

**Q.1) CREATE A TABLE TO REPRESENT SB-ACCOUNT OF A BANK
CONSISTING OF ACCOUNT-NO,
CUSTOMER-NAME, BALANCE-AMOUNT.
WRITE A PL/SQL BLOCK TO IMPLEMENT DEPOSIT AND WITHDRAW.
WITHDRAWS SHOULD NOT BE ALLOWED IF THE BALANCE GOES
BELOW RS.1000.**

SOLUTION 1.

```
CREATE TABLE SB_ACCOUNT (
```

```
ACCOUNT_NO VARCHAR2(10),  
CUSTOMER_NAME VARCHAR2(25),  
BALANCE NUMBER(10,2));
```

Declare

```
mcbal number(10,2);  
mact_no varchar2(10);  
withdraw number(10,2);
```

begin

```
mact_no := &mact_no;  
select balance into mcbal from sb_account where account_no=mact_no;  
withdraw := &withdraw;  
mcbal := mcbal-withdraw;
```

if mcbal < 1000 then

```
dbms_output.put_line('WITHDRAWAL NOT POSSIBLE BALANCE BELOW 1000');
```

else

```
update sb_account set balance = mcbal where account_no = mact_no;
```

end if;

end;

```
/
```

Output

```
SQL> ed
Wrote file afiedt.buf

 1  declare
 2  mcbal number(10,2);
 3  mact_no varchar2(10);
 4  withdraw number(10,2);
 5  begin
 6  mact_no := &mact_no;
 7  select balance into mcbal from sb_account where account_no=mact_no;
 8  withdraw := &withdraw;
 9  mcbal := mcbal-withdraw;
10  if mcbal < 1000 then
11  dbms_output.put_line('sorry balance goes below 1000');
12  else
13  update sb_account set balance = mcbal where account_no = mact_no;
14  end if;
15* end;
SQL> /
Enter value for mact_no: 106
old 6: mact_no := &mact_no;
new 6: mact_no := 106;
Enter value for withdraw: 1000
old 8: withdraw := &withdraw;
new 8: withdraw := 1000;
sorry balance goes below 1000

PL/SQL procedure successfully completed.
```

Q. 2: CREATE THE FOLLOWING TWO TABLES: COLLEGE-INFO, FACULTY-INFO, COLLEGE-INFO CONSISTS OF FIELDS: COLLEGE-CODE, COLLEGE-NAME, ADDRESS. FACULTY-INFO CONSISTS OF FIELDS: COLLEGE-CODE, FACULTY-CODE, FACULTY-NAME, QUALIFICATION, EXPERIENCE-IN-NO-OF-YEARS, ADDRESS.

The field college-code is a foreign key, Generate queries to do the following:

- 1 . List all those faculty members whose experience is greater than equal to 10 years and have MCA degree.
2. List all those faculty members who have at least 10 years of experience but do not have MCA degree.

Solution 2:

```
CREATE TABLE COLLEGE_INFO( COLLEGE_CODE VARCHAR2(10) PRIMARY KEY, COLLEGE_NAME VARCHAR2(30), ADDRESS VARCHAR2(40));
```

```
CREATE TABLE FACULTY_INFO( FACULTY_CODE VARCHAR2(10) PRIMARY KEY, COLLEGE_CODE VARCHAR2(10) REFERENCES COLLEGE_INFO, FACULTY_NAME VARCHAR2(25),
```

QUALIFICATON VARCHAR2(15),
EXP NUMBER(2),
ADDRESS VARCHAR2(30));

QUERY:

1. SELECT * FROM FACULTY_INFO WHERE EXP >= 10 AND QUALIFICATON LIKE 'MCA';

2. SELECT * FROM FACULTY_INFO WHERE EXP >= 10 AND QUALIFICATON NOT LIKE 'MCA';

OUTPUT

1.

FACULTY_CODE	COLLEGE_CODE	FACULTY_NAME	QUALIFICATON	EXP	ADDRESS
F01	C01	NAJEEB	MCA	11	DELHI
F03	C01	SAKEEB	MCA	13	NOIDA
F04	C02	INDRA SINGH	MCA	14	GURGAON
F05	C02	SURESH SINGH	MCA	12	FARIDABAAD
F06	C02	VISHWAJEET	MCA	15	DELHI

2.

FACULTY_CODE	COLLEGE_CODE	FACULTY_NAME	QUALIFICATON	EXP	ADDRESS
F02	C01	JYOTI	BCA	15	NOIDA
F07	C03	PANKAJ	BCA	10	LUCKNOW
F08	C05	GUNJAN	BCA	10	MANESHAR
F09	C04	ASHOK	BCA	17	DELHI

**Q.3)CREATE THE FOLLOWING TABLES FOR LIBRARY INFORMATION SYSTEM:
BOOK(ACCESSION-NO,TITLE,PUBLISHER,AUTHOR,STATUS)
STATUS COULD BE 'ISSUED', 'PRESENT IN THE LIBRARY', 'SENT FOR BINDING', & 'CAN NOT BE ISSUED'
WRITE A TRIGGER WHICH SETS THE STATUS OF A BOOK TO "CAN NOT BE ISSUED", IF IT IS PUBLISHED 20 YEARS BACK.**

Solution no 3:

create or replace trigger checkbook
before insert or update on book
for each row

Declare
dop book.date_of_purchase%type;
yrs number(10);

Begin
dop := :new.date_of_purchase;
yrs := (months_between(sysdate,dop))/12;

if (yrs > 20) then

:new.status := 'CANNOT BE ISSUED';
dbms_output.put_line('This Book Is 20 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"');

end if;

End;

Query:

insert into book values('21','operating system','wiley','galvin','issued','11-mar-1987')

OUTPUT

This Book Is 20 Years Old, Its Status Has Been Changed To "CANNOT BE ISSUED"

1 row(s) inserted.

ACCESSION_NO	TITLE	PUBLISHER	AUTHOR	STATUS	DATE_OF_PURCHASE
7	winproc	petzold	charles	issued	11-MAR-87
8	windows	petzold	charles	issued	11-MAR-87
9	vc++	petzold	charles	issued	11-MAR-87
15	digital	wimble	M M mano	CANNOT BE ISSUED	11-MAR-88
15	digital	wimble	M M mano	CANNOT BE ISSUED	11-MAR-88
20	sad	doacc	kranti	CANNOT BE ISSUED	11-MAR-88
21	operating system	wiley	galvin	CANNOT BE ISSUED	11-MAR-87

13 rows returned in 0.00 seconds CSV Export

Q.4)CREATE THE FOLLOWIN TABLES FOR LIBRARY INFORMATION SYSTEM:

BOOK(ACCESSION-NO,TITLE,PUBLISHER,AUTHOR,STATUS)

STATUS COULD BE ‘ISSUED’, ‘PRESENT IN THE LIBRARY’, ‘SENT FOR BINDING’, & ‘CAN NOT BE ISSUED’

GENERATE QUERIES TO DO THE FOLLOWING:

- 1) LIST ALL THOSE BOOKS WHICH ARE NEW ARRIVALS.THE BOOKS WHICH ARE ACQUIRED DURING THE LAST 6 MONTs ARE CATEGORIZED AS NEW ARRIVALS.**
- 2) LIST ALL THOSE BOOKS THAT A\CANNOT BE ISSUED AND PURCHASED 20 YERS AGO.**

SOLUTION: 4

```
CREATE TABLE BOOK(
ACCESSION_NO VARCHAR2(10),
TITLE VARCHAR2(25),
PUBLISHER VARCHAR2(25),
AUTHOR VARCHAR2(25),
STATUS VARCHAR2(30),
DATE_OF_PURCHASE DATE );
```

QUERY

1.

select accession_no,title,author,date_of_purchase, (sysdate) " current date" from book
 where months_between(sysdate, date_of_purchase) < 6;

2.

select accession_no,title,author,date_of_purchase, (sysdate) " current date", status from
 book where status like 'cannot be issued' and months_between(sysdate,
 date_of_purchase) > 20;

OUTPUT

1.

ACCESSION_NO	TITLE	AUTHOR	DATE_OF_PURCHASE	CURRENT DATE
A01	OOP	NARYAN MURTHY	22-JUN-10	12-NOV-10
A05	SAD	RMAKRISHNAH	20-AUG-10	12-NOV-10
A06	DS	KANETKAR	10-FEB-10	12-NOV-10

2.

ACCESSION_N O	TITLE	AUTHOR	DATE_OF_PURCHASE	CURRENT DATE	STATUS
A01	BBB	GRISHAM	21-OCT-88	12-NOV-10	CANNOT BE ISSUED
A03	XYZ	GRISHASAAM	25-JUL-80	12-NOV-10	CANNOT BE ISSUED
A04	CCC	JOHN	29-SEP-70	12-NOV-10	CANNOT BE ISSUED

Q.5)CREATE THE FOLLOWING TABLES:

STUDENT(ROLL-NO,NAME,DATE-OF-BIRTH,COURSE-ID)

COURSE(COURSE-ID,NAME,FEE,DURATION)

GENERATE QUERIES TO DO THE FOLLOWING:

**1.LIST ALL THOSE STUDENTS WHO ARE GREATER THAN 18 YEARS OF
 AGE AND HAVE OPTED FOR MCA COURSE.**

**2.LIST ALL THOSE COURSES WHOSE FEE IS GREATER THAN THAT OF
 MCA COURSE**

SOLUTION NO . 5

CREATE TABLE STUDENT (
 ROLL_NO VARCHAR2(10),
 NAME VARCHAR2(25),
 DATE_OF_BIRTH DATE,

```
COURSE_ID VARCHAR2(10));
```

```
CREATE TABLE COURSE(  
COURSE_ID VARCHAR2(10),  
NAME VARCHAR2(30),  
FEE NUMBER(6,2),  
DURATION NUMBER(2));
```

QUERY

1. select name,round ((months_between(sysdate,date_of_birth))/12) "age" from student where round(months_between(sysdate,date_of_birth))/12 > 18;
2. select course_id,name from course where fee > (select fee from course where course_id like 'mca');

OUTPUT

1.

NAME	AGE
NAVEEN	23
ANUJ ABHAYA	25

2.

COURSE_ID	NAME
MTECH	MASTERS TECHNOLOGY
BTECH	BACHELOR OF TECHNOLOGY
MPHARMA	MASTER OF PHARMACY

Q.6) CREATE THE FOLLOWING TABLES:

STUDENT(ROLL-NO,NAME,SUBJECT NAME,SUBJECT OPTED)

SUBJECT(FACULTY-CODE,FACULTY-NAME,SPECIALIZATION)

GENERATE QUERIES TO DO THE FOLLOWING:

1)FIND THE NUMBER OF STUDENTS WHO HAVE ENROLLED FOR THE SUBJECT “DBMS”.

2)FIND ALL THOSE FACULTY MEMBERS WHO HAVE NOT OFFERED ANY SUBJECT.

SOLUTION : 6

```
CREATE TABLE STUDENT0(  
ROLL_NO VARCHAR2(10),  
NAME VARCHAR2(25),  
SUBJECT_NAME VARCHAR2(25),  
SUBJECT_OPTED VARCHAR2(25));
```

```
CREATE TABLE SUBJECT(  
FACULTY_CODE VARCHAR2(10),  
FACULTY_NAME VARCHAR2(25),  
SPECIALIZATION VARCHAR2(25));
```

QUERY

1.FIND THE NUMBER OF STUDENTS WHO HAVE ENROLLED FOR THE SUBJECT “DBMS”.

2.FIND ALL THOSE FACULTY MEMBERS WHO HAVE NOT OFFERED ANY SUBJECT.

SOL:-

1.SELECT COUNT(*)"NO. OF STUDENTS " FROM STUDENT0 WHERE SUBJECT_NAME LIKE 'DBMS';

2.SELECT FACULTY_CODE, FACULTY_NAME FROM SUBJECT WHERE SPECIALIZATION IS NULL;

OUTPUT

1.

NO. OF STUDENTS
5

2.

FACULTY_CODE	FACULTY_NAME
JH04	RAKESH
JMI07	RAMAN
IP05	ANIL
DU06	VIKASH

Q.7) CREATE THE FOLLOWING TABLES:

ITEM(ITEM-CODE,ITEM-NAME,QTY-IN-STOCK,REORDER-LEVEL)

SUPPLIER(SUPPLIER-CODE,SUPPLIER-NAME,ADDRESS)

CAN-SUPPLY(SUPPLIER-CODE,ITEM-CODE)

GENERATE QUERIES TO DO THE FOLLOWIN:

1)LIST ALL THOSE SUPPLIERS WHO CAN SUPPLY THE GIVEN ITEM.

2)LIST ALL THOSE TIEMS WHICH CANNOT BE SUPPLIED BY GIVEN COMPANY.

SOLUTION 7

```
create table item(  
item_code varchar(10) primary key,  
item_name varchar(25),  
qty_inhand number(8),  
reorder_level number(8));
```

```
create table supplier(  
supplier_code varchar(10) primary key,  
supplier_name varchar(25),  
address varchar(30));
```

```
create table can_supply(  
supplier_code varchar2(10),  
item_code varchar2(10),  
primary key (supplier_code,item_code));
```

QUERY:

```
select supplier_name, supplier_code from supplier where supplier_code in(select  
supplier_code from can_supply where item_code in (select item_code from item where  
item_name like 'coke'));
```

SUPPLIER_NAME	SUPPLIER_CODE
LEYLAND	S01
HITACHI	S02
ASKIN	S03
BRUCH	S04

select item_code, item_name from item where item_code not in(select item_code from can_supply where supplier_code = 's01');

ITEM_CODE	ITEM_NAME
I05	BREADS
I06	EGG
I10	FRUITS

Q.8) CREATE THE FOLLOWING TABLES:

STUDENT(ROLL-NO,MARKS,CATEGORY,DISTRICT,STATE)

STUDENT-RANK(ROLL-NO,MARKS,RANK)

GENERATE QUERIES TO DO THE FOLLOWING:

1)LIST ALL THOSE STUDENTS WHO HAVE COME FROM “UP” STATE AND SECURED A RANK ABOVE 100;

2)LIST ALL THOSE STUDENTS WHO COME FROM “RAJASTHAN” STATE AND BELONG TO GIVEN CATEGORY WHO HAVE SECURED A RANK ABOVE 100.

SOLUTION 8

```
create table student2(
roll_no varchar2(10),
marks number(10),
category varchar2(10),
district varchar2(10),
state varchar2(10));
```

```
create table student_rank(
roll_no varchar2(10),
marks number(10),
rank number(5));
```

QUERY:

select roll_no, state from student2 where state = 'up'and roll_no in(select roll_no from student where rank< 100);

ROLL_NO	STATE
1	UP
2	UP

select roll_no, state, category from student2 where state like 'rajasthan' and category like 'g' and roll_no in(select roll_no from student_rank where rank < 100);

ROLL_NO	STATE	CATEGORY
5	RAJASTHAN	OBC
6	RAJASTHAN	OBC

Q.9) CREATE THE FOLLOWING TABLES:

BRANCH(BRANCH-ID,BRANCH-NAME,CUSTOMER-CITY,BRANCH-ID)

CUSTOMER(CUSTOMER-ID,CUSTOMER-NAME,CUSTOMER-CITY,BRANCH-ID)

GENERATE QUERIES TO DO THE FOLLOWING:

1)LIST ALL THOSE CUSTOMERS WHO LIVE IN THE SAME CITY AS THE BRANCH IN WHICH THEY HAVE ACCOUNT.

2)LIST ALL THOSE CUSTOMERS WHO HAVE AN ACCOUNT IN A GIVEN BRANCH CITY

SOLUTION 9.

```
create table branch(  
branch_id varchar2(10),  
branch_name varchar2(20),  
branch_city varchar(10));
```

```
create table customer(  
customer_id varchar2(10),  
customer_name varchar2(20),  
customer_city varchar2(10),  
branch_id varchar2(10));
```

query:

```
select A.customer_id, A.customer_name, B.branch_id, B.branch_city, A.customer_city  
from customer A,  
(select branch_id, branch_city from branch) B  
where A.branch_id = B.branch_id and A.customer_city = B.branch_city;
```

CUSTOMER_ID	CUSTOMER_NAME	BRANCH_ID	BRANCH_CITY	CUSTOMER_CITY
K001	NAVEEN	KTR1	KATIHAR	KATIHAR
PR05	ANUJ	PRS	PARASNATH	PARASNATH
ST06	HUSSAIN	STM	SITAMARHI	SITAMARHI

select customer_id, customer_name, customer_city from customer where branch_id
in(select branch_id from branch where branch_city = 'DELHI');

CUSTOMER_ID	CUSTOMER_NAME	CUSTOMER_CITY
DL19	RAKESH	DELHI
DL23	RAHUL	DELHI

Q.10)CREATE THE FOLLOWING TABLES:

BOOK(ACCESSION-NO,TITLE,PUBLISHER,YEAR,DATE-OF-PURCHASE,STATUS)

MEMBER(MEMBER-ID,NAME,NUMBER-OF-BOOKS-ISSUED,MAX-LIMIT)

BOOK-ISSUE(ACCESSION-NO,MEMBE-ID,DATE-OF-ISSUE)

GENERATE QUERIES TO DO THE FOLLOWING:

1)LIST ALL THOSE BOOKS WHICH ARE DUE FROM THE STUDENTS TO BE RETURNED.A BOOK IS CONSIDERED TO BE DUE IF IT HAS BEEN ISSUED 15 DAYS BACK AND YET NOT RETURNED.

2)LIST ALL THOSE MEMBERS WHO CANNOT BE ISSUED ANY MORE BOOKS.

SOLUTION 10.

```
create table book1(
accession_no varchar2(10),
title varchar2(20),
publisher varchar2(10),
dop date,
status varchar2(15));
```

```
create table member(
member_id varchar2(15),
name varchar2(15),
nob_issued number(3),
limit number(2));
```

```
create table books_issued(
accession_no varchar2(10),
member_id varchar(15),
doi date);
```

query:

```
select accession_no , title from book1 where accession_no in(select accession_no from
books_issued where (months_between(sysdate,doi)*30) > 15);
```

ACCESSION_NO	TITLE
IT002	SAD
CA05	WP
CA04	C

```
select * from member where nob_issued = limit;
```

MEMBER_ID	NAME	NOB_ISSUED	LIMIT
5	RAJESH	2	2
2	VIVEK	2	2

Q.14)CREATE THE FOLLOWING TABLES:

BRANCH(BRANCH-ID,BRANCH-NAME,CUSTOMER-CITY,BRANCH-ID)

CUSTOMER(CUSTOMER-ID,CUSTOMER-NAME,CUSTOMER-CITY,BRACH-ID)

GENERATE QUERIES TO DO THE FOLLOWING:

1)LIST ALL THOSE CUSTOMERS WHO LIVE IN THE SAME CITY AS THE BRANCH IN WHICH THEY HAVE ACCOUNT.

2)LIST ALL THOSE CUSTOMERS WHO HAVE AN ACCOUNT IN MORE THAN ONE BRANCH.

SOLUTION 14.

```
create table branch(
branch_id varchar2(10),
branch_name varchar2(20),
branch_city varchar(10));
```

```
create table customer(
customer_id varchar2(10),
customer_name varchar2(20),
customer_city varchar2(10),
branch_id varchar2(10));
```

query:

```
select A.customer_id, A.customer_name, B.branch_id, B.branch_city, A.customer_city
from customer A,
(select branch_id, branch_city from branch) B
where A.branch_id = B.branch_id and A.customer_city = B.branch_city;
```

CUSTOMER_ID	CUSTOMER_NAME	BRANCH_ID	BRANCH_CITY	CUSTOMER_CITY
G1	RAJU	G01	DELHI	DELHI
G2	RAMAN	G02	LUCKNOW	LUCKNOW
G3	RAJEEV	G03	NOIDA	NOIDA

15. CREATE THE FOLLOWING TABLES :

BRANCH (BRANCH-ID, BRANCH-NAME, CUSTOMER-CITY)

CUSTOMER (CUSTOMER-ID, CUSTOMER-NAME, CUSTOMER-CITY, BRANCH-ID)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) GENERATE QUERIES TO DO THE FOLLOWING :

(I) LIST ALL THOSE CUSTOMERS WHO HAVE MORE THAN 100 CUSTOMER.

(II) LIST ALL THOSE CUSTOMERS WHO HAVE AN ACCOUNT IN MORE THAN ONE BRANCH.

SOLUTION:

```
CREATE TABLE Branch2(
branch_id number(8) NOT NULL PRIMARY KEY,
branch_name varchar2(20),
customer_city varchar2(20));
```

```
-----
CREATE TABLE Cusotmer2(
customer_id number(8) NOT NULL PRIMARY KEY,
customer_name varchar2(20),
customer_city varchar2(20),
branch_id number(8),
constraint fk_branch2 foreign key(branch_id) references Branch1(branch_id));
```

(i)select customer_name,customer_id from Cusotmer2
where customer_id>(select count(customer_id) from Cusotmer2);

CUSTOMER_NAME	CUSTOMER_ID
Salaw	10
Sala	20
Qala	30

Q.16. CREATE THE FOLLOWING TABLE :

STUDENT (ROLL-NO, NAME, CATEGORY, DISTRICT, STATE)

STUDENT –RANK (ROLL-NO, MARKS, RANK)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) GENERATE QUERIES TO DO THE FOLLOWING :

(I) LIST NAMES OF THE STUDENTS WHO ARE HAVING SAME RANK BUT THEY SHOULD RESIDE IN DIFFERENT DISTRICTS.

(II) LIST DETAILS OF STUDENTS THEY BELONGS TO SAME CATEGORY WITH SAME RANK.

```
CREATE TABLE Student10(
roll_no number(8) NOT NULL PRIMARY KEY,
name varchar2(30), category char(1),
district varchar2(20), state varchar2(20));
```

```
-----
CREATE TABLE Student_rank10(
roll_no number(8), marks number(3),
rank number(3),
constraint fk_st foreign key(roll_no) references Student10(roll_no));
(ii)select * from Student s,Student_rank sr
where s.category=s.category AND sr.rank=s.rank;
```

ROLL_NO	NAME	CATEGORY	DISTRICT	STATE	ROLL_NO	MARKS	RANK
10	Goldn	A	Koya	UP	10	70	200
10	Goldn	A	Koya	UP	20	70	150
10	Goldn	A	Koya	UP	30	70	100
20	Gogo	A	Koya	UP	10	70	200

17. CREATE THE FOLLOWING TABLES :

STUDENT(ROLL-NO, NAME, DATE-OF-BIRTH, COURSE-ID)

COURSE (COURSE-ID, NAME, FEE, DURATION)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) GENERATE QUERIES TO DO THE FOLLOWING :

(I) LIST ALL THOSE STUDENTS WHO ARE BETWEEN 18-19 YEARS OF AGE AND HAVE OPTED FOR

MCA COURSE.

(II) LIST ALL THOSE COURSES IN WHICH NUMBER OF STUDENTS ARE LESS THAN 10.

```
CREATE TABLE Courses10(
course_id number(8) NOT NULL PRIMARY KEY,
name varchar2(20),
fee number(8,2),
duration varchar2(20));
```

```
-----
CREATE TABLE Student20(
roll_no number(8)primary key,
s_name varchar2(20),
date_of_birth date,
course_id number(8),
constraint fk_cour foreign key(course_id) references Courses10(course_id));
(i)select s_name from Student20 S,Courses10 C
where date_of_birth between '01/Jan/1992' AND '01/Jul/1992' AND name='MCA';
```

S_NAME
AJAY

1 rows returned in 0.00 seconds

[CSV Export](#)

1:::

```
select * from student
```

```
where age >18 AND age<35 and course_id in (select course_id from course where
name='mca');
```

2:::

```
select c.name from
```

```
course c, student s where c.course_id=s.course_id
group by c.name having count(*)>1
```

**Q.18) STUDENT(ROLL-NO, NAME, DATE-OF-BIRTH, COURSE-ID)
COURSE (COURSE-ID, NAME, FEE, DURATION, STATUS)**

**(A) WRITE PL/SQL PROCEDURE TO DO THE FOLLOWING :
SET THE STATUS OF COURSE TO "NOT OFFERED" IN WHICH THE
NUMBER OF CANDIDATES IS
LESS THAN 5.**

SOLUTION:

```
create table student (  
roll_no number(5) NOT NULL,  
name varchar2(20) NOT NULL,  
dob date,  
course_id varchar2(5) PRIMARY KEY  
);
```

```
insert into student values (101,'ADAM','24-OCT-1988','MCS');  
insert into student values (102,'JAMES','06-JAN-1985','MTC');  
insert into student values (102,'CHARLES','17-APR-1987','MCA');
```

```
create table course (  
course_id varchar2(5) references student(course_id),  
name varchar2(40),  
duration number(2),  
fee float(10),  
status varchar2(20)  
);
```

```
insert into course values ('MCS','M.Sc.-CS',2,40000,null);  
insert into course values ('MTC','M.Tech.-CS',2,120000,null);  
insert into course values ('MCA','M.C.A',3,100000,null);
```

```
declare  
v_status course.status%type:='Not Offered';  
cursor c19 is  
select course_id,count(*) from Student  
Group by course_id  
having count(*)<5;  
begin  
for i in c19 loop  
update course set status=v_status;  
end loop;  
end;
```

Results	Explain	Describe	Saved SQL	History
COURSE_ID	NAME	DURATION	FEE	STATUS
MCA	MASTER OF COMPUTER APP	3	-	Not Offered
MTB	M.TECH BIO-INFO	2	-	Not Offered
MCS	M.SC COMP.SC	2	-	Not Offered
BTECH	BACHELOR OF TECH	4	-	Not Offered

4 rows returned in 0.00 seconds [CSV Export](#)

Q.20) 20. CREATE THE FOLLOWING TABLES :

STUDENT(ROLL-NO, NAME, DATE-OF-BIRTH, COURSE-ID)

COURSE (COURSE-ID, NAME, FEE, DURATION, STATUS)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) WRITE PL/SQL PROCEDURE TO DO THE FOLLOWING :

SET THE STATUS OF COURSE TO "OFFERED" IN WHICH THE NUMBER OF CANDIDATES IS AT LEAST 10 OTHERWISE SET IT TO "NOT OFFERED".

SOLUTION:

```

DECLARE
V_STATUS1 COURSE.STATUS%TYPE:='OFFERED';
V_STATUS2 COURSE.STATUS%TYPE:='NOT OFFERED';
CURSOR C20A IS
SELECT COURSE_ID,COUNT(*) FROM STUDENT
GROUP BY COURSE_ID
HAVING COUNT(*)<10;
CURSOR C20B IS
SELECT COURSE_ID,COUNT(*) FROM STUDENT
GROUP BY COURSE_ID
HAVING COUNT(*)>10;
BEGIN
FOR I IN C20A LOOP
UPDATE COURSE SET STATUS=V_STATUS1;
END LOOP;
FOR I IN C20B LOOP
UPDATE COURSE SET STATUS=V_STATUS2;
END LOOP;
END;
```

Results Explain Describe Saved SQL History

COURSE_ID	NAME	DURATION	FEE	STATUS
MCA	MASTER OF COMPUTER APP	3	-	Offered
MTB	M.TECH BIO-INFO	2	-	Offered
MCS	M.SC COMP.SC	2	-	Offered
BTECH	BACHELOR OF TECH	4	-	Offered

1 rows returned in 0.00 seconds

[CSV Export](#)

Q.21) CREATE THE FOLLOWING TABLE :

ITEM (ITEM-CODE, ITEM-NAME, QTY-IN-STOCK, REORDER-LEVEL)

SUPPLIER (SUPPLIER-CODE, SUPPLIER-NAME, ADDRESS)

CAN-SUPPLY(SUPPLIER-CODE, ITEM-CODE)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) WRITE PL/SQL PROCEDURE TO DO THE FOLLOWING :

GENERATE A REPORT TO LIST THE ITEMS WHOSE QTY-IN-STOCK IS LESS THAN OR EQUAL TO THEIR REORDER-LEVELS.

SOLUTION:

```
CREATE TABLE ITEM(  
ITEM_CODE VARCHAR(10) PRIMARY KEY,  
ITEM_NAME VARCHAR(25),  
QTY_INHAND NUMBER(8),  
REORDER_LEVEL NUMBER(8));
```

```
CREATE TABLE SUPPLIER(  
SUPPLIER_CODE VARCHAR(10) PRIMARY KEY,  
SUPPLIER_NAME VARCHAR(25),  
ADDRESS VARCHAR(30));
```

```
CREATE TABLE CAN_SUPPLY(  
SUPPLIER_CODE VARCHAR2(10),  
ITEM_CODE VARCHAR2(10),  
PRIMARY KEY (SUPPLIER_CODE,ITEM_CODE));
```

```
INSERT INTO ITEM VALUES('ITM0001','BISCUIT',200,6);  
INSERT INTO ITEM VALUES('ITM0002','BREAD',165,4);  
INSERT INTO ITEM VALUES('ITM0003','TOOTH PASTE',16,2);  
INSERT INTO ITEM VALUES('ITM0004','COKE',10,1);
```

```
INSERT INTO SUPPLIER VALUES ('SC0001','PARLE','NEW DELHI');  
INSERT INTO SUPPLIER VALUES ('SC0002','PEPSI','MUMBAI');  
INSERT INTO SUPPLIER VALUES ('SC0003','AMUL','JALANDAR');
```

```
INSERT INTO CAN_SUPPLY VALUES('SC0001','ITM0001');
```

```

INSERT INTO CAN_SUPPLY VALUES('SC0001','ITM0002');
INSERT INTO CAN_SUPPLY VALUES('SC0002','ITM0004');
INSERT INTO CAN_SUPPLY VALUES('SC0003','ITM0002');

```

```

DECLARE
CURSOR C21 IS
SELECT QTY_INHAND,REORDER_LEVEL,ITEM_NAME FROM ITEM;
REPORT ITEM%ROWTYPE;
BEGIN
FOR I IN C21 LOOP
IF I.QTY_INHAND<=I.REORDER_LEVEL THEN
DBMS_OUTPUT.PUT_LINE('ITEMS WHOSE STOCK IS LESS OR EQUAL TO
THEIR REORDER LEVEL');
DBMS_OUTPUT.PUT_LINE(I.ITEM_NAME);
END IF;
END LOOP;
END;
/

```

Q.22) CREATE THE FOLLOWING TABLE :

ITEM (ITEM-CODE, ITEM-NAME, QTY-IN-STOCK, REORDER-LEVEL)

SUPPLIER (SUPPLIER-CODE, SUPPLIER-NAME, ADDRESS, STATUS)

CAN-SUPPLY(SUPPLIER-CODE, ITEM-CODE)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) WRITE PL/SQL PROCEDURE TO DO THE FOLLOWING :

SET THE STATUS OF THE SUPPLIER TO "IMPORTANT" IF THE SUPPLIER CAN SUPPLY MORE THAN FIVE ITEMS.

SOLUTION:

```

CREATE OR REPLACE FUNCTION STATUS_CHANGE(VS_CODE VARCHAR2)
RETURN NUMBER

```

```

AS

```

```

MYCOUNT NUMBER(4);

```

```

BEGIN

```

```

SELECT COUNT(*) INTO MYCOUNT FROM CAN_SUPPLY

```

```

WHERE SUPPLIER_CODE=VS_CODE;

```

```

RETURN MYCOUNT;

```

```

END;

```

```

DECLARE

```

```

CURSOR C21 IS

```

```

SELECT SUPPLIER_CODE INTO VS_CODE FROM SUPPLIER;

```

```

COUNT NUMBER(4);

```

```

BEGIN

```

```

FOR I IN C21 LOOP

```

```

COUNT=STATUS_CHANGE(VS_CODE NUMBER);

```

```

IF COUNT >= 5 THEN
UPDATE SUPPLIER SET STATUS='IMPORTANT';
END IF;
END LOOP;
END;

```

23. CREATE THE FOLLOWING TABLES :

ITEM (ITEM-CODE, ITEM-NAME, QTY-IN-STOCK, REORDER-LEVEL)

SUPPLIER (SUPPLIER-CODE, SUPPLIER-NAME, ADDRESS, STATUS)

CAN-SUPPLY(SUPPLIER-CODE, ITEM-CODE)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) WRITE PL/SQL PROCEDURE TO DO THE FOLLOWING :

GENERATE A REPORT OF THOSE ITEMS THAT ARE SUPPLIED BY THOSE SUPPLIERS WHOSE STATUS IS "IMPORTANT".

SOLUTION:

```

create or replace function report(vs_code varchar2)
return varchar2
as
mylist varchar2(25);
begin
select item_code into mylist
from Can_supply
where supplier_code=vs_code;
return mylist;
end;
/

```

```

declare
vs_code supplier.supplier_code%type;
cursor c23 is
select supplier_code into vs_code from supplier
where status='important';
myitem varchar2(25);
begin
for i in c23 loop
if i.status='important' then
vs_code:=i.supplier_code;
myitem:=report(vs_code);
select item_code, item_name
from item
where item_code=myitem;
dbms_output.put_line(item_code || 'is ' || item_name);
end if;

```

```
end loop;  
end;
```

Q. 24. CREATE THE FOLLOWING TABLES :

STUDENT (ROLL-NO, NAME, CATEGORY, DISTRICT, STATE)

STUDENT –RANK (ROLL-NO, MARKS, RANK)

WRITE PL/SQL PROCEDURE TO THE FOLLOWING :

GENERATE A REPORT TO LIST OF THOSE DISTRICTS FROM WHICH THE FIRST HUNDRED RANKERS COME FROM.

SOLUTION:

```
create table student3(  
roll_no varchar2(10),  
marks number(10),  
category varchar2(10),  
district varchar2(10),  
state varchar2(10));
```

```
create table student_rank(  
roll_no varchar2(10),  
marks number(10),  
rank number(5));
```

```
insert into student3 values (101,453,'GC','Rampur','UP');  
insert into student3 values (102,389,'GC','Agra','UP');  
insert into student3 values (103,422,'RB','Agartala','UP');  
insert into student3 values (104,410,'RB','Ganganagar','Rajasthan');  
insert into student3 values (105,413,'SC','Silampur','Rajasthan');
```

```
insert into student_rank values(101,453,56);  
insert into student_rank values(102,389,113);  
insert into student_rank values(103,422,68);  
insert into student_rank values(104,413,91);  
insert into student_rank values(105,410,92);
```

```
create or replace procedure dist_list  
as  
cursor c24 is  
select * from student3 join student_rank on student3.roll_no=student_rank.roll_no;  
begin  
for i in c24 loop  
if i.rank<=100 then  
dbms_output.put_line(i.district);  
end if;
```

```
end loop;  
end;  
/
```

Q.25). CREATE THE FOLLOWING TABLES :

STUDENT (ROLL-NO, NAME, SUBJECT-OPTED)

SUBJECT –RANK (SUBJECT-CODE, SUBJECT-NAME, FACULTY-CODE, SPECIALIZATION)

FACULTY (FACULTY-CODE, FACULTY-NAME, SPECIALIZATION)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) WRITE PL/SQL PROCEDURE TO THE FOLLOWING :

SET THE STATUS OF THE SUBJECT TO "NOT OFFERED" IF THE SUBJECT IS NOT OPTED BY AT LEAST 5 STUDENTS.

SOLUTION:

```
CREATE TABLE STUDENT4(  
  ROLL_NO VARCHAR2(10),  
  STUDENT_NAME VARCHAR2(25),  
  SUBJECT_OPTED VARCHAR2(10)  
);
```

```
CREATE TABLE FACULTY4  
(  
  FACULTY_CODE VARCHAR2(15),  
  FACULTY_NAME VARCHAR(20),  
  SPECIALIZATION VARCHAR2(10)  
);
```

```
CREATE TABLE SUBJECT4(  
  SUBJECT_NAME VARCHAR2(20),  
  FACULTY_CODE VARCHAR2(20),  
  SPECIALIZATION VARCHAR2(10)  
);
```

```
INSERT INTO STUDENT4 VALUES(
```

```
  CREATE OR REPLACE FUNCTION STATUS_CH  
  RETURN NUMBER  
  AS  
  MYCOUNT NUMBER(4);  
  OPTED VARCHAR2(10);  
  BEGIN
```

```

SELECT SUBJECT_OPTED,COUNT(*) INTO OPTED,MYCOUNT FROM
STUDENT4
GROUP BY SUBJECT_OPTED;
RETURN MYCOUNT;
END;

```

```

DECLARE
CURSOR C25 IS
SELECT STATUS FROM SUBJECT4
FOR UPDATE;
COUNTS NUMBER(4);
BEGIN
FOR I IN C25 LOOP
COUNTS:=STATUS_CH;
IF COUNTS<5 THEN
UPDATE SUBJECT4 SET STATUS='NOT OFFERED'
WHERE CURRENT OF C25;
END IF;
END LOOP;
END;

```

Q.26) CREATE THE FOLLOWING TABLES :

STUDENT (ROLL-NO, NAME, SUBJECT-OPTED)

SUBJECT –RANK (SUBJECT-CODE, SUBJECT-NAME, FACULTY-CODE, SPECIALIZATION)

FACULTY (FACULTY-CODE, FACULTY-NAME, SPECIALIZATION)

(A) CREATE A FORM TO ACCEPT THE DATA FROM THE USER WITH APPROPRIATE VALIDATION CHECKS.

(B) WRITE PL/SQL PROCEDURE TO THE FOLLOWING :

SET THE STATUS OF THE SUBJECT TO "NOT OFFERED" IF THE SUBJECT IS NOT OFFERED BY ANY OF THE FACULTY MEMBERS.

SOLUTION:

```

create or replace procedure chk_spl
as
cursor c26
is
select * from subject4
for update;
begin
for value in c26 loop
if value.specialization="" then
update subject4 set status='not offered'

```



```

where current of c26;
enf if;
end loop;
end;

```

Q.27) CREATE THE FOLLOWING TABLES :

STUDENT (ROLL-NO, NAME, SUBJECT-OPTED)

SUBJECT –RANK (SUBJECT-CODE, SUBJECT-NAME, FACULTY-CODE)

FACULTY (FACULTY-CODE, FACULTY-NAME, SPECIALIZATION)

GENERATE QUERIES TO DO THE FOLLOWING :

(I) FIND THE NUMBER OF STUDENTS WHO HAVE ENROLLED FOR THE SUBJECT "DBMS"

(II) FIND ALL THOSE SUBJECTS WHICH ARE NOT OFFERED BY ANY FACULTY MEMBERS.

SOLUTION:

27-

-----create table-----

1- table student :

```

create table student ( roll_no number(5) PRIMARY
KEY ,
student_name varchar2(15),
subject_opted varchar2(15)
);

```

ROLL_NO	STUDENT_NAME	SUBJECT_OPTED
204	ali ahmad	DBMS
205	asam khan	DBMS
201	ali ahmad	c++
202	Aso	java
203	hassan	network

2-table Faculty

```

create table faculty ( faculty_cod number(5)
PRIMARY KEY ,
faculty varchar2(15),
specialization varchar2(25)
);

```

FACULTY_COD	FACULTY	SPECIALIZATION
101	computer	assignment to comp.
102	farmacy	assignment to Farm.
103	Engne.	assignment to engineer.

3-table subject_rank

```

create table subject_rank(subject_code number(4)
PRIMARY KEY,
subject_name varchar2(44),
faculty_cod number(5) REFERENCES
faculty(faculty_cod)
);

```

-----insert data into table-----

SUBJECT_CODE	SUBJECT_NAME	FACULTY_COD
301	C++	101
304	software engineer	103

```

1-table student
insert into student
(roll_no,student_name,subject_opted)
values(204,'ali ahmad','DBMS');
insert into student
(roll_no,student_name,subject_opted)
values(205,'asam khan','DBMS');
insert into student
(roll_no,student_name,subject_opted)
values(201,'ali ahmad','c++');
insert into student
(roll_no,student_name,subject_opted)
values(202,'Aso','java');
insert into student
(roll_no,student_name,subject_opted)
values(203,'hassan','network');
table faculty
insert into faculty(faculty_cod,faculty,specialization)
values(101,'computer','assignment to comp. ');
insert into faculty(faculty_cod,faculty,specialization)
values(102,'farmacy','assignment to Farm. ');
insert into faculty(faculty_cod,faculty,specialization)
values(103,'Engne.','assignment to engineer. ');

```

```

insert into
subject_rank(subject_code,subject_name,faculty_cod)
values(301,'C++',101);

```

```

insert into
subject_rank(subject_code,subject_name,faculty_cod)
values(304,'software engineer',103);

```

```

I) select count(roll_no) from student
where subject_opted='DBMS';

```

OR

```

select student_name from student
where subject_opted='DBMS'

```

II)? it is not true.

```

select faculty from faculty f
where f.faculty_cod not in (select s.faculty_cod
from subject_rank s);

```

COUNT(ROLL_NO)
2

OR

STUDENT_NAME
ali ahmad
asam khan

FACULTY
farmacy

Q.28)CREATE THE FOLLOWING TABLES :

STUDENT (ROLL-NO, NAME, SUBJECT-OPTED)

SUBJECT –RANK (SUBJECT-CODE, SUBJECT-NAME, FACULTY-CODE)

FACULTY (FACULTY-CODE, FACULTY-NAME, SPECIALIZATION)

GENERATE QUERIES TO DO THE FOLLOWING :

(I) FIND THE NUMBER OF STUDENTS WHO HAVE ENROLLED FOR THE SUBJECT "DBMS"

(II) FIND ALL THOSE SUBJECTS WHICH ARE OFFERED BY MORE THAN ONE FACULTY MEMBER.

SOLUTION:

28) -----create table-----

1- table student :

```
create table student ( roll_no number(5) PRIMARY
KEY ,
student_name varchar2(15),
subject_opted varchar2(15)
);
```

2-table Faculty

```
create table faculty ( faculty_cod number(5)
PRIMARY KEY ,
faculty varchar2(15),
specialization varchar2(25)
);
```

3-table subject_rank

```
create table subject_rank(subject_code
number(4)PRIMARY KEY ,
subject_name varchar2(44),
faculty_cod number(5) REFERENCES
faculty(faculty_cod)
);
```

-----insert data into table-----

1-table student

```
insert into student
(roll_no,student_name,subject_opted)
values(204,'ali ahmad','DBMS');
insert into student
(roll_no,student_name,subject_opted)
values(205,'asam khan','DBMS');
insert into student
(roll_no,student_name,subject_opted)
values(201,'ali ahmad','c++');
insert into student
(roll_no,student_name,subject_opted)
values(202,'Aso','java');
insert into student
(roll_no,student_name,subject_opted)
values(203,'hassan','network');
```

3-table subject

```
insert into subject_rank(subject_code,subject_name,
faculty_cod)values(301,'C++',101);
insert into subject_rank(subject_code,subject_name,
faculty_cod)values(304,'software engineer',103);
insert into subject_rank(subject_code,subject_name,
faculty_cod)values(305,'Adv.java',101);
```

I) select student_name from student
where subject_opted !='DBMS';
II)??!

ROLL_NO	STUDENT_NAME	SUBJECT_OPTED
204	ali ahmad	DBMS
205	asam khan	DBMS
201	ali ahmad	c++
202	Aso	java
203	hassan	network

FACULTY_COD	FACULTY	SPECIALIZATION
101	computer	assignment to comp.
102	farmacy	assignment to Farm.
103	Engne.	assignment to engineer.

SUBJECT_CODE	SUBJECT_NAME	FACULTY_COD
301	C++	101
304	software engineer	103
305	Adv.java	101
306	Adv.OS	101

STUDENT_NAME
ali ahmad
Aso
hassan

Q.29) Create the following tables:

Student (roll-no, name, subject-opted)

Student-rank (subject-code, subject-name, faculty-code)

Faculty (faculty-code, faculty-name, specialization)

Generate queries to do the following:

- (i) Find the number of students who have enrolled for the subjects “OS”**
- (ii) Find all those students who opted for more than 5 subjects.**

29)

-----create table-----

1- table student :

```
create table student ( roll_no number(5),
student_name varchar2(15),
subject_opted varchar2(15)
);
```

2-table Faculty

```
create table faculty ( faculty_cod number(5) PRIMARY KEY ,
faculty varchar2(15),
specialization varchar2(25)
);
```

3-table subject_rank

```
create table subject_rank(subject_code number(4) PRIMARY KEY,
subject_name varchar2(44),
faculty_cod number(5) REFERENCES faculty(faculty_cod)
);
```

ROLL_NO	STUDENT_NAME	SUBJECT_OPTED
201	ali ahmad	DBMS
201	ali ahmad	OS
201	ali ahmad	C++
201	ali ahmad	java
201	ali ahmad	Network
201	ali ahmad	Graphic
205	asam khan	DBMS
201	ali ahmad	c++
202	Aso	java
203	hassan	network

-----insert data into table-----

1-table student

```
insert into student (roll_no,student_name,subject_opted)
values(201,'ali ahmad','Graphic');
```

```
insert into student (roll_no,student_name,subject_opted)
values(205,'asam khan','DBMS');
```

```
insert into student (roll_no,student_name,subject_opted)
values(201,'ali ahmad','c++');
```

```
insert into student (roll_no,student_name,subject_opted)
values(202,'Aso','java');
```

```
insert into student (roll_no,student_name,subject_opted)
values(203,'hassan','network');
```

```
select * from student;
```

Q.30.) CREATE THE FOLLOWING TABLES :

STUDENT (ROLL-NO, NAME, SUBJECT-OPTED)

SUBJECT –RANK (SUBJECT-CODE, SUBJECT-NAME, FACULTY-CODE)

FACULTY (FACULTY-CODE, FACULTY-NAME, SPECIALIZATION)

GENERATE QUERIES TO DO THE FOLLOWING :

(I) FIND THE NUMBER OF STUDENTS WHO HAVE NOT ENROLLED FOR THE SUBJECT "DBMS"

(II) FIND ALL THOSE SUBJECTS WHICH ARE OFFERED BY MORE THAN ONE FACULTY MEMBER

SOLUTION:

30) -----create table-----

1- table student :

```
create table student ( roll_no number(5) PRIMARY
KEY ,
student_name varchar2(15),
subject_opted varchar2(15)
);
```

ROLL_NO	STUDENT_NAME	SUBJECT_OPTED
204	ali ahmad	DBMS
205	asam khan	DBMS
201	ali ahmad	c++
202	Aso	java
203	hassan	network

2-table Faculty

```
create table faculty ( faculty_cod number(5)
PRIMARY KEY ,
faculty varchar2(15),
specialization varchar2(25)
);
```

FACULTY_COD	FACULTY	SPECIALIZATION
101	computer	assignment to comp.
102	farmacy	assignment to Farm.
103	Engne.	assignment to engineer.

3-table subject_rank

```
create table subject_rank(subject_code
number(4)PRIMARY KEY ,
subject_name varchar2(44),
faculty_cod number(5) REFERENCES
faculty(faculty_cod)
);
```

SUBJECT_CODE	SUBJECT_NAME	FACULTY_COD
301	C++	101
304	software engineer	103
305	Adv.java	101
306	Adv.OS	101

-----insert data into table-----

1-table student

```
insert into student
(roll_no,student_name,subject_opted)
values(204,'ali ahmad','DBMS');
insert into student
(roll_no,student_name,subject_opted)
values(205,'asam khan','DBMS');
insert into student
(roll_no,student_name,subject_opted)
values(201,'ali ahmad','c++');
insert into student
(roll_no,student_name,subject_opted)
values(202,'Aso','java');
insert into student
(roll_no,student_name,subject_opted)
values(203,'hassan','network');
```

3-table subject

```
insert into subject_rank(subject_code,subject_name,
faculty_cod)values(301,'C++',101);
insert into subject_rank(subject_code,subject_name,
faculty_cod)values(304,'software engineer',103);
insert into subject_rank(subject_code,subject_name,
faculty_cod)values(305,'Adv.java',101);
```


Q.31)A HOSPITAL MAITAINS BLOOD DONORS RECORD INFORMATION IN A FILE.THE ITEMS ARE DONOR NUMBER,DONOR NAME,DONOR AGE,DONOR ADDRESS,PIN,PLACE OF BIRTH,BLOOD GROUP(A,B,AB &O) WRITE A PROGRAM TO PRINT OUT THE NUMBER,NAME AND ADDRESS OF THE DONORS FOR THE FOLLOWING CATEGORIESL
A)BLOOD DONOR HAVING THE BLOOD GROUP AB.
B)BLODD DONOR IN THE AGE GROUP BETWEEN 16-25.
C)FEMALE DONORS HAVING BLODD GROUP A IN THE AGE BETWEEN 20-25.

SOLUTION:

31- -----create table-----

```
create table blood_donor( donor_no number(4) PRIMARY KEY,
donor_name varchar2(10),
donor_age number(5),
donor_address varchar2(17),
place_birth varchar(12),
blood_group varchar(12)
);
```

-----insert table-----

```
insert into blood_donor values (1,'ali',23,'delhi','11-12-1987','A');
insert into blood_donor values (2,'jack',24,'delhi','11-12-1988','b');
insert into blood_donor values (3,'micle',15,'nework','11-12-1994','AB');
insert into blood_donor values (4,'ASo',16,'sul','11-12-1991','O');
insert into blood_donor values (5,'AZAd',20,'hawler','11-12-1999','A');
insert into blood_donor values (6,'KURDUSTAN',22,'istanbul','11-12-1997','O');
insert into blood_donor values (7,'KURDA',25,'puna','11-12-1990','AB');
```

DONOR_NO	DONOR_NAME	DONOR_AGE	DONOR_ADDRESS	PLACE_BIRTH	BLOOD_GROUP
1	ali	23	delhi	11-12-1987	A
2	jack	24	delhi	11-12-1988	b
3	micle	15	nework	11-12-1994	AB
4	ASo	16	sul	11-12-1991	O
5	AZAd	20	hawler	11-12-1999	A
6	KURDUSTAN	22	istanbul	11-12-1997	O
7	KURDA	25	puna	11-12-1990	AB

a)select donor_no,donor_name,donor_address
from blood_donor
where blood_group='AB';

DONOR_NO	DONOR_NAME	DONOR_ADDRESS
3	micle	nework
7	KURDA	puna

b)select donor_no,donor_name,donor_address
from blood_donor
where donor_age between 16 and 25 ;

DONOR_NO	DONOR_NAME	DONOR_ADDRESS
1	ali	delhi
2	jack	delhi
4	ASo	sul
5	AZAd	hawler
6	KURDUSTAN	istanbul
7	KURDA	puna

c)select donor_no,donor_name,donor_address
from blood_donor
where donor_age between 16 and 25 and blood_group='A';

DONOR_NO	DONOR_NAME	DONOR_ADDRESS
1	ali	delhi
5	AZAd	hawler


```

values(3,'hassan','2000',5);
insert into sales_person
values(4,'hussain','2003',7);
insert into sales_person
values(5,'aso','2006',3);
insert into sales_person
values(6,'ali','2007',2);

```

```

2-trip
insert into trip
values(110,'Delhi','puna','1-11-2003','3-
11-20003',1);
insert into trip
values( 111,'Delhi','hayderabad','1-11-
2003','3-11-20003', 2);
insert into trip values(112 ,'Delhi','UP
','1-11-2003','3-11-20003',3 );
insert into trip
values( 113,'Delhi','Banglore ','1-11-
2003','3-11-20003',4 );
insert into trip
values( 114,'Delhi','alahabad ','1-11-
2003','3-11-20003', 5);
insert into trip
values(115 ,'Delhi','kshmir ','1-11-
2003','3-11-20003',6 );

```

```

3-expense
insert into expense
values(112,500,110);
insert into expense
values(113,600,110);
insert into expense
values(112,5000,112);
insert into expense
values(112,500,110);
insert into expense
values(115,500,113);

```

```

1- select p.ssn
from sales_person p,trip t
where p.ssn=t.ssn and t.to_city='puna'

```

```

1- select p.ssn
from sales_person p,trip t
where p.ssn=t.ssn and t.to_city='puna'

```

SSN
1

```

2- select sum(amount)
from trip t,expense e
where t.trip_id=e.trip_id and
e.trip_id= 110
      or
select sum(amount)
from trip t,expense e
where t.trip_id=e.trip_id and t.ssn= 4;

```

SUM(AMOUNT)
2100

OR

SUM(AMOUNT)
500

Q.33)AN EXAMINATION HAS BEEN CONDUCTED TO A CLASS OF 7 STUDENTS AND FOUR SCORES FOR EACH STUDENT HAVE BEEN PROVIDED IN THE DATA ALONG WITH REGISTER NUMBER,NAME,WRITE SQL PROGRAM TO DO THE FOLLOWING:

ASSIGN A LETTER GRADE TO EACH STUDENT BASED IN THE AVERAGE SCORE AND LIST OUT THE STDENTS REGISTER NUMBER AND AVERAGE SCORE,GRADE.THE MINIMUM PASS FOR EACH SUBJECT IS 50;

THE GRADING SYSTM :
AVERAGE SCORE GRADE

90-100 ->A

75-89 -> B

60-74 -> C

50-59 -> D

0-49 -> F(FAIL)

SOLUTION:

```

33)
create table student1( register_no number(5)
PRIMARY KEY, student_name varchar2(44),mark1
number(5),
      mark2 number(5), grade varchar2(5)
);

```

REGISTER_NO	STUDENT_NAME	MARK1	MARK2	GRADE
110	ali	60	70	-
111	aso	50	55	-
112	othman	90	70	-
113	hassan	60	90	-
114	husain	70	70	-
115	akbar	40	70	-

```

update student1
set grade='A'
where ((mark1+mark2)/2) between 90 and 100 ;
update student1
set grade='B'
where ((mark1+mark2)/2) between 75 and 89 ;
update student1
set grade='C'
where ((mark1+mark2)/2) between 60 and 74 ;
update student1
set grade='D'
where ((mark1+mark2)/2) between 50 and 59 ;
update student1
set grade='Fail'
where ((mark1+mark2)/2) between 0 and 49 ;

```

REGISTER_ID	STUDENT_NAME	MARK1	MARK2	GRADE
110	ali	60	70	C
111	aso	50	55	D
112	othman	90	70	B
113	hassan	60	90	B
114	husain	70	70	C
115	akbar	40	70	D

Q.35)WRITE A PL/SQL PROGRAM TO IMPLEMENT THE FOLLOWING EXCEPTIONS:

- A)TOO_MANY_ROWS**
- B)DVD_VAL_ON_INDEX**

SOLUTION:

```

create table
customers(
  cus_id
  number(5)
  PRIMARY
  KEY,
  f_name
  varchar2(44),
  l_name
  varchar2(33)

```

```

);
-----program-----
a)
BEGIN
  INSERT INTO customers(
    cus_id , f_name , l_name
  ) VALUES (

```

Results	Explain	Describe	Saved SQL	History
---------	---------	----------	-----------	---------

Conversion of string to number failed

1 row(s) inserted.

```

    '123X', 'Greg', 'Green'
);
EXCEPTION
    WHEN INVALID_NUMBER THEN
        DBMS_OUTPUT.PUT_LINE('Conversion of string to number failed');
END;
B)
create table Employee(
    ID          VARCHAR2(4 BYTE)      NOT NULL primary key,
    First_Name   VARCHAR2(10 BYTE),
    Last_Name    VARCHAR2(10 BYTE),
    Start_Date   DATE,
    End_Date     DATE,
    Salary       Number(8,2),
    City         VARCHAR2(10 BYTE),
    Description   VARCHAR2(15 BYTE)
);
insert into Employee(ID,First_Name,Last_Name,
Start_Date,End_Date,Salary,City,Description)
    values ('01','Jason','Martin', to_date('19960725','YYYYMMDD'),
to_date('20060725','YYYYMMDD'), 1234.56, 'Toronto', 'Programmer')
insert into Employee(ID, First_Name, Last_Name,
Start_Date,End_Date,Salary,City,Description)
    values('02','Alison','Mathews', to_date('19760321','YYYYMMDD'),
to_date('19860221','YYYYMMDD'), 6661.78, 'Vancouver','Tester')

insert into Employee(ID, First_Name, Last_Name,
Start_Date,End_Date,Salary,City,Description)
    values('03','James','Smith',to_date('19781212','YYYYMMDD'),
to_date('19900315','YYYYMMDD'), 6544.78, 'Vancouver','Tester')

insert into Employee(ID, First_Name, Last_Name,
Start_Date,End_Date,Salary,City,Description)
    values('04','Celia','Rice',to_date('19821024','YYYYMMDD'),
to_date('19990421','YYYYMMDD'), 2344.78, 'Vancouver','Manager')

insert into Employee(ID, First_Name, Last_Name,
Start_Date,End_Date,Salary,City,Description)
    values('05','Robert', 'Black', to_date('19840115','YYYYMMDD'),
to_date('19980808','YYYYMMDD'), 2334.78, 'Vancouver','Tester')
insert into Employee(ID, First_Name, Last_Name,
Start_Date,End_Date,Salary,City,Description)
    values('05','Robert','Black',to_date('19840115','YYYYMMDD'),
to_date('19980808','YYYYMMDD'), 2334.78, 'Vancouver','Tester')

insert into Employee(ID, First_Name, Last_Name, Start_Date,End_Date,Salary,
City,Description)
    values('06','Linda','Green',to_date('19870730','YYYYMMDD'),

```

```

to_date('19960104','YYYYMMDD'), 4322.78,'New York', 'Tester')

insert into Employee(ID, First_Name, Last_Name, Start_Date,End_Date,Salary,
City,Description)
        values('07','David','Larry',to_date('19901231','YYYYMMDD'),to_date('
19980212','YYYYMMDD'), 7897.78,'New York', 'Manager')

insert into Employee(ID, First_Name, Last_Name, Start_Date,End_Date,Salary,
City,Description)
        values('08','James','Cat',to_date('19960917','YYYYMMDD'),
to_date('20020415','YYYYMMDD'), 1232.78,'Vancouver', 'Tester')

select * from Employee
BEGIN
    INSERT INTO employee (id)VALUES(1);
EXCEPTION
    WHEN DUP_VAL_ON_INDEX THEN
        DBMS_OUTPUT.PUT_LINE('Duplicate value on an index');
END;
```

Results	Explain	Describe	Saved SQL	History
Duplicate value on an index				
1 row(s) inserted.				

**Q.36)WRITE A PROGRAM IN SQL FOR TELEPHONE BILLING SYSTEM WITH THE FOLLOWING SYSTEMS.
CUSTOMER NUMBER,NAME,ADDRESS,OPENING READING,CLOSING READING WITH THE FOLLOWING CONDITIONS:**

- A)READING 0-100 FREE.**
- B)READING 101-TO 200 PER CALL CHARGE RS.1**
- C)READING 201 TO 300 PER CALL CHARGE RS.2**
- D)READING 301-400 PER CALL CHARGE RS.3**
- E)READING >401 PER CALL CHARGE RS.5;**

PREPARE A TELEPHONE BILL REPORT.

SOLUTION:

-----create table-----

```
create table telph_bill(  
  c_no number(5) PRIMARY KEY,  
  c_name varchar2(15),  
  c_address varchar2(8),  
  open_read number(5),  
  close_read number(5),  
  total number(8)  
);
```

C_NO	C_NAME	C_ADDRESS	OPEN_READ	CLOSE_READ	TOTAL
110	ali	delhi	100	500	1
111	aso	iraq	200	600	1
112	hassan	hawler	150	300	1
113	hussain	sna	250	500	1
114	sara	puna	250	600	1
115	sam	puna	100	900	1
116	sara2	puna	50	70	1

-----insert table -----

```
insert into telph_bill values (110,'ali','delhi',100,500,1);  
insert into telph_bill values (111,'aso','Iraq',200,600,1);
```

```
insert into telph_bill values (112,'hassan','hawler',150,300,1);  
insert into telph_bill values (113,'hussain','sna',250,500,1);  
insert into telph_bill values (114,'sara','puna',250,600,1);  
insert into telph_bill values (115,'sam','puna',100,900,1);  
insert into telph_bill values (116,'sara2','puna',50,70,1);
```

a)
update telph_bill
set total=(close_read-open_read)*0
where (close_read-open_read)between 0 and 100 ;

b)
update telph_bill
set total=(close_read-open_read)*1
where (close_read-open_read)between 101 and 200 ;

c)
update telph_bill
set total=(close_read-open_read)*2
where (close_read-open_read)between 201 and 300 ;

d)
update telph_bill
set total=(close_read-open_read)*3
where (close_read-open_read)between 301 and 400 ;
e)

```

update telph_bill
set total=(close_read-open_read)*5
where (close_read-open_read)>400;

```

C_HO	C_NAME	C_ADDRESS	OPEN_READ	CLOSE_READ	TOTAL
110	ali	delhi	100	500	1200
111	aso	iraq	200	600	1200
112	hassan	hawler	150	300	150
113	hussain	sna	250	500	500
114	sara	puna	250	600	1050
115	sam	puna	100	900	4000
116	sara2	puna	50	70	0

Q.37) A SALARY STATEMENT CONTAINS NAME,BASIC PAY,ALLOW,TOTAL DEDUCTION(INCLUDES IT),G PAY, N PAY

ALLOWANCE=58% OF BASIC PAY;

GROSS PAY=BASIC PAY + ALLOWANCE

DEDUCTION=P F+LOAN AMOUNT

IT IS CALCULATED ON THE BASIS OF ANNUAL INCOME UNDER THE FOLLOWING CONDITION.

ANNUAL SALARY INCOME TAX AMOUNT

<=1 LAKH NIL

>1 LAKH BUT <=1.5 LAKH 10%

>1.5 LAKH BUT <= 3 LAKH 20%

>3 LAKH AND ABOVE 30%

TOTAL DEDUCTION=DEDUCTION+INCOME TAX

WRITE A PROGRAM IN SQL TO PREPARE SALARY REPORT FOR 25 EMPLOYEES.

SOLUTION:

```

UPDATE SALARY S SET ALLOW=58/100*S.BASIC_PAY;

```

```

UPDATE SALARY S SET GPAY=S.ALLOW+S.BASIC_PAY;

```

```

UPDATE SALARY S SET TOTAL_DEDUCTION=10/100*BASIC_PAY;

```

FOR INCOME TAX

```
UPDATE SALARY S SET IT=0 WHERE BASIC_PAY<100000;
```

```
UPDATE SALARY S SET IT=10/100*(BASIC_PAY)
WHERE BASIC_PAY>100000 OR BASIC_PAY<150000;
```

```
UPDATE SALARY S SET IT=20/100*(BASIC_PAY)
WHERE BASIC_PAY>=150000 AND BASIC_PAY<300000 ;
```

```
UPDATE SALARY S SET IT=30/100*(BASIC_PAY)
WHERE BASIC_PAY>=300000;
```

```
UPDATE SALARY S SET NPAY=BASIC_PAY
+ALLOW-TOTAL_DEDUCTION+GROSS_PAY;
```

Q.38)WRITE A PL/SQL PROGRAM TO IMPLEMENT THE EXCEPTIONS:

SOLUTION:

38)

```
DECLARE
    e_TooManyEmployee EXCEPTION; -- Exception to indicate an error condition
BEGIN
    RAISE e_TooManyEmployee;
EXCEPTION
    WHEN e_TooManyEmployee THEN
        DBMS_OUTPUT.put_line('e_TooManyEmployee');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('OTHERS');
END;
```

Results	Explain	Describe	Saved SQL	History
e_TooManyEmployee				
Statement processed.				

Q.43)CONSIDER THE FOLLOWING RELATIONS:

SUPPLIER(S-ID:INTEGER,P-NAME:STRING,ADDRESS:STRING)

PARTS(P-ID:INTEGER,P-NAME:STRING,COLOR:STRING)

CATALOUES(S-ID:INTEGER,P-ID:INTEGER,COST:REAL)

CREATE TABLES FOR THE ABOVE RELATIONS.WRITE SQL FOR THE FOLLOWING:

A)FIND THE NAME OF SUPPLIERS WHO SUPPLY SAME RED PART.

B)FIND THE P-IDS OF PARTS THAT ARE SUPPLIED BY ATLEAST TWO DIFFERENT SUPPLIERS.

C)FIND THE S-IDS OF SUPPLIERS WHO SUPPLY SAME RED PART AND GREEN PARTS

D)FIND THE S-IDS OF SUPPLIERS WHO SUPPLY EVERY PART.

SOLUTION:

```
create table supplier(s_id number(4)PRIMARY
KEY,
s_name varchar2(20),
address varchar2(20)
);
```

```
create table parts(p_id number(4)PRIMARY
KEY,
p_name varchar2(20),
colour varchar2(20)
);
```

```
create table catalo(s_id number(4)
REFERENCES supplier(s_id),
p_id number(4) REFERENCES
parts(p_id),
cost varchar2(20)
);
```

```
insert into
supplier(s_id,s_name,address)values(1,'ali','delhi'
);
insert into
supplier(s_id,s_name,address)values(2,'aso','puna
');
insert into
supplier(s_id,s_name,address)values(3,'adaz','del
hi');
insert into
supplier(s_id,s_name,address)values(4,'alias','ne
w delhi');
insert into
parts(p_id,p_name,colour)values(1,'com','red');
```

S_ID	S_NAME	ADDRESS
1	ali	delhi
2	aso	puna
3	adaz	delhi
4	alias	new delhi

P_ID	P_NAME	COLOUR
1	com	red
2	com	red
3	phisc	red
4	com	green
5	hist	green

S_ID	P_ID	COST
1	1	1000\$
1	2	1000\$
2	3	1000\$
3	3	1500\$
2	2	1100\$

```

insert into
parts(p_id,p_name,colour)values(2,'com','red');
insert into
parts(p_id,p_name,colour)values(3,'phisc','red');
insert into
parts(p_id,p_name,colour)values(4,'com','green');
insert into
parts(p_id,p_name,colour)values(5,'hist','green');
insert into
catalo(s_id,p_id,cost)values(1,1,'1000$');
insert into
catalo(s_id,p_id,cost)values(1,2,'1000$');
insert into
catalo(s_id,p_id,cost)values(2,3,'1000$');
insert into
catalo(s_id,p_id,cost)values(3,3,'1500$');
insert into
catalo(s_id,p_id,cost)values(2,2,'1100$');

```

A)

```

select s.s_name
from supplier s,parts p,catalo c
where s.s_id=c.s_id and c.p_id =p.p_id and
colour='red';

```

S_NAME
ali
ali
aso
aso
adaz

b)

```

select p.p_id
from supplier s,parts p,catalo c
where s.s_id=c.s_id and c.p_id =p.p_id and
s.s_id>=2;

```

P_ID
3
3
2

C)

```

select s.s_id
from supplier s,parts p,catalo c
where s.s_id=c.s_id and c.p_id =p.p_id and
p.colour='red' and p.colour='green';

```

S_ID
1
3
2

D)

I am not sure.

```

select s.s_id
from supplier s,parts p,catalo c
where s.s_id=c.s_id and c.p_id =p.p_id and
s.s_id =all(p.p_id)

```