Lab Assignment–1

By: Asavri Kaur(102203940)

**1. Create table Student (Rno, Name, DOB, Gender, Class, College, City, Marks)**

CREATE TABLE student(

Rno number,

Name varchar(10),

DOB date,

Gender varchar(10),

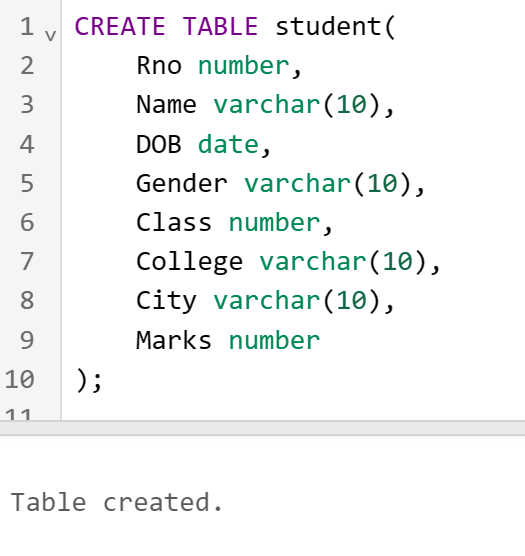
Class number,

College varchar(10),

City varchar(10),

Marks number

);



**2. Insert 5 records in student table**

INSERT INTO student(Rno,Name,DOB,Gender,Class,College,City, Marks)

VALUES(1,'tina','12 January 1990','female',1,'thapar','patiala',97);

INSERT INTO student(Rno,Name,DOB,Gender,Class,College,City, Marks)

VALUES(2,'seema','10 May 1990','female',2,'thapar','amritsar',81);

INSERT INTO student(Rno,Name,DOB,Gender,Class,College,City, Marks)

VALUES(9,'meena','7 June 1990','female',3,'thapar','patiala',72);

INSERT INTO student(Rno,Name,DOB,Gender,Class,College,City, Marks)

VALUES(4,'beena','8 July 1990','female',4,'thapar','patiala',12);

INSERT INTO student(Rno,Name,DOB,Gender,Class,College,City, Marks)

VALUES(5,'reema','30 April 1990','female',5,'thapar','patiala',20);

A screenshot of a computer

Description automatically generated

**3. Display the information of all the students**

SELECT \* FROM student;

A table of information

Description automatically generated with medium confidence

**4. Display the detail structure of student table**

DESC student;

A table of names and numbers

Description automatically generated

**5. Display Rno, Name and Class information of ‘Patiala’ students.**

SELECT Rno, Name, Class FROM student WHERE City='patiala';

A screenshot of a cell phone

Description automatically generated

**6. Display information on ascending order of marks**

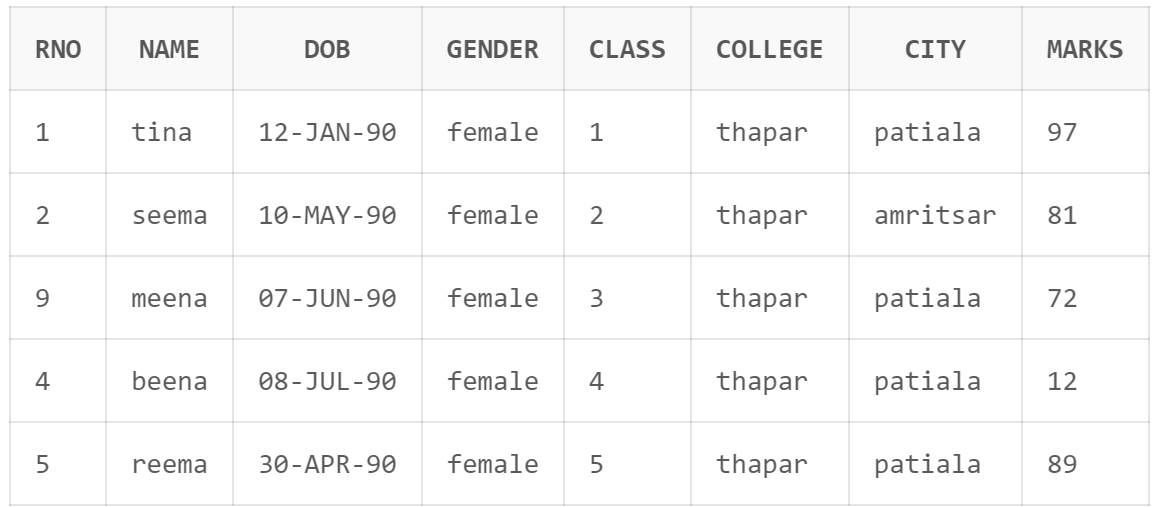
SELECT \* FROM student ORDER BY Marks;

A table of information

Description automatically generated with medium confidence

**7. Change the marks of Rno 5 to 89.**

UPDATE student SET Marks=89 WHERE Rno=5;



**8. Change the name and city of Rno 9.**

UPDATE student SET Name='leena', City='amritsar' WHERE Rno=9;

A white grid with black text

Description automatically generated

**9. Delete the information of ‘Amritsar’ city records**

DELETE FROM student WHERE city = 'amritsar';

A screen shot of a table

Description automatically generated

**10. Delete the records of student where marks<30.**

DELETE FROM student WHERE marks<30;

A grid of black and white text

Description automatically generated

**Lab Assignment–2**

**By: Asavri Kaur 102203940**

**Based on Emp table**

**Columns are EmpNo, Ename, Job, Salary, Commission, DeptNO**

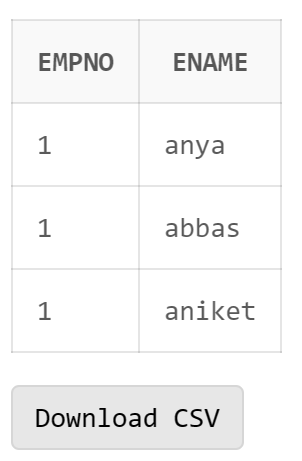
**Insert 5 records by storing Null value in some records for commission column.**

A table with numbers and words

Description automatically generated

**Q1) Get employee no and employee name who works in dept no 10?**

select EmpNo, Ename from EmpTable where DeptNO=10;



**Q2) Display the employee names of those clerks whose salary > 2000?**

select Ename from EmpTable where Job='clerk' and salary>2000;

A screenshot of a computer

Description automatically generated

**Q3) Display name and job of Salesperson & Clerks?**

select Ename, Job from EmpTable where Job='clerk' or Job='salesperson';

A screenshot of a computer

Description automatically generated

**Q4) Display all details of employees whose salary between 2000 and 3000?**

select \* from EmpTable where salary between 2000 and 3000;

A table with text on it

Description automatically generated

**Q5) Display all details of employees whose dept no is 10, 20, or 30?**

select \* from EmpTable where DeptNo in (10,20,30)

A table with numbers and words

Description automatically generated

**Q6) Display name of those employees whose commission is NULL?**

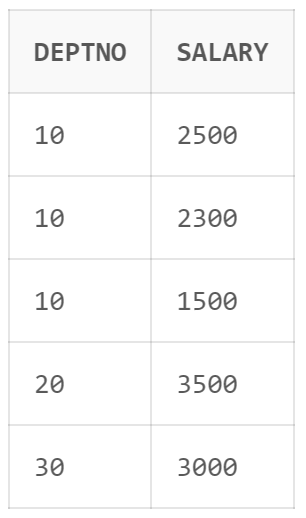
select Ename from Emptable where Commission is NULL;

A screenshot of a phone

Description automatically generated

**Q7) Display dept no & salary in ascending order of dept no and with in each dept no salary should be in descending order?**

select DeptNO,Salary from EmpTable order by DeptNO, Salary Desc;



**Q8) Display name of employees having two ‘a’ or ‘A’ chars in the name?**

select Ename from EmpTable where Ename like '%aa%' or Ename like '%AA%';

A white rectangular sign with black text

Description automatically generated

**Q9) Display the name of the employees whose second char is‘b’ or ‘B’?**

select Ename from EmpTable where Ename like '%\_b%' or Ename like '%\_B%';



**Q10) Display the name of the employees whose first or last char is ‘a’ or ‘A’?**

select Ename from EmpTable where Ename like 'a%' or Ename like 'A%' or Ename like '%a' or Ename like '%A';

A screenshot of a phone

Description automatically generated

**Q11) Display maximum, minimum, average salary of deptno 10 employees.**

select max(Salary), min(Salary), avg(Salary) from EmpTable where DeptNO=10;

A close-up of a number

Description automatically generated

**Q12) Display total number of employees working in deptno 20**

select count (\*) from EmpTable where DeptNO=20;

A close-up of a white card

Description automatically generated

**Q13) Display total salary paid to clerks**

select sum(Salary) from EmpTable where Job='clerk';

A close-up of a number

Description automatically generated

**Q14) Display system date**

select sysdate from dual;

A close-up of a white sign

Description automatically generated

**Q15) Display the result of (12\*12)/13**

select 12\*12/13 from dual;

A close up of numbers

Description automatically generated

**Q16) Display info of ‘raj’ irrespective to the case in which the data is stored.**

select \* from EmpTable where Ename='raj';

A white squares with black text

Description automatically generated

**Lab Assignment–3**

**Q1) Use the following functions**

1. **chr (n):**

SELECT chr(67) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **cancat(char1,char2):**

SELECT concat('asavri','kaur') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **instr(string,char):**

SELECT INSTR('I am Asavri','vr') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **length(n):**

SELECT length('asavri') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **lpad(char1 ,n [,char2]):**

SELECT lpad('asavri',10,'\*') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **ltrim(string [,char(s)]):**

SELECT ltrim('asavri','a') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **rpad(char1 ,n [,char2]):**

SELECT rpad('asavri',10,'\*') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **rtrim(string [,char(s)]):**

SELECT rtrim('asavri','i') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **replace(char ,search\_string , replacement\_string):**

SELECT replace('hardware','hard','soft') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **substr(string ,position ,substring length):**

SELECT substr('hardware',2,3) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **initcap(char):**

SELECT initcap('asavri kaur') FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **lower(string):**

SELECT lower('ASAVRI') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **upper(string):**

SELECT upper('asavri') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **translate(char ,from string ,to string):**

SELECT translate('abc','ab','ed') FROM DUAL;

A screenshot of a chat

Description automatically generated

1. **abs(n):**

SELECT abs(-43) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **ceil(n):**

SELECT ceil(4.3) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **cos(n):**

SELECT cos(2) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **exp(n):**

SELECT exp(3) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **floor(n):**

SELECT floor(4.3) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **mod(m ,n):**

SELECT mod(13,3) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **power(x ,y):**

SELECT power(2,3) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **round(x [,y]):**

SELECT round(23.1254,3) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **sign(n):**

SELECT sign(-12) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **sqrt(n);**

SELECT sqrt(12) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **trunc(x ,n):**

SELECT trunc(602.1122,2) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **sysdate:**

SELECT sysdate FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **add\_months(d ,n):**

SELECT add\_months(DATE '2024-02-02',1) FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **last\_day():**

SELECT last\_day(DATE '2024-02-02') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **months\_between(date1 ,date2):**

SELECT months\_between(DATE '2024-02-02', DATE '2024-04-05') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **next\_day(date ,char):**

SELECT next\_day(DATE '2024-02-02', 'THURSDAY') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **greatest(expr):**

SELECT GREATEST(2,4,3,1) FROM DUAL;

A screenshot of a phone

Description automatically generated

1. **least(expr):**

SELECT LEAST(2,4,3,1) FROM DUAL;

A screenshot of a computer

Description automatically generated

**Q2) Display current time in hour : min : sec format**

SELECT TO\_CHAR(sysdate, '"Hours: "hh", Minutes: "mi" Seconds: "ss') FROM DUAL;

A screenshot of a phone

Description automatically generated

**Q3) Display salary + commission of emp table**

update emp set emp.commission=emp.commission + emp.salary;

****

**Q4) Store any date value in hiredate column of table ?**

UPDATE emp

SET hiredate='22 February 1990'

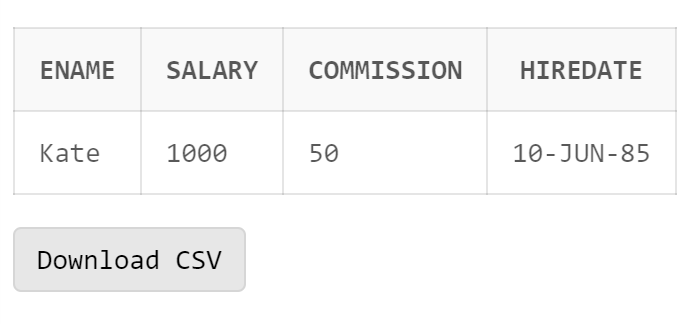
WHERE ename='Bob';

A screenshot of a computer

Description automatically generated

**Q5) Display name of employee(s) who join the company in 1985?**

SELECT \* FROM emp WHERE hiredate >='01 JANUARY 1985' AND hiredate < '01 JANUARY 1986';



**Q6) Display name of the employee(s) who join the company this year?**

SELECT \* FROM emp WHERE hiredate >='01 JANUARY 2024' AND hiredate < '01 JANUARY 2025';

A screenshot of a computer

Description automatically generated

**Assignment-4**

**(Working on Date Functions)**

**Write queries to:**

1. **Display the system date**

SELECT sysdate FROM DUAL;



1. **Display current day**

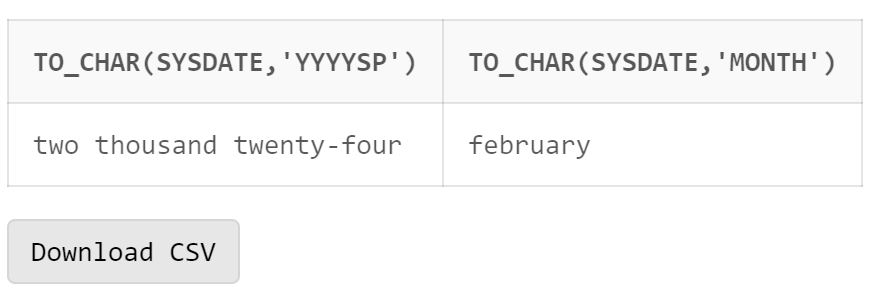
SELECT to\_char(sysdate,'day') FROM DUAL;

A screenshot of a chat

Description automatically generated

1. **Display current month and spell out year**

SELECT to\_char(sysdate, 'yyyysp'), to\_char(sysdate,'month') FROM DUAL;



1. **Display spell out current date**

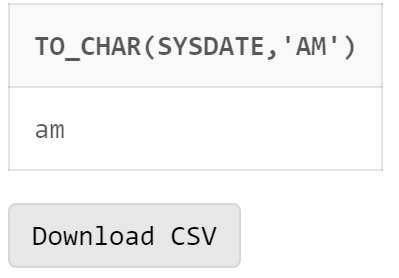
SELECT to\_char(sysdate, 'ddsp') FROM DUAL;

**A screenshot of a computer

Description automatically generated**

1. **Check whether it is AM or PM right now**

SELECT to\_char(sysdate,'am') FROM DUAL;



1. **Display the date of next Friday**

SELECT next\_day(sysdate,'Friday') FROM DUAL

**A screenshot of a computer

Description automatically generated**

1. **Round the system date on month**

SELECT round(sysdate,'month') FROM DUAL**;**

A screenshot of a computer

Description automatically generated

1. **Truncate the system date on month**

SELECT trunc(sysdate,'month') FROM DUAL;

**A screenshot of a computer

Description automatically generated**

1. **Round the system date on year**

SELECT round(sysdate,'year') FROM DUAL;

**A screenshot of a computer

Description automatically generated**

1. **Truncate the system date on year**

SELECT trunc(sysdate,'year') FROM DUAL;

A screenshot of a computer

Description automatically generated

1. **Find the day after three days**

SELECT to\_char(sysdate+3,'day') FROM DUAL;

**A screenshot of a chat

Description automatically generated**

**Queries Based on EMP table**

1. **Create an EMP table (Empno number, Name Varchar2(20),** **date\_of\_joining date)**

CREATE TABLE EMP2(

Empno NUMBER,

Name VARCHAR(20),

date\_of\_joining DATE

);

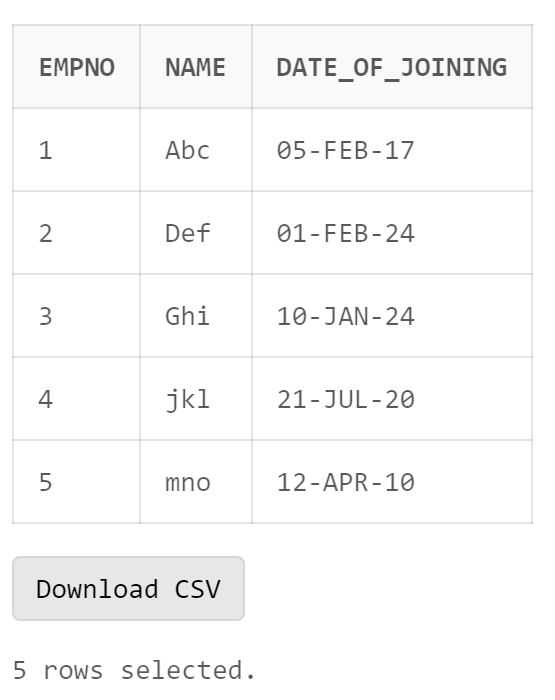
insert into EMP2 values (1, 'Abc', '05-Feb-2017');

insert into EMP2 values (2, 'Def', '01-Feb-2024');

insert into EMP2 values (3, 'Ghi', '10-Jan-2024');

insert into EMP2 values (4, 'jkl', '21-Jul-2020');

insert into EMP2 values (5, 'mno', '12-April-2010');

****

1. **Display day of date of joining column**

SELECT to\_char(date\_of\_joining,'day') FROM EMP2;

A screenshot of a calendar

Description automatically generated

1. **Display those employees who join the company on Monday**

SELECT Name FROM EMP2 WHERE to\_char(date\_of\_joining,'fmday')='monday';

**A screenshot of a phone

Description automatically generated**

1. **Display those employees who join the company this month**

SELECT Name FROM EMP2 WHERE to\_char(date\_of\_joining,'fmmon')='feb';

**A screenshot of a phone

Description automatically generated**

1. **Display those employees who join the company in last 30 days**

SELECT Name FROM EMP2 WHERE date\_of\_joining between sysdate-30 and sysdate;

****

**Queries based on Train Arrival and Departure**

1. **Create a table Train having four columns (TrainNo Number (6) primary key, date of departure, time of departure, time of arrival)**

Create table Train(TrainNo Number(6) primary key,

departure date,

time\_arrival timestamp,

time\_departure timestamp

);

1. **Insert five records into the table Train.**

insert into train values(123,'12-jan-2024','12-Jan-2024 09:45:32pm','12-Jan-2024 11:34:10pm');

insert into train values(132,'09-jan-2024','09-Feb-2024 09:45:32am','09-Feb-2024 09:54:10am');

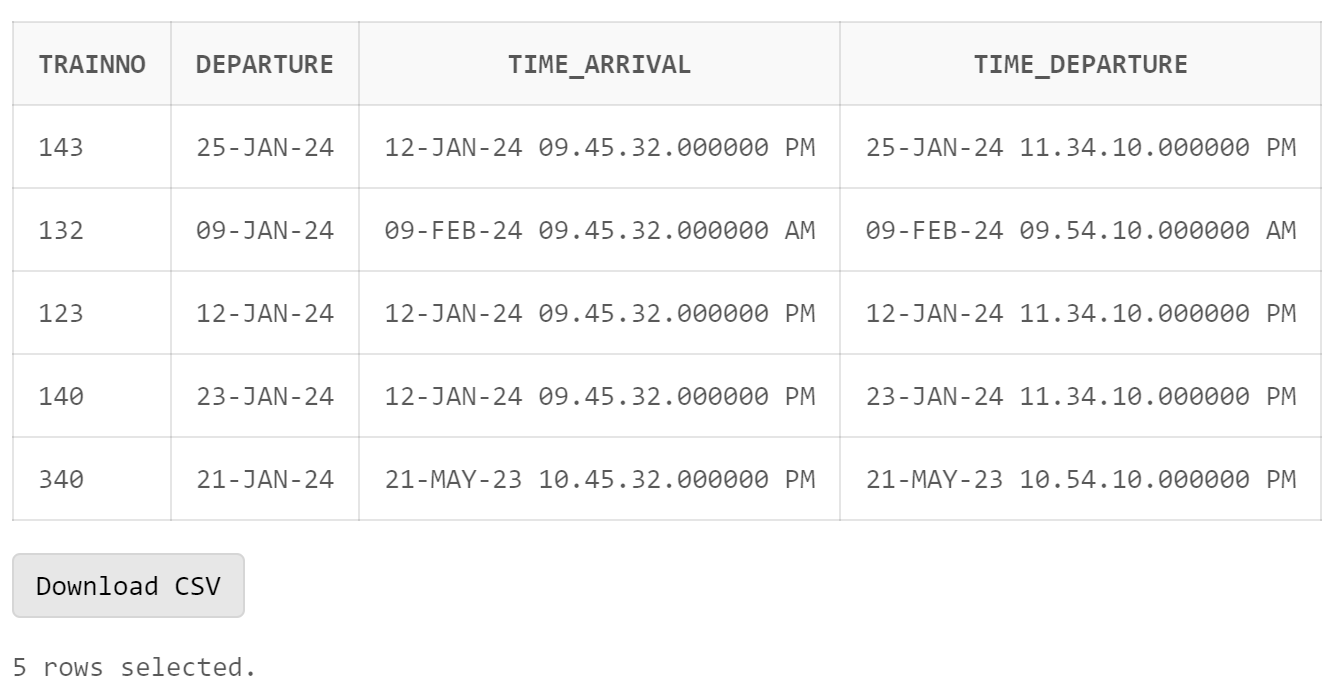
insert into train values(340,'21-jan-2024','21-May-2023 10:45:32pm','21-May-2023 10:54:10pm');

insert into train values(140,'23-jan-2024','12-Jan-2024 09:45:32pm','23-Jan-2024 11:34:10pm');

insert into train values(143,'25-jan-2024','12-Jan-2024 09:45:32pm','25-Jan-2024 11:34:10pm');

1. **Display all the three records**

select \* from train;

****

1. **Display the time values inserted in the columns**

select to\_char(time\_arrival,'hh:mm:ss'),to\_char(time\_departure,'hh:mm:ss') from train;

**A screenshot of a computer

Description automatically generated**

1. **Display those trains which arrived on PM**

select \* from Train where to\_char(time\_arrival, 'pm')='pm';

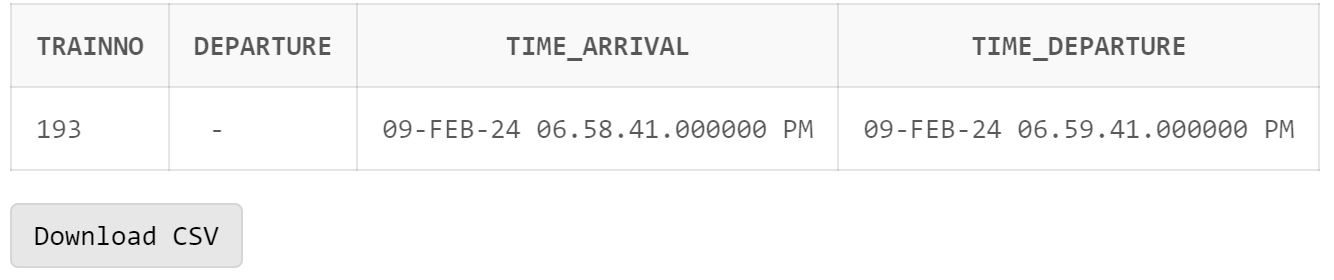
**A screen shot of a schedule

Description automatically generated**

1. **Display train number who are going to depart in next one hour.**

insert into Train values (193,'','09-Feb-2024 06:58:41pm','09-Feb-2024 06:59:41pm');

select \* from Train where time\_departure between sysdate and sysdate+1/24;

****

**Assignment-5**

**(Constraints in SQL)**

**1. Create table dept which has the following attributes (department table)**

**(deptno, dept\_name) where deptno is primary key, dept\_name in (Acc, comp, elect)**

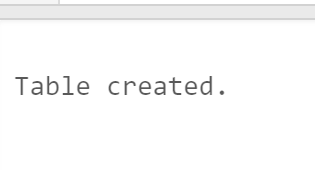
create table dept(

deptno number,

dept\_name varchar(20) constraint dept\_name\_check check(dept\_name in ('acc','comp','elect')),

constraint d\_pk primary key (deptno)

);

****

**2. Create table emp which has the following attributes (employee table)**

**(empno, emp\_name, job, sal, deptno, mgr\_no) where empno is primary key, emp\_name is unique,**

**job in (Prof, AP, and Lect), sal is NOT NULL, deptno is foreign key, mgr\_no is a self-referential**

**foreign key.**

create table emp(

empno number,

emp\_name varchar(20),

job varchar(20),

sal number not null,

deptno number,

mgr\_no number,

constraint empno\_pk primary key(empno),

constraint emp\_name\_unique unique(emp\_name),

constraint check\_job check(job in ('prof','ap','lect')),

constraint deptno\_fk foreign key (deptno) references dept(deptno),

constraint mgr\_no\_fk foreign key (mgr\_no) references emp(empno)

);

**3. Create table S which has the following attributes (Salesperson table)**

**(sno, sname, city) where sno is primary key**

create table s(

sno number,

sname varchar(20),

city varchar(20),

constraint s\_pk primary key(sno)

);

**4. Create table P which has the following attributes (Part table)**

**(pno, pname, color) where pno is primary key**

create table p(

pno number,

pname varchar(20),

color varchar(20),

constraint p\_pk primary key(pno)

);

**5. Create table J which has the following attributes (ProJect table)**

**(jno, jname, city) where jno is primary key**

create table j(

jno number,

jname varchar(20),

city varchar(20),

constraint j\_pk primary key(jno)

);

**6. Create table SPJ which has the following attributes**

**(sno, pno, jno, qty) where combination of (sno, pno, jno) is a composite primary key. Also,**

**sno, pno, jno are foreign keys.**

create table spj(

sno number,

pno number,

jno number,

qty number,

constraint spj\_pk primary key(sno,pno,jno),

constraint spj\_sfk foreign key (sno) references s(sno),

constraint spj\_pfk foreign key (pno) references p(pno),

constraint spj\_jfk foreign key (jno) references j(jno)

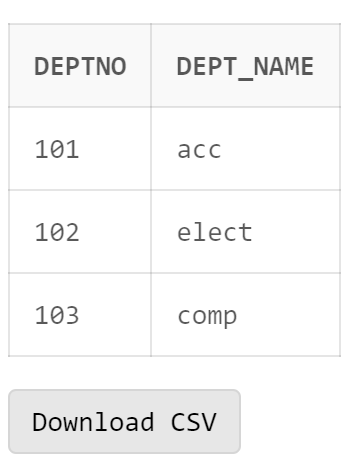
);

**7. Insert at least 5 appropriate records in the above tables.**

insert into dept values(101,'acc');

insert into dept values(102,'elect');

insert into dept values(103,'comp');



insert into emp values(111,'Ramesh','prof',1000,101,null);

insert into emp values(114,'Suresh','ap',2000,101,111);

insert into emp values(112,'Ravi','lect',3000,102,null);

insert into emp values(113,'Harish','ap',4000,102,112);

insert into emp values(100,'Manish','prof',5000,103,111);

A table of numbers and letters

Description automatically generated

insert into s values(1,'A','delhi');

insert into s values(2,'B','patiala');

insert into s values(3,'C','gurgaon');

insert into s values(4,'D','amritsar');

insert into s values(5,'E','ludhiana');

A screenshot of a white grid

Description automatically generated

insert into p values(11,'AA','orange');

insert into p values(12,'BB','blue');

insert into p values(13,'CC','red');

insert into p values(14,'DD','yellow');

insert into p values(15,'EE','green');

A screenshot of a cell phone

Description automatically generated

insert into j values(101, 'AAA', 'delhi');

insert into j values(102,'BBB','patiala');

insert into j values(103,'CCC','gurgaon');

insert into j values(104,'DDD','amritsar');

insert into j values(105,'EEE','ludhiana');

A screenshot of a phone

Description automatically generated

insert into spj values(1,11,101,1);

insert into spj values(2,12,102,2);

insert into spj values(3,13,103,3);

insert into spj values(4,14,104,4);

insert into spj values(5,15,105,5);

select \* from spj;

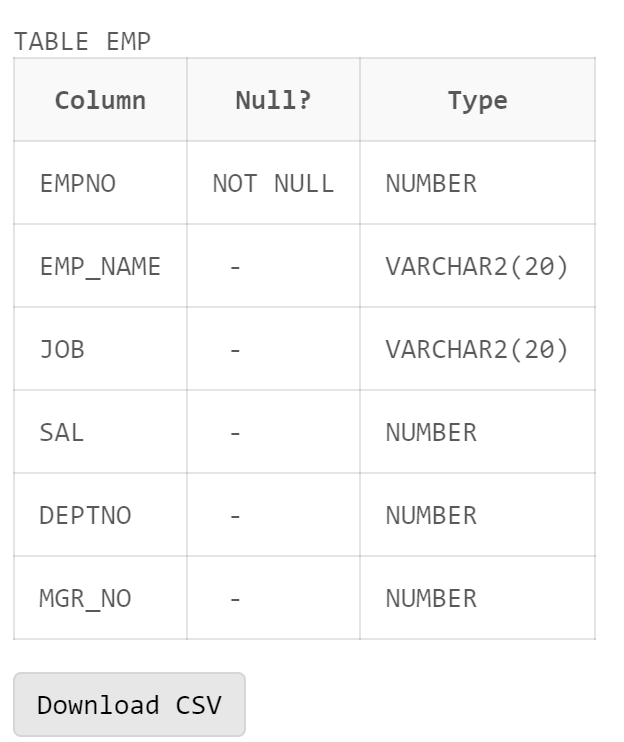
A screenshot of a computer

Description automatically generated

**8. Drop the NOT NULL constraints from EMP table.**

alter table emp modify(sal null);

--here, empno cannot be made null as it is the primary key

****

**9. Check all the constraints name and their type of EMP table.**

SELECT \* FROM user\_constraints WHERE table\_name='EMP';



**10. Drop the unique constraint on EMP\_NAME of EMP table.**

alter table emp drop constraint emp\_name\_unique;

A close-up of a check

Description automatically generated

**11. Drop the Foreign Key constraint on DEPTNO**

alter table emp drop constraint deptno\_fk;

A screenshot of a computer

Description automatically generated

**12. Add Foreign Key constraint on DEPTNO as a table label constraint.**

alter table emp add constraint deptno\_fk foreign key (deptno) references dept(deptno);

A white grid with black text

Description automatically generated

**13. Drop the Check constraint from DEPT table.**

alter table dept drop constraint dept\_name\_check;

A screenshot of a computer

Description automatically generated

**14. Add COMM column in EMP table (default value 0).**

alter table emp add (comm number);

alter table emp modify comm number default 0;

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**15. Drop Default constraint from EMP.**

alter table emp modify comm default null;

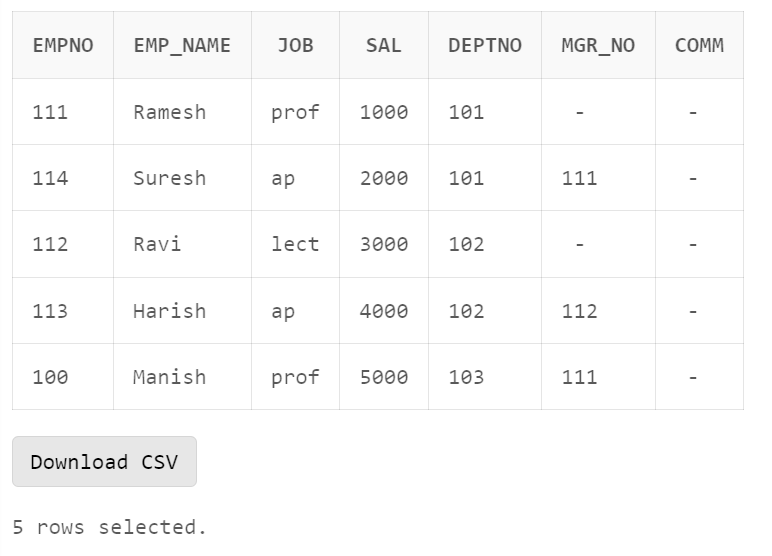
**A close-up of a word

Description automatically generated**

**16. Create duplicate copy of EMP table.**

create table new\_emp\_table as select \* from emp;

select \* from new\_emp\_table;



**17. Copy the structure of DEPT table to a new table with different column names.**

create table n\_dept\_table as select

deptno as A,

dept\_name as B

from dept

where 1=1;

desc n\_dept\_table;

select \* from n\_dept\_table;

A screenshot of a computer

Description automatically generated

**18. Copy the structure of DEPT table to a new table with different column names without any**

records copied from DEPT.

create table new\_dept\_table as select

deptno as A,

dept\_name as B

from dept

where 1=2;

desc new\_dept\_table;

select \* from new\_dept\_table;

A screenshot of a computer

Description automatically generated

A close-up of a white background

Description automatically generated

**19. Change the name and job of the employee whose EMPNO =100.**

update emp set emp\_name='Leena',job='lect' where empno=100;

select \* from emp where empno=100;

A screenshot of a computer

Description automatically generated

**20. Delete the record of employee who belong to computer department.**

delete from emp where deptno =

(select deptno from dept where dept\_name='comp');

select \* from emp;

A screenshot of a computer

Description automatically generated

**21. Delete deptno 101 from Dept table and set NULL to the corresponding deptno in EMP table.**

alter table emp drop constraint deptno\_fk;

alter table emp add constraint deptno\_fk foreign key (deptno) references dept(deptno) on delete set null;

delete from dept where deptno=101;

select \* from emp;

A screenshot of a computer

Description automatically generated

**22. Delete deptno 102 from Dept table and its corresponding record from EMP table.**

delete from dept where deptno=102;

select \* from emp;

select \* from dept;

A screenshot of a computer

Description automatically generated

**23. Delete the empno 111 who is the manager of the employee whose empno is 114.**

alter table emp drop constraint mgr\_no\_fk;

alter table emp add constraint mgr\_no\_fk foreign key (mgr\_no) references emp(empno) on delete set null;

delete from emp where empno=111;

select \* from emp;

A screenshot of a computer

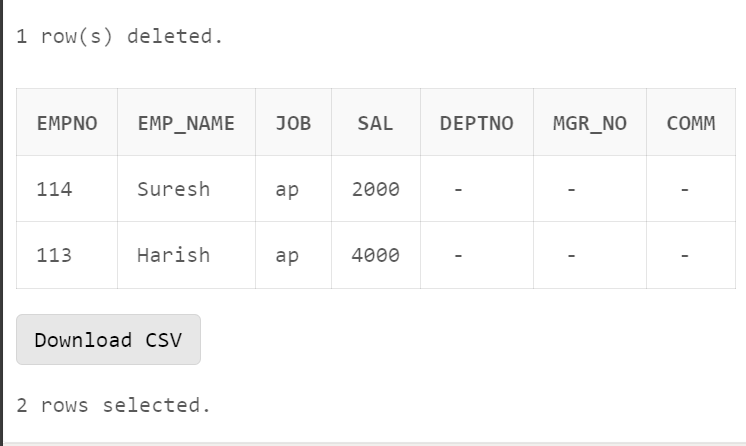
Description automatically generated

**24. Delete the record of ‘Ravi’ whose empno is 112 and set the mgr\_no to NULL for all the employees for whom Ravi is the manager.**

update emp set mgr\_no = null where mgr\_no=112;

delete from emp where empno=112;

select \* from emp;



**25. Drop the duplicate table of EMP.**

drop table new\_emp\_table;

A screenshot of a computer

Description automatically generated

**LAB ASSIGNMENT-6**

**Group By and Joins in SQL (Part-I)**

**TABLE DEPT (DEPTNO, DEPTNAME)**

**TABLE EMP (EMPNO, EMPNAME, DEPTNO, JOB, SALARY): DEPTNO is a foreign**

**key w.r.t to DEPT Table.**

**Answer the following SQL Queries**

1. **Insert 10 appropriate records in each table as per the SQL queries mentioned below.**

insert into emp values(101,'A','salesman',1000,10);

insert into emp values(102,'B','salesman',1000,20);

insert into emp values(103,'C','clerk',1000,20);

insert into emp values(104,'D','clerk',2000,20);

insert into emp values(105,'E','salesman',3000,20);

insert into emp values(106,'F','engineer',4000,20);

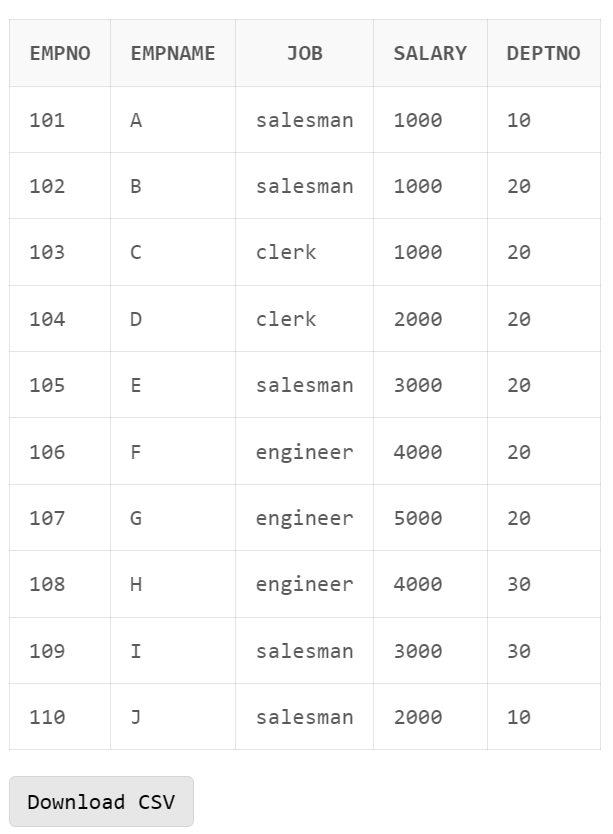
insert into emp values(107,'G','engineer',5000,20);

insert into emp values(108,'H','engineer',4000,30);

insert into emp values(109,'I','salesman',3000,30);

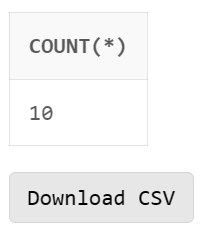
insert into emp values(110,'J','salesman',2000,10);

select \* from emp;



1. **List the total number of employees?**

select count(\*) from emp;



1. **List the total no of departments?**

select count(\*) from dept;

A screenshot of a phone number

Description automatically generated

1. **Display the employee details of each department for which they are working.**

select \* from emp order by deptno;

A table of numbers and letters

Description automatically generated

1. **List the total, maximum, & minimum salary for deptno 30?**

select min(salary), max(salary), sum(salary)

from emp

where deptno=30;

A screenshot of a phone

Description automatically generated

1. **Display the name of the employee getting maximum salary?**

select empname

from emp

where salary = (

select max(salary)

from emp

);

A screenshot of a computer

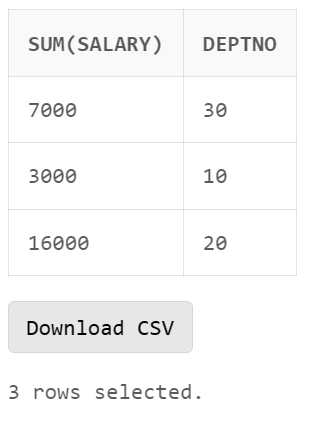
Description automatically generated

1. **Display the total salary for each department?**

select sum(salary), deptno

from emp

group by deptno;



1. **Display the total salary for each job.**

select sum(salary), job

from emp

group by job;



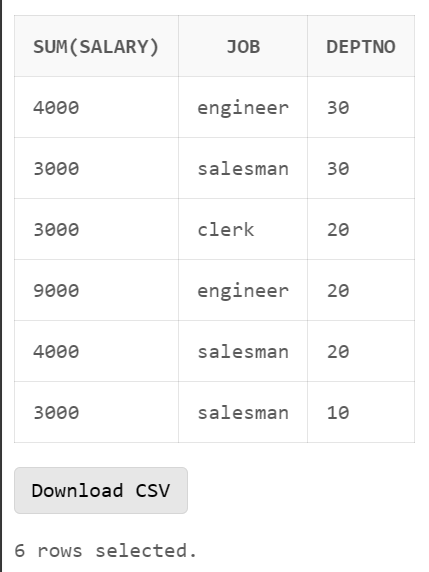
1. **Display the total salary for each job within each department.**

select sum(salary),job,deptno

from emp

group by deptno,job

order by deptno desc;



1. **Display the average salary for each job in deptno 20.**

select avg(salary), job

from emp

where deptno=20 AND job in (

select distinct job

from emp

where deptno=20

)

group by job;

A screenshot of a phone

Description automatically generated

1. **Display the total salary for each job excluding the ‘manager’ and ‘salesman’ job.**

select job, sum(salary) from emp

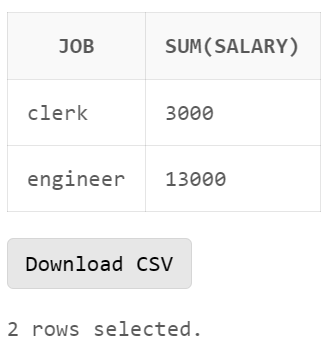
where job not in (

'salesman',

'manager'

)

group by job;



**12. Display the average salary for each job in deptno 20, but only display those jobs where**

**average salary is greater than 2000 & display the output in descending order of salary?**

select avg(salary), job

from emp

where deptno=20

group by job

having avg(salary)>2000

order by avg(salary) desc;



**13. Display the total no of employees for each department excluding the deptno 10 & display**

**only those departments where more than five employees work. Display the output in**

**descending order of total no of employees?**

select count(\*),deptno

from emp

where deptno <> 10

group by deptno

having count(\*)>5

order by count(\*) desc;

A screenshot of a computer

Description automatically generated

**14. Display the total no of employees for each department excluding the ‘comp’ dept & display**

**only those departments where more than five employees work. Display the output in**

**descending order of total no of employees?**

select count(\*),deptno

from emp

where deptno <> (

select deptno

from dept

where deptname = 'comp'

)

group by deptno

having count(\*)>5

order by count(\*) desc;

A screenshot of a computer

Description automatically generated

**15. Display total number of emp working in each job in each dept.**

select count(\*),deptno, job

from emp

group by deptno,job

order by deptno desc;

A screenshot of a computer

Description automatically generated

**16. List all the department name and their employees name even if for a department there is no**

**employees (A newly created department).**

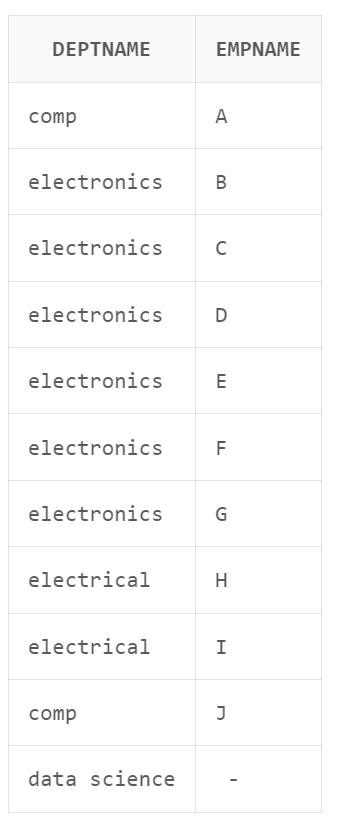
insert into dept values(40,'data science');

select dept.deptname, emp.empname

from dept

left join emp

on dept.deptno = emp.deptno;



**17. List all the employees name and their department name even if for an employee there is no**

**department assigned (A newly joined employee).**

insert into emp values(112,'H','engineer',1000,null);

select dept.deptname, emp.empname

from emp

left join dept

on dept.deptno = emp.deptno;

A screenshot of a computer

Description automatically generated

**18. List all the departments name and their employees name irrespective whether for a**

**department any employees are there are not, and irrespective whether for an employee**

**there is any department assigned or not.**

select dept.deptname, emp.empname

from emp

full outer join dept

on dept.deptno = emp.deptno;

A screenshot of a computer

Description automatically generated

**Assignment 7**

**(Subquery in SQL)**

**Table 1**: SalesPeople

**Snum is Primary key**

**Sname is Unique constraint**

**Snum Sname City Comm**

1001 Peel London 0.12

1002 Serres Sanjose 0.13

1004 Motika London 0.11

1007 Rifkin Barcelona 0.15

1003 Axelrod New York 0.10

**Table 2: Customers**

**Cnum is Primary Key**

**City has not null constraint.**

**Snum is foreign key constraint refers Snum column of SalesPeople table.**

**Cnum Cname City Snum**

2001 Hoffman London 1001

2002 Giovanni Rome 1003

2003 Liu Sanjose 1002

2004 Grass Berlin 1002

2006 Clemens London 1001

2008 Cisneros Sanjose 1007

2007 Pereira Rome 1004

**Table 3: Orders**

**Onum is Primary key**

**Cnum is foreign key refers to Cnum column of Customers table.**

**Snum is foreign key refers Snum column of SalesPeople table.**

**Onum Amt Odate Cnum Snum**

3001 18.69 3-10-1990 2008 1007

3003 767.19 3-10-1990 2001 1001

3002 1900.10 3-10-1990 2007 1004

3005 5160.45 3-10-1990 2003 1002

3006 1098.16 3-10-1990 2008 1007

3009 1713.23 4-10-1990 2002 1003

3007 75.75 4-10-1990 2004 1002

3008 4273.00 5-10-1990 2006 1001

3010 1309.95 6-10-1990 2004 1002

3011 9891.88 6-10-1990 2006 1001

1. **Create the above three tables and insert those records.**

CREATE TABLE SALESPEOPLE (SNUM NUMBER,

SNAME VARCHAR2(10),

CITY VARCHAR2(15),

COMM NUMBER,

CONSTRAINT S\_UNIQUE UNIQUE (SNAME),

CONSTRAINT S\_PK PRIMARY KEY(SNUM)

);

INSERT INTO SALESPEOPLE VALUES(1001,'Peel','London',0.12);

INSERT INTO SALESPEOPLE VALUES(1002,'Serres','Sanjose',0.13);

INSERT INTO SALESPEOPLE VALUES(1004,'Motika','London',0.11);

INSERT INTO SALESPEOPLE VALUES(1007,'Rititka','Barcelona',0.15);

INSERT INTO SALESPEOPLE VALUES(1003,'Axelrod','New York',0.10);

SELECT \* FROM SALESPEOPLE;

A screenshot of a computer

Description automatically generated

CREATE TABLE CUSTOMERS (CNUM NUMBER,

CNAME VARCHAR2(10),

CITY VARCHAR2(10) NOT NULL,

SNUM NUMBER,

CONSTRAINT C\_PK PRIMARY KEY (CNUM),

CONSTRAINT C\_FK FOREIGN KEY (SNUM) REFERENCES SALESPEOPLE(SNUM)

);

INSERT INTO CUSTOMERS VALUES(2001,'Hoffaman','London',1001);

INSERT INTO CUSTOMERS VALUES(2002,'Giovanni','Rome',1003);

INSERT INTO CUSTOMERS VALUES(2003,'Liu','Sanjose',1002);

INSERT INTO CUSTOMERS VALUES(2004,'Grass','Berlin',1002);

INSERT INTO CUSTOMERS VALUES(2006,'Clemens','London',1001);

INSERT INTO CUSTOMERS VALUES(2008,'Cineros','Sanjose',1007);

INSERT INTO CUSTOMERS VALUES(2007,'Pereira','Rome',1004);

SELECT \* FROM CUSTOMERS;

A screenshot of a cell phone

Description automatically generated

CREATE TABLE ORDERS (ONUM NUMBER,

AMT NUMBER,

ODATE DATE,

CNUM NUMBER ,

SNUM NUMBER,

CONSTRAINT O\_PK PRIMARY KEY (ONUM),

CONSTRAINT OC\_FK FOREIGN KEY (CNUM) REFERENCES CUSTOMERS(CNUM),

CONSTRAINT OS\_FK FOREIGN KEY (SNUM) REFERENCES SALESPEOPLE(SNUM)

);

INSERT INTO ORDERS VALUES (3001,18.69,'3 DECEMBER 1990',2008,1007);

INSERT INTO ORDERS VALUES (3003,767.19,'3 DECEMBER 1990',2001,1001);

INSERT INTO ORDERS VALUES (3002,1900.10,'3 DECEMBER 1990',2007,1004);

INSERT INTO ORDERS VALUES (3005,5160.45,'3 DECEMBER 1990',2003,1002);

INSERT INTO ORDERS VALUES (3006,1098.16,'3 DECEMBER 1990',2008,1007);

INSERT INTO ORDERS VALUES (3009,1713.23,'4 DECEMBER 1990',2002,1002);

INSERT INTO ORDERS VALUES (3007,75.75,'4 DECEMBER 1990',2004,1002);

INSERT INTO ORDERS VALUES (3008,4273.00,'5 DECEMBER 1990',2006,1001);

INSERT INTO ORDERS VALUES (3010,1309.95,'6 DECEMBER 1990',2004,1002);

INSERT INTO ORDERS VALUES (3011,9891.88,'6 DECEMBER 1990',2006,1001);

SELECT \* FROM ORDERS;

A table of numbers and numbers

Description automatically generated

1. **Display all the salesperson whose all orders worth is more than Rs. 2000.**

SELECT \*

FROM SALESPEOPLE S

WHERE S.SNUM IN (

SELECT O.SNUM

FROM ORDERS O

GROUP BY SNUM

HAVING SUM(AMT)>2000

);

A screenshot of a computer

Description automatically generated

1. **Display the salesperson id handling at least two customers.**

SELECT SNUM

FROM CUSTOMERS

GROUP BY SNUM

HAVING COUNT(\*)>1;

A screenshot of a computer

Description automatically generated

1. **Display the salesperson names handling at least two customers.**

SELECT S.SNAME

FROM SALESPEOPLE S

WHERE SNUM IN (

SELECT C.SNUM

FROM CUSTOMERS C

GROUP BY SNUM

HAVING COUNT(\*)>1

);

A screenshot of a computer

Description automatically generated

1. **Display the number of Salespeople belonging to London and Paris.**

SELECT CITY,COUNT(\*)

FROM SALESPEOPLE

GROUP BY CITY

HAVING CITY IN ('London','Paris');

A screenshot of a computer

Description automatically generated

1. **Display the customer names corresponding to salesperson belonging to** **London and Paris.**

SELECT C.CNAME, C.SNUM

FROM CUSTOMERS C

WHERE C.SNUM IN (

SELECT SNUM

FROM SALESPEOPLE

WHERE CITY IN ('London','Paris')

);

**A screenshot of a phone

Description automatically generated**

1. **Display the customer names whose order amount is worth 1200.**

SELECT C.CNAME

FROM CUSTOMERS C

WHERE C.CNUM IN (

SELECT CNUM

FROM ORDERS

GROUP BY CNUM

HAVING SUM(AMT)=1200

);

A close up of a text

Description automatically generated

--no customers had order values that equalled to exactly 1200

**8) Display the customer names who belong to the same city as their**

**salesperson.**

SELECT C.CNAME

FROM CUSTOMERS C

WHERE C.CITY = (

SELECT S.CITY

FROM SALESPEOPLE S

WHERE S.SNUM = C.SNUM

);

A screenshot of a phone

Description automatically generated

**9) Display the salesperson names having commission more than 10 percent**

**and who handled more than two customers.**

SELECT S.SNAME

FROM SALESPEOPLE S

WHERE S.COMM>0.10

AND

S.SNUM IN (

SELECT C.SNUM

FROM CUSTOMERS C

GROUP BY C.SNUM

HAVING COUNT(\*)>2

); --THERE ARE NO SALESPEOPLE WHO HANDLE MORE THAN 2 CUSTOMERS

A close-up of a text

Description automatically generated

**10) Display the customer’s name who orders exactly one order and his**

**salesperson name.**

SELECT C.CNAME, S.SNAME

FROM CUSTOMERS C JOIN SALESPEOPLE S

ON C.SNUM = S.SNUM

WHERE C.CNUM IN (

SELECT O.CNUM

FROM ORDERS O

GROUP BY O.CNUM

HAVING COUNT(\*)=1

);

A screenshot of a phone

Description automatically generated

**11) Insert the details of a salesperson in the table who is a new joined and did not handle any customer. Write a query for the salesperson who has not handle any customer.**

INSERT INTO SALESPEOPLE VALUES(1008,'Aman','Delhi',0.3);

--inserting new joinee into table

SELECT \*

FROM SALESPEOPLE S

WHERE S.SNUM NOT IN (

SELECT DISTINCT C.SNUM

FROM CUSTOMERS C

);

A screenshot of a computer

Description automatically generated

**12) Write a query that counts the number of Salespeople registering orders**

**for each day.**

SELECT COUNT(DISTINCT O.SNUM), O.ODATE

FROM ORDERS O

GROUP BY O.ODATE;

A screenshot of a computer

Description automatically generated

**13) Find out the no. of Salesperson who belongs to same city and have same**

**commission percentage.**

INSERT INTO SALESPEOPLE VALUES(1009,'Raman','London',0.12);

INSERT INTO SALESPEOPLE VALUES(1010,'Manan','London',0.12);

INSERT INTO SALESPEOPLE VALUES(1011,'Ravi','London',0.11);

INSERT INTO SALESPEOPLE VALUES(1012,'Suman','New York',0.10);

SELECT COUNT(\*), S1.CITY, S1.COMM

FROM SALESPEOPLE S1

GROUP BY S1.CITY, S1.COMM;

A screenshot of a phone

Description automatically generated

**Lab Assignment 8- Join and Sub Queries**

**Create the following tables and insert some tuples in these tables shown below. Where sid is the**

**primary key for the Sailors table, bid is the primary key for the Boats table and sid and bid are**

**the foreign keys for the Reserves table referencing to the Sailors and Boats table, respectively.**

**Sailors(sid: integer, sname: string, rating: integer, age: real)**

**Boats(bid: integer, bname: string, color: string)**

**Reserves(sid: integer, bid: integer, day: date)**

**After inserting the records in these tables, the instances should look like as follows:**

A table with numbers and a few words

Description automatically generated with medium confidence

CREATE TABLE SAILORS(

SID NUMBER,

SNAME VARCHAR(20),

RATING NUMBER,

AGE DECIMAL(3,1),

CONSTRAINT S\_PK PRIMARY KEY (SID)

);

INSERT INTO SAILORS VALUES(22,'Dustin',7,45.0);

INSERT INTO SAILORS VALUES(29,'Brutus',1,33.0);

INSERT INTO SAILORS VALUES(31,'Lubber',8,55.5);

INSERT INTO SAILORS VALUES(32,'Andy',8,25.5);

INSERT INTO SAILORS VALUES(58,'Rusty',10,35.0);

INSERT INTO SAILORS VALUES(64,'Horatio',7,35.0);

INSERT INTO SAILORS VALUES(71,'Zorba',10,16.0);

INSERT INTO SAILORS VALUES(74,'Horatio',9,35.0);

INSERT INTO SAILORS VALUES(85,'Art',3,25.5);

INSERT INTO SAILORS VALUES(95,'Bob',3,63.5);

SELECT \* FROM SAILORS;

CREATE TABLE BOATS(

BID NUMBER,

BNAME VARCHAR(20),

COLOR VARCHAR(20),

CONSTRAINT B\_PK PRIMARY KEY(BID)

);

INSERT INTO BOATS VALUES(101,'Interlake','blue');

INSERT INTO BOATS VALUES(102,'Interlake','red');

INSERT INTO BOATS VALUES(103,'Clipper','green');

INSERT INTO BOATS VALUES(104,'Marine','red');

SELECT \* FROM BOATS;

CREATE TABLE RESERVES(

SID NUMBER,

BID NUMBER,

DAY DATE,

CONSTRAINT R\_SFK FOREIGN KEY (SID) REFERENCES SAILORS(SID),

CONSTRAINT R\_BPK FOREIGN KEY (BID) REFERENCES BOATS(BID),

CONSTRAINT R\_PK PRIMARY KEY (SID,BID)

);

INSERT INTO RESERVES VALUES(22,101,'10 OCTOBER 1998');

INSERT INTO RESERVES VALUES(22,102,'10 OCTOBER 1998');

INSERT INTO RESERVES VALUES(22,103,'10 AUGUST 1998');

INSERT INTO RESERVES VALUES(22,104,'10 JULY 1998');

INSERT INTO RESERVES VALUES(31,102,'11 OCTOBER 1998');

INSERT INTO RESERVES VALUES(31,103,'11 JUNE 1998');

INSERT INTO RESERVES VALUES(31,104,'11 DECEMBER 1998');

INSERT INTO RESERVES VALUES(64,101,'9 MAY 1998');

INSERT INTO RESERVES VALUES(64,102,'9 AUGUST 1998');

INSERT INTO RESERVES VALUES(74,103,'9 AUGUST 1998');

SELECT \* FROM RESERVES;

**Write SQL command using JOINs and/or Sub-queries for the following:**

**Q1. Find the names of sailors who have reserved a red or a green boat.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.SID IN(

SELECT DISTINCT R.SID

FROM RESERVES R

WHERE R.BID IN (

SELECT B.BID

FROM BOATS B

WHERE B.COLOR IN ('red','green')

)

);

A screenshot of a phone

Description automatically generated

**Q2. Find the names of sailors who have reserved both a red and a green boat.**

SELECT SNAME

FROM SAILORS

WHERE SID IN (

SELECT DISTINCT R.SID

FROM RESERVES R

WHERE EXISTS (

SELECT SID

FROM RESERVES

WHERE BID IN (

SELECT BID FROM BOATS

WHERE COLOR='red'

)

AND SID=R.SID

) AND EXISTS(

SELECT SID

FROM RESERVES

WHERE BID=(

SELECT BID FROM BOATS

WHERE COLOR='green'

)

AND SID = R.SID

)

);

A screenshot of a phone

Description automatically generated

**Q3. Find the names of sailors who have reserved boat 103**

SELECT S.SNAME

FROM SAILORS S

WHERE S.SID IN (

SELECT R.SID

FROM RESERVES R

WHERE R.BID=103

);

**A screenshot of a phone

Description automatically generated**

**Q4. Find the names of sailors who have reserved a red boat.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.SID IN (

SELECT R.SID

FROM RESERVES R

WHERE R.BID IN (

SELECT B.BID

FROM BOATS B

WHERE COLOR = 'red'

)

);

A screenshot of a phone

Description automatically generated

**Q5. Find the names of sailors who have NOT reserved a red boat.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.SID NOT IN (

SELECT R.SID

FROM RESERVES R

WHERE R.BID IN (

SELECT B.BID

FROM BOATS B

WHERE COLOR = 'red'

)

);

A screenshot of a phone

Description automatically generated

**Q6. Find the names of sailors who have reserved at least one boat.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.SID IN (

SELECT DISTINCT R.SID

FROMRESERVES R

);

A screenshot of a phone

Description automatically generated

**Q7. Find sailors whose rating is better than some sailor called Horatio.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.RATING > ANY (

SELECT RATING

FROM SAILORS

WHERE SNAME='Horatio'

);

**A screenshot of a phone

Description automatically generated**

**Q8. Find sailors whose rating is better than every sailor called Horatio.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.RATING > ALL (

SELECT RATING

FROM SAILORS

WHERE SNAME='Horatio'

);

A screenshot of a computer

Description automatically generated

**Q9. Find the sailors with the highest rating.**

SELECT S.SNAME

FROM SAILORS S

WHERE S.RATING = (

SELECT MAX(RATING)

FROM SAILORS

);

A screenshot of a phone

Description automatically generated

**Q10. Find the names of sailors who have reserved all boats.**

SELECT S.SNAME

FROM SAILORS S

WHERE NOT EXISTS(

SELECT B.BID

FROM BOATS B

WHERE NOT EXISTS (

SELECT R.BID

FROM RESERVES R

WHERE R.BID=B.BID

AND R.SID = S.SID

)

);

A screenshot of a computer

Description automatically generated

**PLSQL (LAB Assignments-1)**

1. **WAP to find the greatest of three numbers.**

declare

a number:=20;

b number:=10;

c number:=30;

m number:=a;

begin

if (b>m) then

m:=b;

end if;

if (c>m) then

m:=c;

end if;

dbms\_output.put\_line(m);

end;

/

A black background with white text

Description automatically generated

1. **WAP to check whether number is odd or even.**

declare

a number;

begin

a:=&a;

if mod(a,2)=1 then

dbms\_output.put\_line('odd');

else

dbms\_output.put\_line('even');

end if;

end;

/

**A black background with white text

Description automatically generated**

**3) WAP to find the grade. Consider the following:**

**Marks > 80 A grade**

**Marks >70 B grade**

**Marks >50 C grade**

**Marks > 40 D grade**

**Marks < 40 E grade**

declare

marks number;

begin

marks:=&marks;

if (marks > 80) and (marks < 100) then

dbms\_output.put\_line('A grade');

elsif (marks>70) and (marks<=80) then

dbms\_output.put\_line('B grade');

elsif (marks>50) and (marks<=70) then

dbms\_output.put\_line('C grade');

elsif (marks>40) and (marks<=50) then

dbms\_output.put\_line('D grade');

elsif (marks<=40) and (marks>=0) then

dbms\_output.put\_line('E grade');

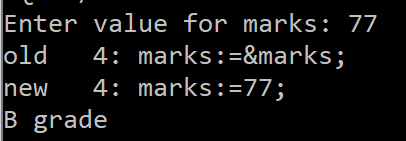
else

dbms\_output.put\_line('invalid');

end if;

end;

/



1. **WAP to print the table of a given number.(use for loop)**

declare

n number;

m number;

counter number;

begin

n:=&n;

for counter in 1..10 loop

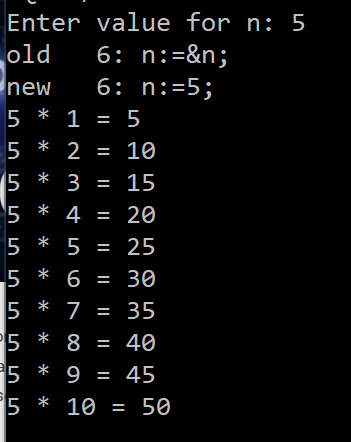
m:=n\*counter;

dbms\_output.put\_line(n||' \* '||counter||' = '||m);

end loop;

end;

/

****

1. **WAP to find out the factorial of a given number.(use while loop)**

declare

n number;

ans number;

begin

n:=&n;

ans:=n;

while n>2

loop

n:=n-1;

ans:=ans\*n;

end loop;

dbms\_output.put\_line(ans);

end;

/

**A black background with white text

Description automatically generated**

1. **WAP to find out the Fibonacci series.**

declare

n number;

n1 number;

n2 number;

n3 number;

i number;

begin

n:=&n;

n1:=0;

n2:=1;

for i in 1..n loop

n3:=n1+n2;

dbms\_output.put\_line(n3);

n1:=n2;

n2:=n3;

end loop;

end;

/

**A black screen with white text

Description automatically generated**

1. **WAP to find the reverse of a number**

declare

n number;

n1 number;

a number;

p number:=0;

i number:=0;

digits number:=0;

begin

n:=&n;

n1:=n;

while n1>0 loop

digits:=digits+1;

n1:=trunc(n1/10,0);

end loop;

while (n>0) loop

i:=i+1;

a:=mod(n,10);

p:=(a\*power(10,(digits-i)))+p;

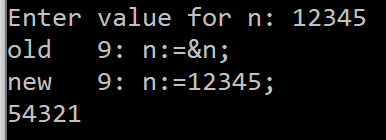
n:=trunc(n/10,0);

end loop;

dbms\_output.put\_line(p);

end;

/

****

**8) Write PL/SQL block that performs addition (+), subtraction (-), multiplication (\*) and**

**division (/) of two numbers as choice by the user.**

declare

op1 number;

op2 number;

ans number;

operand varchar2(1);

c varchar2(1);

begin

op1:=&op1;

op2:=&op2;

operand:=&operand;

case operand

when '+' then

ans:=op1+op2;

when '-' then

ans:=op1-op2;

when '\*' then

ans:=op1\*op2;

when '/' then

ans:=op1/op2;

else

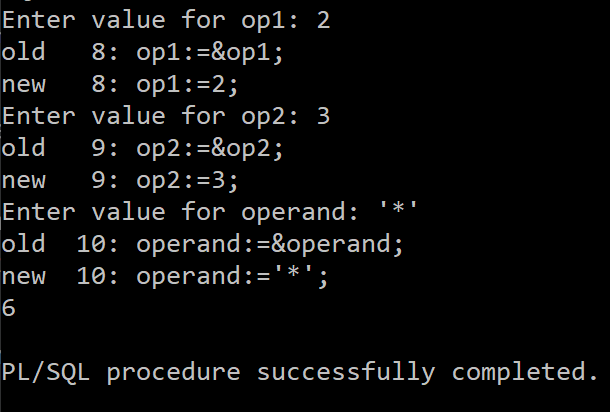
ans:=(-1);

end case;

dbms\_output.put\_line(ans);

end;

/

****

**9) Write PL/SQL block to print 5, 10, 15,20 by using For Loop.**

declare

i number;

counter number;

begin

for counter in 1..4 loop

i:=5\*counter;

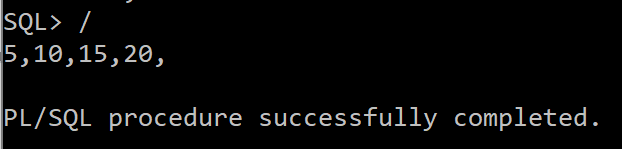
dbms\_output.put(i||',');

end loop;

dbms\_output.new\_line;

end;

/

****

**10) Write Pl/SQL block to display welcome message like good morning, good afternoon, good night depending on system time.**

declare

t timestamp:=systimestamp;

begin

if t>3 and t<12 then

dbms\_output.put\_line('gm');

elsif t>=12 and t<=16 then

dbms\_output.put\_line('ga');

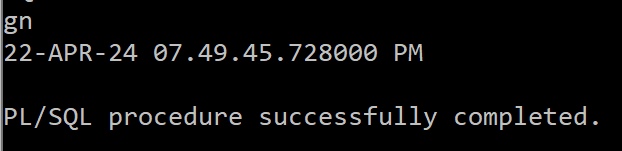
else

dbms\_output.put\_line('gn');

end if;

dbms\_output.put\_line(t);

end;



**PL-SQL (Lab-2)**

**(Procedures and Functions in PL/SQL)**

**1. Create table EMP (eno, ename, bp, da, hra, total). Initially, the ‘total’ column values for**

**all the employees are null. Write a stored procedure that accepts the eno of an employee**

**as an input and computes his total by adding bp + da + hra. Your pl/sql code should send**

**the total to the main block for printing and update it on the emp table.**

**A black background with white text

Description automatically generated**

create or replace procedure totalsal(empno in number, t out number) is

begin

select bp+da+hra into t from emp where eno=empno;

update emp set total=t where empno=eno;

end;

SQL> /

declare

t number;

begin

totalsal(101,t);

dbms\_output.put\_line(t);

end;

/

**A screen shot of a computer

Description automatically generated**

**2. Write a local function that will accept a number as an input, compute it factorial, and**

**send it to the main block for printing.**

declare

n number:=5;

factans number;

function fact(n1 in number) return number as ans number:=1;

begin

for i in 1..n1 loop

ans:=i\*ans;

end loop;

return ans;

end fact;

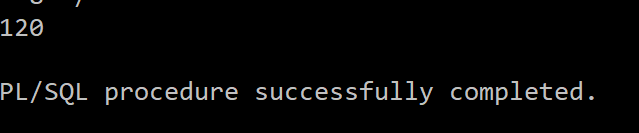
begin

factans:=fact(n);

dbms\_output.put\_line(factans);

end;

/

****

**3. Write a stored procedure that will accept a number, check whether positive, negative or**

**zero and accordingly print the message.**

create or replace procedure fun(n in number) is

begin

if (n>0) then

dbms\_output.put\_line('positive number');

elsif (n<0) then

dbms\_output.put\_line('negative');

else

dbms\_output.put\_line('zero');

end if;

end;

SQL> /

A black screen with white text

Description automatically generated

**4. Write a stored function that will find the sum of elements from 1 to n and send the result**

**to the main block. ‘n’ should send as the input to the called function.**

create or replace function sumN(n in number) return number is s number:=0;

begin

for i in 1..n loop

s:=s+i;

end loop;

return s;

end;

/

declare

ans number;

begin

ans:=sumN(5);

dbms\_output.put\_line(ans);

end;

/

**A black background with white text

Description automatically generated**

**5. Write a local procedure that will accept two ranges n1 and n2, find the sum of the average of elements between these ranges and return the result to the main block.**

declare

s number:=0;

ans2 number:=0;

procedure f1(n1 in number, n2 in number, ans out number) is

begin

for i in n1..n2 loop

s:=s+i;

end loop;

ans:=s/(n2-n1+1);

end;

begin

f1(2,5,ans2);

dbms\_output.put\_line(ans2);

end;

/

**A black background with white text

Description automatically generated**

**6. Write a recursive function to display ‘hello world’ 10 times on the screen. The function**

**should accept the string as input from the main block.**

create or replace procedure hw(s in varchar2, n in number) is

begin

if n>0 then

dbms\_output.put\_line(s);

hw(s,n-1);

end if;

end;

/

declare

s varchar2(20):='hello world';

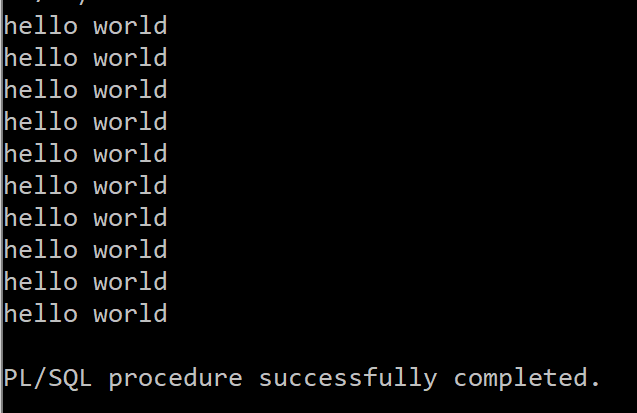
n number:=10;

begin

hw(s,n);

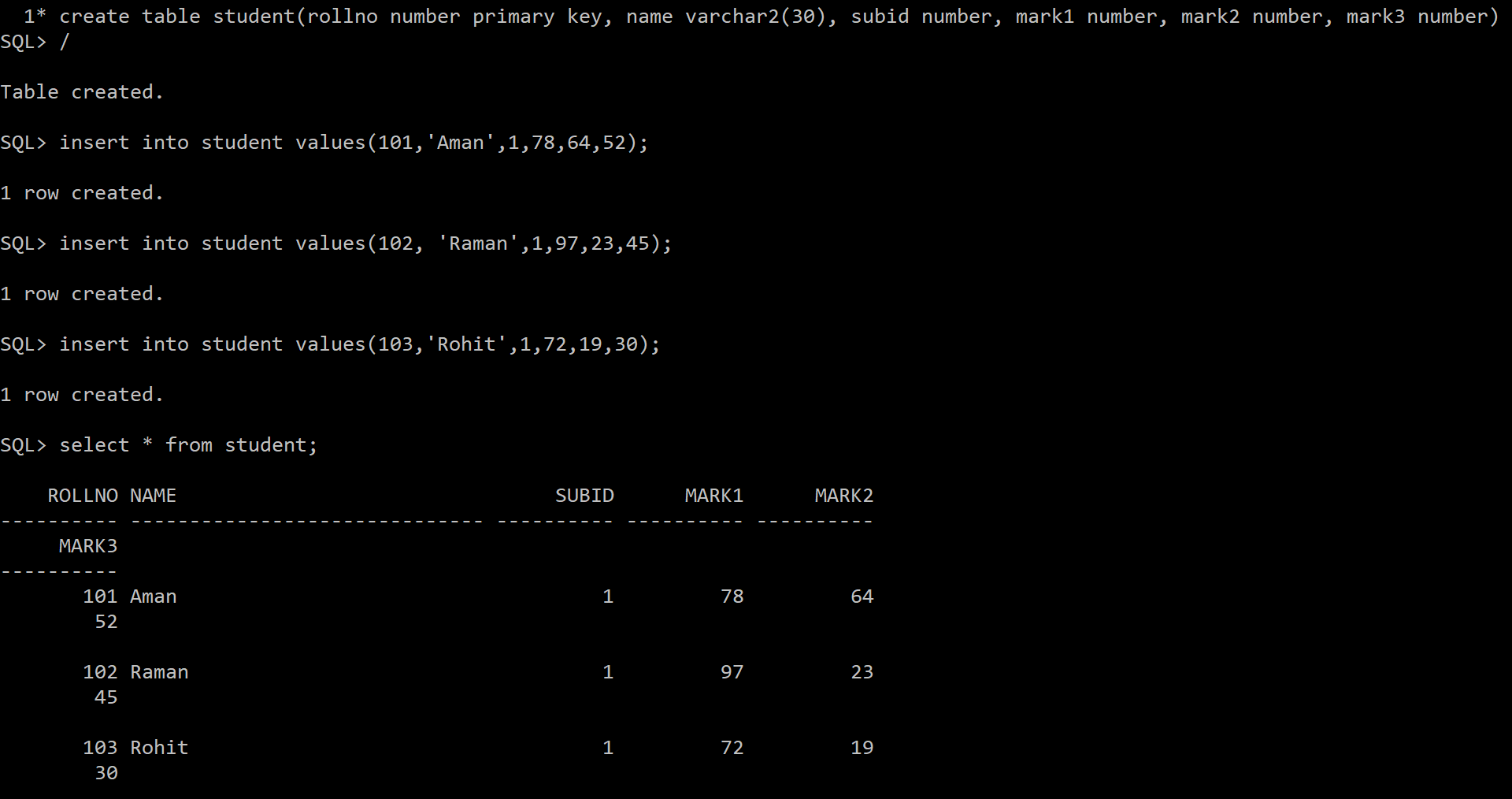
end;

/

****

**Consider a table student (rollno number, name varchar (30), sub-id number, mark1**

**number, mark2 number, mark3 number). Q7-Q10 are based on the student table.**

****

**7. Write a stored procedure ‘p1’, which will accept the rollno of a student as the input, and**

**then find the sum of the three subject marks and keep it in a variable ‘total’. Send this**

**‘total’ to the main block of the program and display it.**

create or replace procedure p1(rno in number, total out number)

is

begin

select mark1+mark2+mark3 into total from student where student.rollno=rno;

end;

/

declare

rno number;

total number;

begin

rno:=&rno;

p1(rno,total);

dbms\_output.put\_line(total);

end;

/

**A black background with white text

Description automatically generated**

**8. Write a local function ‘fun2’, which will call the procedure ‘p1’ of Q7 by passing the**

**rollno of a student. It then finds the average of the total marks and stores it in a variable**

**‘avg’. Send this ‘avg’ to the main block of the program and display it.**

declare

avg2 number;

total number:=0;

function fun2(rno in number) return number is avg1 number;

begin

p1(rno,total);

avg1:=total/3;

return avg1;

end fun2;

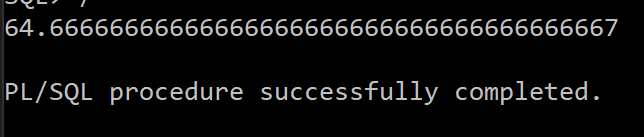
begin

avg2:=fun2(101);

dbms\_output.put\_line(to\_char(avg2));

end;

/

****

**9. Write a stored function ‘fun3’ which will accept the rollno of a student as the input and**

**return the highest marks among the three subjects to the main block.**

create or replace function fun3(rno in number) return number is m number;

begin

select greatest(mark1, mark2, mark3) into m from student where rollno=rno;

return m;

end;

/

declare

m number;

begin

m:=fun3(103);

dbms\_output.put\_line(m);

end;

/

**A black background with white text

Description automatically generated**

**10. Write a local procedure p2, which will call the procedure ‘p1’ of Q7 by passing the rollno**

**of a student. It then received the total marks from the procedure ‘p1’ and sent the**

**student’s three subject marks and the total marks to the main block for printing.**

declare

m1 number;

m2 number;

m3 number;

total number;

rno number;

procedure p2(rno in number, m1 out number, m2 out number, m3 out number, total out number) is

begin

p1(rno,total);

select mark1, mark2, mark3 into m1,m2,m3 from student where rollno=rno;

end;

begin

p2(102,m1,m2,m3,total);

dbms\_output.put\_line('mark1: '||m1);

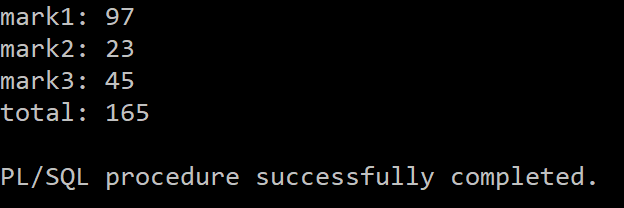
dbms\_output.put\_line('mark2: '||m2);

dbms\_output.put\_line('mark3: '||m3);

dbms\_output.put\_line('total: '||total);

end;

/

****

**PL-SQL\_Lab-3**

**(Cursors in PL/SQL)**

**Consider the table Student (Rollno, name, age, mark1, mark2, mark3, total). (For**

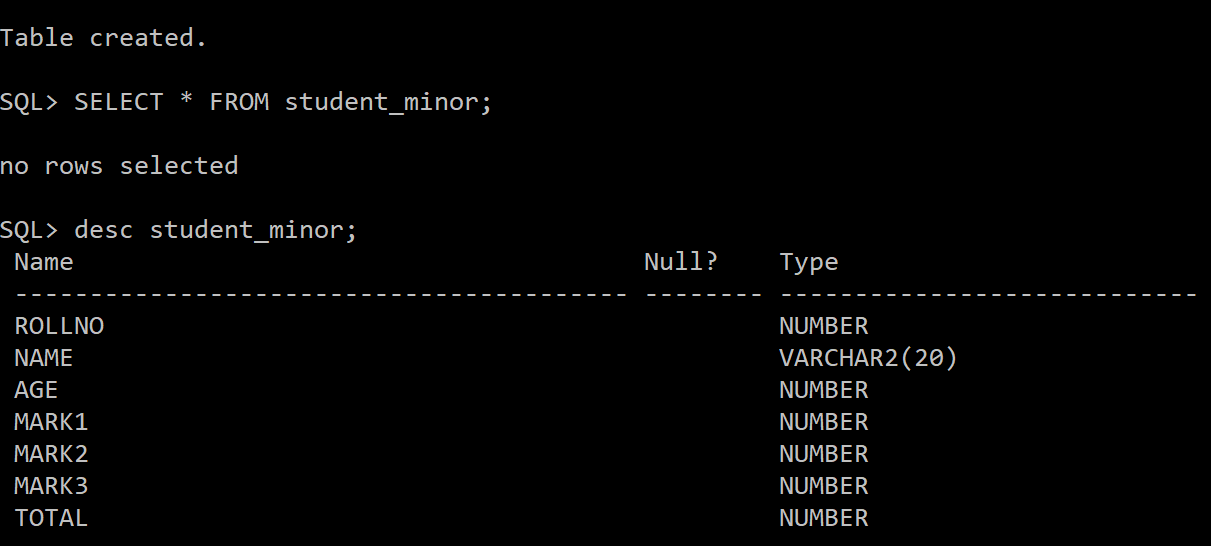
**Q1 to Q6)**

**1. Write a pl/sql code using cursor, which will delete all those records from the**

**Student table where age < 25 and insert those records into another table**

**called Student-minor.**

CREATE TABLE student\_minor AS SELECT \* FROM student WHERE 1=2;



DECLARE

CURSOR C1 IS SELECT \* FROM student WHERE AGE<25;

temp C1%ROWTYPE;

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO temp;

EXIT WHEN C1%NOTFOUND;

INSERT

INTO student\_minor VALUES(temp.rollno, temp.name,temp.age, temp.mark1, temp.mark2, temp.mark3, temp.total);

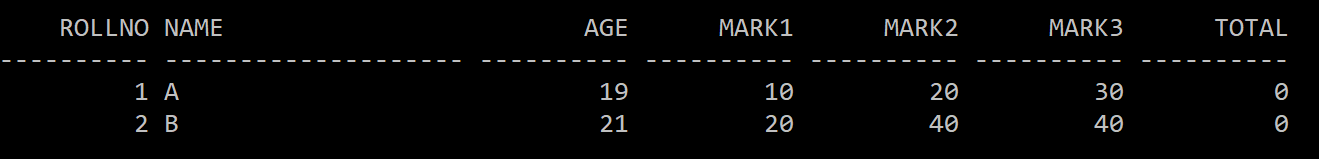
DELETE FROM student WHERE rollno=temp.rollno;

END LOOP;

CLOSE C1;

END;

/



A black screen with white text

Description automatically generated

**2. Write a pl/sql code using cursor, which will find the total marks of each**

**student and update the total column (assume that initially, the total is zero**

**for all the students).**

DECLARE

CURSOR C1 IS SELECT \* FROM student;

temp C1%ROWTYPE;

t student.total%TYPE;

BEGIN

OPEN C1;

LOOP

FETCH C1 INTO temp;

EXIT WHEN C1%NOTFOUND;

SELECT temp.mark1+temp.mark2+temp.mark3 INTO t FROM STUDENT WHERE temp.rollno=student.rollno;

UPDATE student SET student.total=t WHERE temp.rollno=student.rollno;

END LOOP;

CLOSE C1;

END;

/

**A screenshot of a computer

Description automatically generated**

**3. Write a pl/sql code using cursor to find out how many students are there**

**whose total marks are greater than 90, and then display their details.**

DECLARE

c number:=0;

CURSOR C1 IS SELECT \* FROM STUDENT WHERE TOTAL>90;

BEGIN

FOR r IN C1 LOOP

EXIT WHEN C1%NOTFOUND;

SELECT \* INTO r FROM STUDENT WHERE rollno=r.rollno;

DBMS\_OUTPUT.PUT\_LINE(r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

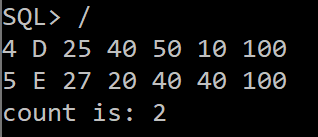
c:=c+1;

END LOOP;

dbms\_output.put\_line('count is: '||c);

END;

/



**4. Write a pl/sql code using cursor to find the highest and lowest marks and**

**display the corresponding student’s details.**

DECLARE

max\_marks NUMBER;

min\_marks NUMBER;

CURSOR C1 IS SELECT \* FROM STUDENT;

BEGIN

SELECT max(total) INTO max\_marks FROM student;

SELECT min(total) INTO min\_marks FROM student;

FOR R IN C1 LOOP

EXIT WHEN C1%NOTFOUND;

IF R.total=max\_marks THEN

DBMS\_OUTPUT.PUT\_LINE('MAX MARKS: '||r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

ELSIF R.total=min\_marks THEN

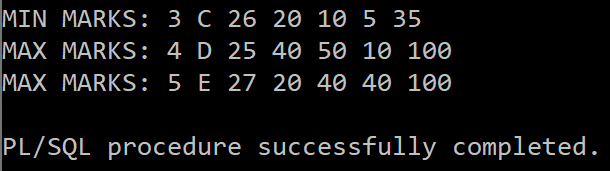
DBMS\_OUTPUT.PUT\_LINE('MIN MARKS: '||r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

END IF;

END LOOP;

END;

/



**5. Write a pl/sql code using cursor to find the average mark of all the students**

**and display it on the screen.**

DECLARE

a NUMBER(4,2);

CURSOR C1 IS SELECT \* FROM student;

BEGIN

FOR R IN C1 LOOP

EXIT WHEN C1%NOTFOUND;

SELECT (R.mark1+R.mark2+R.mark3)/3 INTO a FROM student WHERE rollno=R.rollno;

DBMS\_OUTPUT.PUT\_LINE('ROLLNO: '||R.rollno||' AVERAGE: '||a);

END LOOP;

END;

/

A screen shot of a black background

Description automatically generated

**6. Write a stored procedure using a parameterized cursor, which will display**

**the student details whose rollno is passing as a parameter to the cursor from**

**the stored procedure.**

CREATE OR REPLACE PROCEDURE f(n in number) AS

CURSOR C1 IS SELECT \* FROM student WHERE rollno=n;

BEGIN

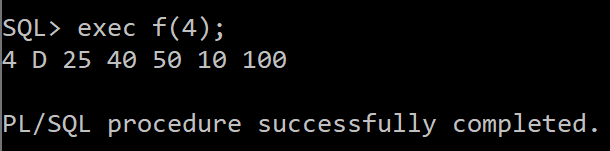
FOR R IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(r.rollno||' '|| r.name||' '||r.age||' '||r.mark1||' '||r.mark2||' '||r.mark3||' '||r.total);

END LOOP;

END**;**

**/**

****

**Consider the table EMP (empno, ename, job, sal, deptno) (For Q7 to Q10)**

**7. Write a PL/SQL code to demonstrate %TYPE and %ROWTYPE to display**

**details of employees in EMP table.**

**A black screen with white text

Description automatically generated**

DECLARE

CURSOR C1 IS SELECT \* FROM EMP WHERE EMPNO=1;

ENO EMP.EMPNO%TYPE;

R EMP%ROWTYPE;

BEGIN

FOR R IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(R.EMPNO||' '||R.ENAME||' '||R.JOB||' '||R.SAL||' '||R.DEPTNO);

SELECT EMPNO INTO ENO FROM EMP WHERE EMPNO=R.EMPNO;

DBMS\_OUTPUT.PUT\_LINE(ENO);

END LOOP;

END;

/

A black background with white text

Description automatically generated

**8. Write a stored function to display the empno, ename, and job of employees**

**of a department for EMP table using a parameterized cursor where deptno**

**will be sent as a parameter to the cursor from the stored function.**

CREATE OR REPLACE FUNCTION FUN(N IN NUMBER) RETURN NUMBER IS ANS NUMBER:=0;

CURSOR C1 IS SELECT \* FROM EMP WHERE EMPNO=N;

BEGIN

FOR R IN C1 LOOP

DBMS\_OUTPUT.PUT\_LINE(R.EMPNO||' '||R.ENAME||' '||R.JOB);

END LOOP;

ANS:=1;

RETURN ANS;

END;

/

DECLARE

ANS NUMBER;

BEGIN

ANS:=FUN(2);

END;

/

A black background with white text

Description automatically generated

**9. Write a local function to display the employee number and name of the top**

**‘n’ highest-paid Employees using parameterized cursor. The value of ‘n’ is**

**passed to the cursor as a parameter from the local function.**

DECLARE

ANS NUMBER;

FUNCTION FUN(N IN NUMBER) RETURN NUMBER IS ANS NUMBER:=0;

CURSOR C1 IS SELECT \* FROM EMP;

ENO NUMBER;

BEGIN

FOR R IN C1 LOOP

SELECT EMPNO INTO ENO FROM EMP E1 WHERE N-1 = (SELECT COUNT(\*) FROM EMP E2 WHERE E2.SAL>E1.SAL);

IF R.EMPNO=ENO THEN

DBMS\_OUTPUT.PUT\_LINE(R.EMPNO||' '||R.ENAME);

END IF;

END LOOP;

ANS:=1;

RETURN ANS;

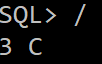
END;

BEGIN

ANS:=FUN(2);

END;

/



**10. Write a local procedure to calculate the total salary of the first ‘n’ records of**

**EMP table using parameterized cursor. The value of ‘n’ is passed to cursor**

**as a parameter from the local procedure.**

DECLARE

N NUMBER:=3;

S NUMBER:=0;

PROCEDURE FINDSUM(N IN NUMBER) IS

CURSOR C1 IS SELECT \* FROM EMP;

C NUMBER:=0;

BEGIN

FOR R IN C1 LOOP

EXIT WHEN C>=N OR C1%NOTFOUND;

S:=S+R.SAL;

C:=C+1;

END LOOP;

END;

BEGIN

FINDSUM(N);

DBMS\_OUTPUT.PUT\_LINE(S);

END;

/

A black background with white text

Description automatically generated

**(Exception Handling in PL/SQL)**

**11. Write a PL/SQL program to demonstrate the following exceptions:**

**When Too Many Rows**

**When No Data Found**

**When Others**

--too many rows exception

DECLARE

A NUMBER;

BEGIN

SELECT EMPNO INTO A FROM EMP;

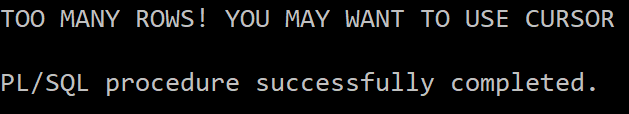
EXCEPTION

WHEN TOO\_MANY\_ROWS THEN

DBMS\_OUTPUT.PUT\_LINE('TOO MANY ROWS! YOU MAY WANT TO USE CURSOR');

END;

/



--NO DATA FOUND ERROR

DECLARE

A NUMBER;

BEGIN

SELECT EMPNO INTO A FROM EMP WHERE EMPNO=5;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR! NO DATA FOUND');

END;

/

A black background with white text

Description automatically generated

--OTHERS

--HERE NO SUCH TABLE EXISTS

DECLARE

A NUMBER;

BEGIN

SELECT EMPNO INTO A FROM EMP1 WHERE EMPNO=5;

EXCEPTION

WHEN OTHERS THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR!');

END;

/

A black background with white text

Description automatically generated

**12. Write a PL/SQL program to demonstrate the User Defined Exceptions.**

DECLARE

A NUMBER;

B NUMBER;

C NUMBER;

EX EXCEPTION;

BEGIN

A:=&A;

B:=&B;

IF MOD(B,A)<>0 THEN

RAISE EX;

END IF;

C:=B/A;

DBMS\_OUTPUT.PUT\_LINE(C);

EXCEPTION

WHEN EX THEN

DBMS\_OUTPUT.PUT\_LINE('ERROR! ENTER MULTIPLE OF FIRST NUMBER AS SECOND NUMBER');

END;

/

**A black screen with white text

Description automatically generated**

**A black screen with white text

Description automatically generated**

**Assignment-12**

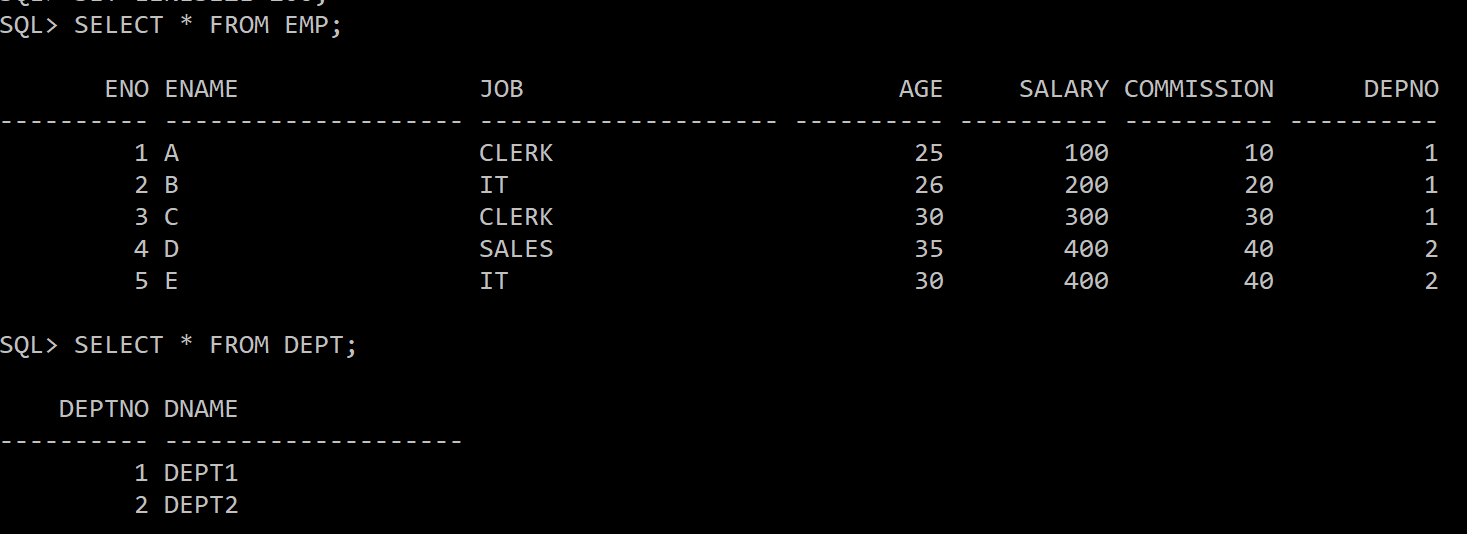
**(Database Triggers in PLSQL)**

**EMP (ENO, ENAME, JOB, AGE, SALARY, COMMISSION, DEPTNO).**

**DEPT (DEPTNO, DNAME)**

**(Deptno of EMP is a foreign key which refers to the Deptno of Dept Table)**

**Assume and create required tables with appropriate attributes, if any**

****

**1. Write a database trigger on the EMP table, which will store each deleted**

**record from the EMP table in another table called EMP\_OLD.**

CREATE OR REPLACE TRIGGER EONE

BEFORE DELETE ON EMP

FOR EACH ROW

BEGIN

INSERT INTO EMP\_OLD VALUES(:OLD.ENO, :OLD.ENAME, :OLD.JOB, :OLD.AGE, :OLD.SALARY, :OLD.COMMISSION, :OLD.DEPTNO);

END;

/

A black screen with white text

Description automatically generated

**2. Create a database trigger on the EMP table, verifying that no employee is**

**under 25 while entering the details (ENO, AGE) into the EMP table.**

CREATE TRIGGER ETWO

AFTER INSERT ON EMP FOR EACH ROW

BEGIN

IF :NEW.AGE<25 THEN

RAISE\_APPLICATION\_ERROR(-20000, 'AGE IS LESS THAN 25');

END IF;

END;

/

**A black screen with white text

Description automatically generated**

**3. Write a database trigger on the EMP table, which verifies that each employee's**

**updated salary should not less than their old salary.**

CREATE TRIGGER ETHREE

BEFORE UPDATE ON EMP FOR EACH ROW

BEGIN

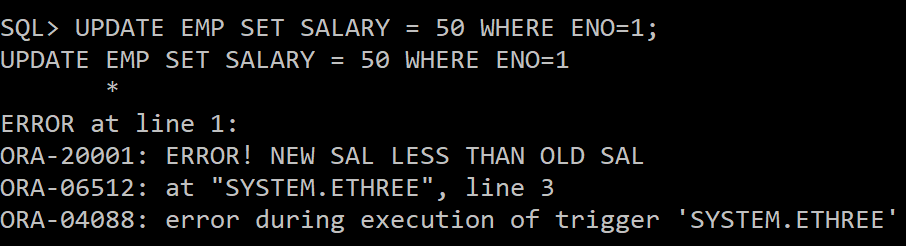
IF :NEW.SALARY<:OLD.SALARY THEN

RAISE\_APPLICATION\_ERROR(-20001,'ERROR! NEW SAL LESS THAN OLD SAL');

END IF;

END;

/

****

**4. Write a database trigger which is an example of a statement-level trigger.**

CREATE TRIGGER EFOUR

BEFORE UPDATE ON EMP

BEGIN

DBMS\_OUTPUT.PUT\_LINE('TABLE EMP IS UPDATED');

END;

/

**A black screen with white text

Description automatically generated**

--here four rows are updated but the trigger is only fired once

**5. Create a trigger on the EMP table to convert supplied values of the ENAME**

**column to uppercase for INSERT and UPDATE statements.**

**CREATE TRIGGER EFIVE**

**BEFORE INSERT OR UPDATE ON EMP**

**FOR EACH ROW**

**BEGIN**

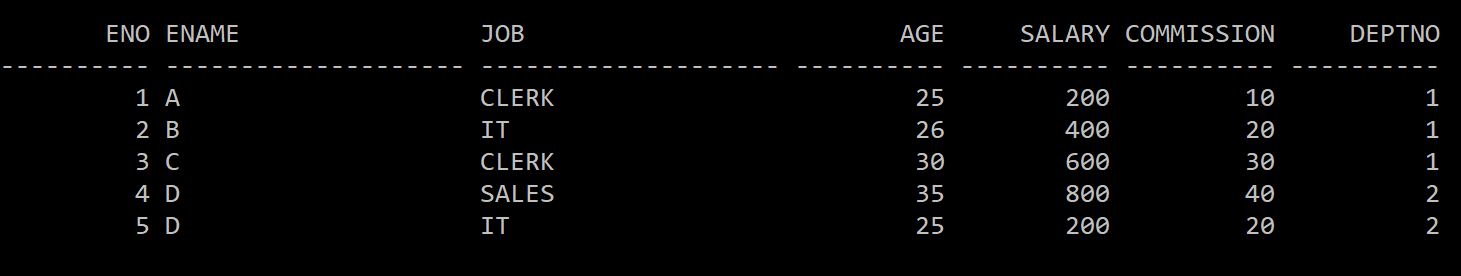
**:NEW.ENAME:=UPPER(:NEW.ENAME);**

**END;**

**/**

**INSERT INTO EMP VALUES(5,’d’,’IT’, 25, 200, 20, 2);**

**SELECT \* FROM EMP;**

****

**6. Create a trigger on the emp table, which shows the old and new values of**

**ENAME after every updation on the ENAME of the EMP table.**

CREATE TRIGGER ESIX

AFTER UPDATE ON EMP

FOR EACH ROW

BEGIN

IF :OLD.ENAME <> :NEW.ENAME THEN

DBMS\_OUTPUT.PUT\_LINE('OLD NAME IS: '||:OLD.ENAME);

DBMS\_OUTPUT.PUT\_LINE('NEW NAME IS: '||:NEW.ENAME);

END IF;

END;

/

UPDATE EMP SET ENAME='E' WHERE ENO=5;

A black screen with white text

Description automatically generated

**7. Write a trigger to ensure that an employee's commission cannot be greater**

**than his salary.**

CREATE TRIGGER ESEVEN

BEFORE INSERT OR UPDATE ON EMP

FOR EACH ROW

BEGIN

IF :NEW.COMMISSION > :NEW.SALARY THEN

RAISE\_APPLICATION\_ERROR(-20003, 'COMMISSION CANNOT BE GREATER THAN SALARY');

END IF;

END;

/

INSERT INTO EMP VALUES(7,'F','IT',28,10,30,2);

A black screen with white text

Description automatically generated

**8. Create a trigger so that no operation can be performed on the EMP table on**

**Sunday.**

CREATE TRIGGER EEIGHT

BEFORE INSERT OR DELETE OR UPDATE ON EMP

FOR EACH ROW

DECLARE

N VARCHAR2(20);

BEGIN

SELECT TO\_CHAR(SYSDATE,'DAY') INTO N FROM DUAL;

IF N='SUNDAY' THEN

RAISE\_APPLICATION\_ERROR(-20004, 'CANNOT MAKE CHANGES ON SUNDAY');

END IF;

END;

/

**9. Create a trigger to implement the primary key constraint on column ENO of**

**table EMP.**

CREATE TRIGGER ENINE

BEFORE INSERT OR UPDATE ON EMP

FOR EACH ROW

BEGIN

IF :NEW.ENO IS NULL THEN

RAISE\_APPLICATION\_ERROR(-20000,'PRIMARY KEY NOT NULL');

END IF;

FOR R IN (SELECT ENO FROM EMP WHERE ENO=:NEW.ENO) LOOP

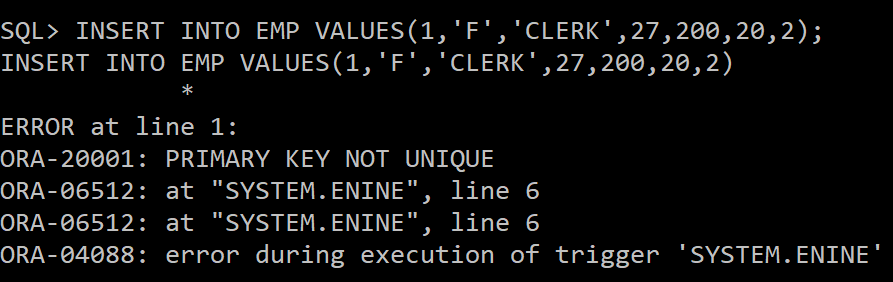
RAISE\_APPLICATION\_ERROR(-20001,'PRIMARY KEY NOT UNIQUE');

END LOOP;

END;

/

INSERT INTO EMP VALUES(1,'F','CLERK',27,200,20,2);



**10. Write a trigger to implement a foreign key constraint on the DEPTNO column**

**of the EMP table which refers to the DEPTNO of DEPT Table.**

CREATE OR REPLACE TRIGGER ETEN

BEFORE UPDATE OR INSERT ON EMP

FOR EACH ROW

DECLARE

D NUMBER;

BEGIN

SELECT DEPT.DEPTNO INTO D FROM DEPT WHERE DEPTNO=:NEW.DEPTNO;

EXCEPTION

WHEN NO\_DATA\_FOUND THEN

RAISE\_APPLICATION\_ERROR(-20005,'FOREIGN KEY CONSTRAINT VIOLATED');

END;

/

INSERT INTO EMP VALUES (6,'F','SALES',28,200,20,3);

