**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;

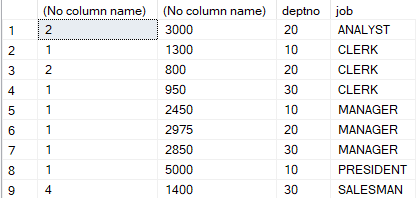
Output:



--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;

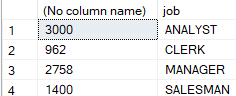
Output:



--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;

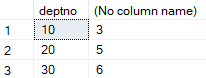
Output:



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

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

Output:



--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;

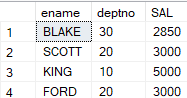
Output:



--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:



--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:

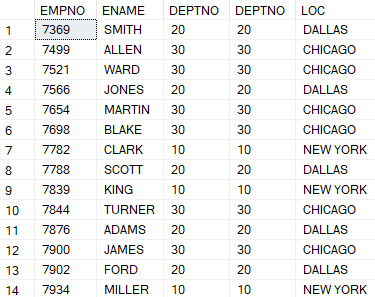


---JOIN OPERATIONS

---Equi join

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

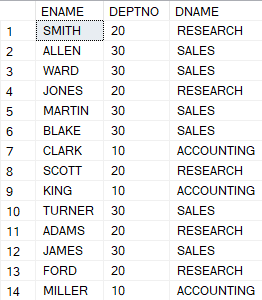
Output:



----Outer Joins

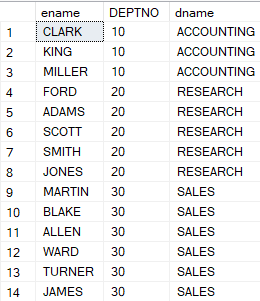
SELECT e.ENAME, e.DEPTNO, d.DNAME FROM EMPLOYEE e, DEPARTMENT d WHERE e.DEPTNO = d.DEPTNO;

Output:



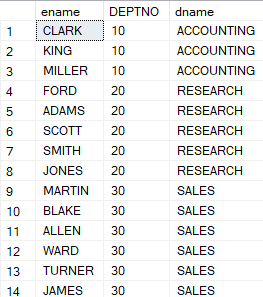
SELECT e.ename, d.DEPTNO, d.dname FROM EMPLOYEE e, DEPARTMENT d WHERE e.deptno =d.deptno ORDER BY e.deptno;

Output:



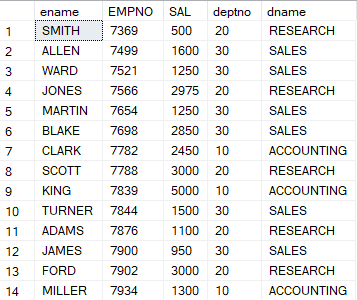
SELECT e.ename, d.DEPTNO, d.dname FROM EMPLOYEE e INNER JOIN DEPARTMENT AS d ON e.deptno =d.deptno ORDER BY e.deptno;

Output:



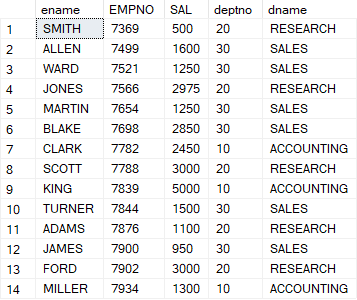
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE e INNER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



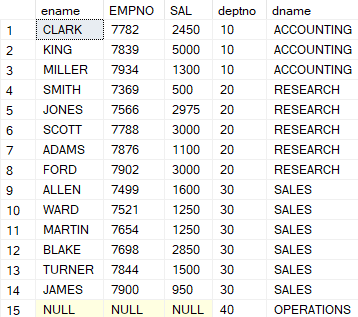
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE e LEFT OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



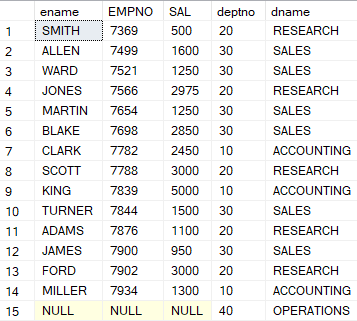
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE e RIGHT OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



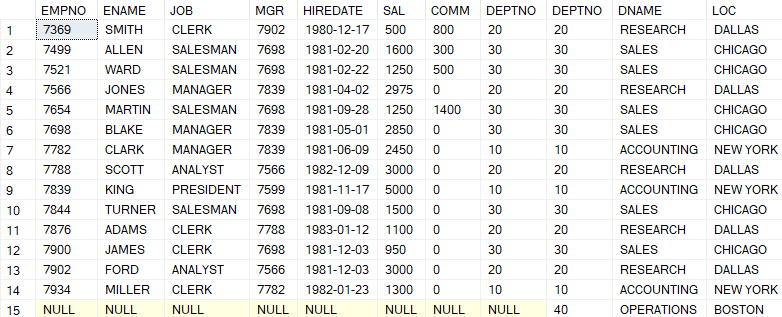
SELECT e.ename, e.EMPNO, e.SAL, d.deptno, d.dname FROM EMPLOYEE e FULL OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



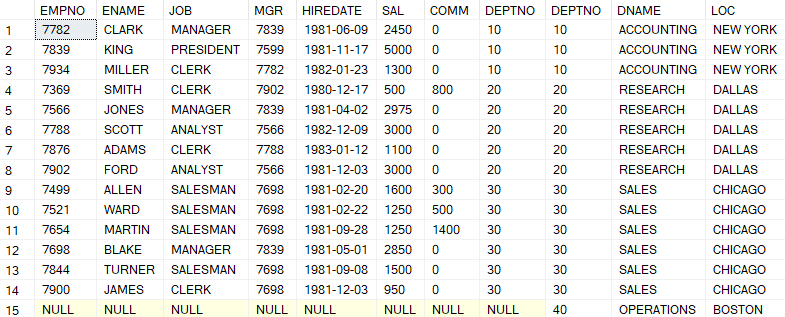
SELECT \* FROM EMPLOYEE e FULL OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



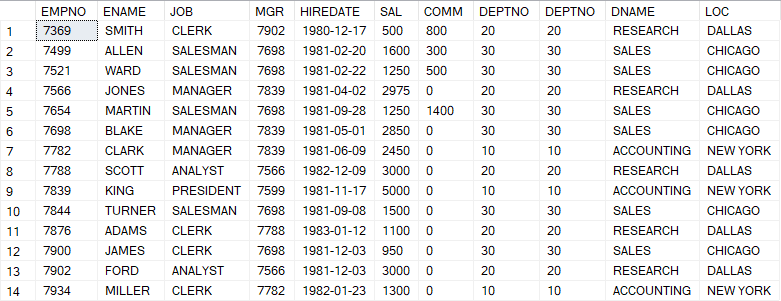
SELECT \* FROM EMPLOYEE e RIGHT OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



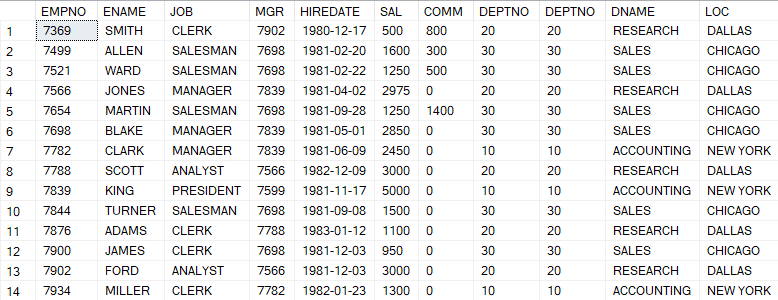
SELECT \* FROM EMPLOYEE e LEFT OUTER JOIN DEPARTMENT AS d ON e.deptno =d.deptno;

Output:



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

Output:



**Advance Database Management Systems Lab**

**Experiment- 6**

**To understand the concepts of Views**

**Aryan Mohan**

**500092142**

**Batch- 2**

CREATE DATABASE LabExperiment6;

USE LabExperiment6;

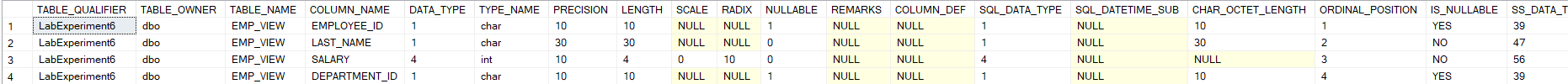
CREATE TABLE EMPLOYEES( Employee\_id VARCHAR(10) NOT NULL PRIMARY KEY, First\_Name VARCHAR(30) NOT NULL, Last\_Name VARCHAR(30) NOT NULL, DOB Date, salary DECIMAL(25,0) NOT NULL, Department\_id VARCHAR(10) );

--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');

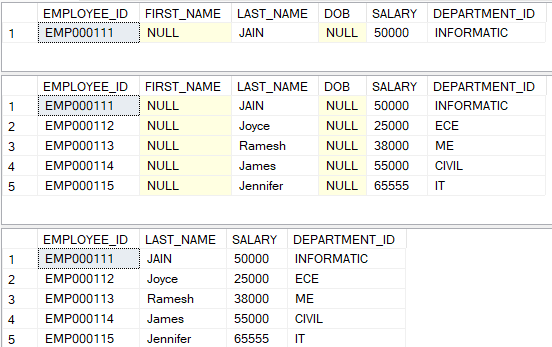
INSERT INTO emp\_view VALUES ('EMP000114', 'James', 55000, 'CIVIL');

INSERT INTO emp\_view VALUES('EMP000115','Jennifer',65555, 'IT');

SELECT \* FROM EMPLOYEES;

SELECT \* FROM emp\_view;

Output:



--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';

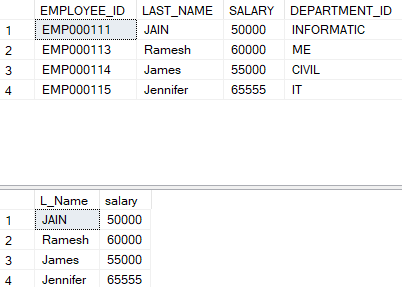
SELECT \* FROM emp\_view;

CREATE VIEW ADBMS(L\_Name, salary) AS SELECT Last\_Name, salary FROM EMPLOYEES;

SELECT \* FROM ADBMS;

DROP VIEW ADBMS;

Output:



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

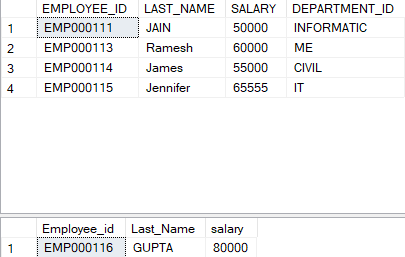
SELECT \* FROM emp\_view;

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:



**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:



--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:



--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:







--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:



--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 ;

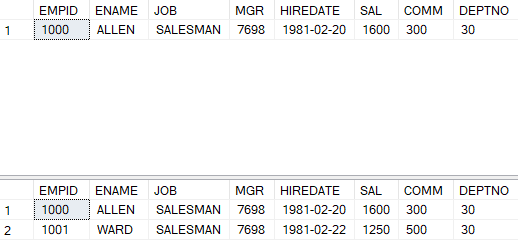
INSERT TEST.EMPLOYEE (EMPID, ENAME, JOB,MGR, HIREDATE,SAL,COMM,DEPTNO) values (NEXT VALUE FOR TEST.emp\_id, 'ALLEN','SALESMAN', 7698,'20-FEB-81',1600,300,30);

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:



--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:



**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:



--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

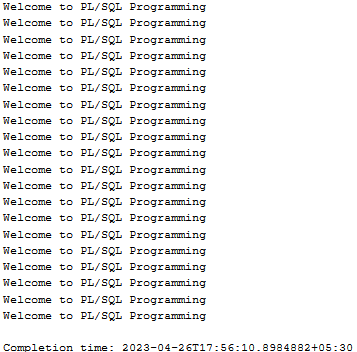
BEGIN

PRINT 'Welcome to PL/SQL Programming';

set @i=@i+1;

END

Output:



-- 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:



--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

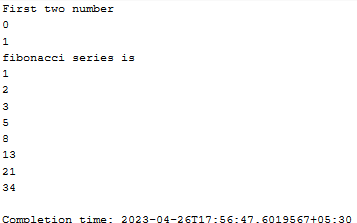
set @f1=@f2;

set @f2=@f3;

set @i=@i+1;

end;

Output:



--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

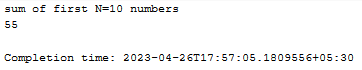
set @i=@i+1

end

print 'sum of first N=10 numbers'

print @sum

Output:



**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:



----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

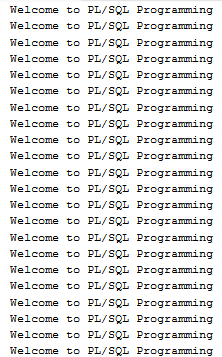
set @i=@i+1;

END

END

Execute display\_message 'Welcome to PL/SQL Programming';

Output:



--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;

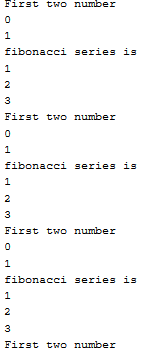
set @i=@i+1;

end;

END;

execute Fibonacci 5;

Output:



--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

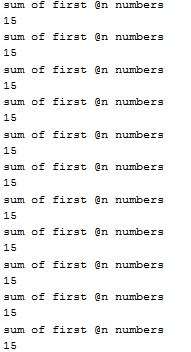
print 'sum of first @n numbers'

print @sum

END

EXECUTE sum\_number 5;

Output:



**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

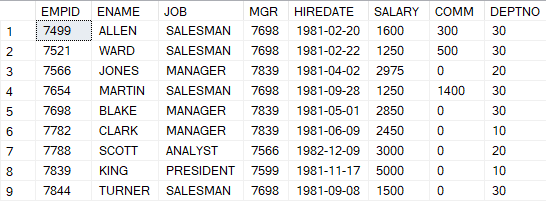
CLOSE UPDATE\_EM22

DEALLOCATE UPDATE\_EM22

SELECT \* FROM EMPLOYEE

ORDER BY EMPID

Output:



--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

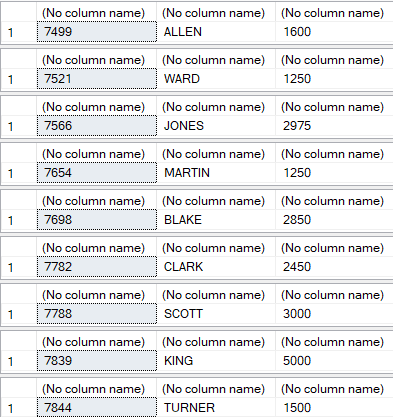
FETCH NEXT FROM FETCH\_CURSOR INTO @employee\_id, @emp\_name, @emp\_salary

END

CLOSE FETCH\_CURSOR

DEALLOCATE FETCH\_CURSOR

Output:



--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());

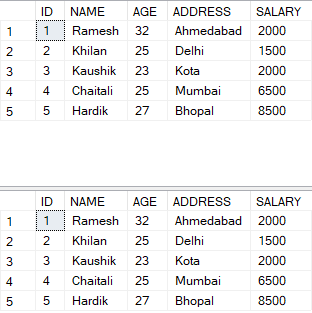
PRINT 'AFTER DELETE trigger fired.'

DELETE FROM CUSTOMER WHERE ID=6

select \* from CUSTOMER

select \* from CUSTOMER

Output:

****