=i.AGE from inserted i;
select @EMPADDRESS=i.ADDRESS from inserted i;
select @EMPSALARY=i.SALARY from inserted i;
set @audit_action='Inserted Record -- After Insert
Trigger.';

insert into CUSTOMER_AUDIT (ID,NAME,AGE,ADDRESS,SALARY,
Audit_Action,Audit_Timestamp)
```

```
values(@EMPID,@EMPNAME,@EMPAGE,@EMPADDRESS,
@EMPSALARY,@audit_action,getdate());
PRINT 'AFTER INSERT trigger fired.'
INSERT INTO CUSTOMER VALUES(105,'Chetan', 25,
'jaipur',6500.00);
SELECT * FROM CUSTOMER_AUDIT;
CREATE TRIGGER After_Delete ON [dbo].[Employee] AFTER
DELETE AS declare @EMPID INTEGER; declare @EMPNAME
VARCHAR(20); declare @EMPAGE INTEGER; declare @EMPADDRESS
VARCHAR(30); declare @EMPSALARY DECIMAL(20,0); declare
@audit_Action varchar(100);
select @EMPID=i.ID from deleted d;
select @EMPNAME=i.NAME from deleted d;
select @EMPAGE=i.AGE from deleted d;
select @EMPADDRESS=i.ADDRESS from deleted d;
select @EMPSALARY=i.SALARY from deleted d;
set @audit_action='deleted Record -- After delete
Trigger.';
insert into CUSTOMER_AUDIT (ID,NAME,AGE,ADDRESS,SALARY,
Audit_Action,Audit_Timestamp)
values(@EMPID,@EMPNAME,@EMPAGE,@EMPADDRESS,
@EMPSALARY,@audit_action,getdate());
PRINT 'AFTER DELETE trigger fired.'
DELETE FROM CUSTOMER WHERE ID=6
select * from CUSTOMER
select * from CUSTOMER
```

Output:

	ID	NAME	AGE	ADDRESS	SALARY
1	1	Ramesh	32	Ahmedabad	2000
2	2	Khilan	25	Delhi	1500
3	3	Kaushik	23	Kota	2000
4	4	Chaitali	25	Mumbai	6500
5	5	Hardik	27	Bhopal	8500

	ID	NAME	AGE	ADDRESS	SALARY
1	1	Ramesh	32	Ahmedabad	2000
2	2	Khilan	25	Delhi	1500
3	3	Kaushik	23	Kota	2000
4	4	Chaitali	25	Mumbai	6500
5	5	Hardik	27	Bhopal	8500