B.M.S COLLEGE OF ENGINEERING BENGALURU
Autonomous Institute, Affiliated to VTU

LAB REPORT

DATABASE MANAGEMENT SYSTEMS

Submitted in partial fulfillment for the award of degree of

Bachelor of Technology
in
Computer Science and Engineering

Submitted by:
VIVEK RAJEEV
1BM18CS142

Work carried out at

Internal Guide

Nandini Vineeth
Assistant Professor
BMSCE

Department of Computer Science and Engineering
B.M.S College of Engineering
Bull Temple Road, Basavanagudi, Bangalore 560 019
2019-2020

PROGRAM 1: INSURANCE DATABASE

Consider the Insurance database given below. The data types are specified.

PERSON (driver_id: String, name: String, address: String)
CAR (reg_num: String, model: String, year: int)
ACCIDENT (report_num: int, accident_date: date, location: String)
OWNS (driver_id: String, reg_num: String)
PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)

i) Create the above tables by properly specifying the primary keys and the foreign keys.
ii)Enter at least five tuples for each relation.
iii)Demonstrate how you
a.Update the damage amount to 25000 for the car with a specific reg-num(example 'K A053408') for which the accident report number was 12.
b.Add a new accident to the database.
iv)Find the total number of people who owned cars that involved in accidents in 2008.
v)Find the number of accidents in which cars belonging to a specific model (example)were involved.

Tables

PERSON

driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R Colony
A05	Jhon	Hanumanth nagar

CAR

reg_num	model	year
KA052250	Indica	1990
KA031181	Lancer	1957
KA095477	Toyota	1998
KA053408	Honda	2008
KA041702	Audi	2005

OWNS

driver_id	reg_num
A01	KA052250
A02	KA053408
A03	KA031181
A04	KA095477
A05	KA041702

ACCIDENT

report_num	accident_date	location
11	01-JAN-03	Mysore Road
12	02-FEB-04	South end Circle
13	21-JAN-03	Bull temple Road
14	17-FEB-08	Mysore Road
15	04-MAR-05	Kanakpura Road

PARTICIPATED

driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA053408	12	50000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

QUERY 1: Create the above tables by properly specifying the primary keys and the foreign keys.

SQL>**create table person (driver_id varchar(10),**
name varchar(20),
address varchar(30),
primary key(driver_id));
Table created.

SQL>**desc person**

Name	Null?	Type
DRIVER_ID	NOT NULL	VARCHAR2(10)
NAME		VARCHAR2(20)
ADDRESS		VARCHAR2(30)

SQL>**create table car(reg_num varchar(10),model varchar(10),year int,primary key(reg_num));**
Table created.

SQL>**desc car**

Name	Null?	Type
REG_NUM	NOT NULL	VARCHAR2(10)
MODEL		VARCHAR2(10)
YEAR		NUMBER(38)

SQL>**create table accident(report_num int,accident_date date,location varchar(20),primary key(report_num));**

Table created.

SQL>**desc accident**

Name	Null?	Type
REPORT_NUM	NOT NULL	NUMBER(38)
ACCIDENT_DATE		DATE
LOCATION		VARCHAR2(20)

SQL>**create table owns(driver_id varchar(10),reg_num varchar(10),**
primary key(driver_id,reg_num),
foreign key(driver_id) referencesperson(driver_id),
foreign key(reg_num) references car(reg_num));

Table created.

SQL>**desc owns**

Name	Null?	Type
DRIVER_ID	NOT NULL	VARCHAR2(10)
REG_NUM	NOT NULL	VARCHAR2(10)

SQL>**create table participated(driver_id varchar(10), reg_num varchar(10),**
report_num int, damage_amount int,
primary key(driver_id,reg_num,report_num),
foreign key(driver_id) references person(driver_id),
foreign key(reg_num) references car(reg_num),
foreign key(report_num) references accident(report_num));
Table created.
SQL>**desc participated**

Name	Null?	Type
DRIVER_ID	NOT NULL	VARCHAR2(10)
REG_NUM	NOT NULL	VARCHAR2(10)
REPORT_NUM	NOT NULL	NUMBER(38)
DAMAGE_AMOUNT		NUMBER(38)

QUERY 2: Enter at least five tuples for each relation

SQL> insert into person values('&driver_id','&name','&address');

SQL>**commit;**
Commit complete.

SQL> select * from person;

DRIVER_ID	NAME	ADDRESS
A01	Richard	Srinivas Nagar
A02	Pradeep	Rajajinagar
A03	Smith	Ashoknagar
A04	Venu	N.R.Colony
A05	John	Hanumanth Nagar

SQL> insert into car values('&reg_num','&model', &year);

Enter value for reg_num: KA052250
Enter value for model: Indica
Enter value for year: 1990
old 1: insert into car values('&reg_num','&model', &year)
new 1: insert into car values('KA052250','Indica', 1990)

1 row created.

SQL>/
Enter value for reg_num: KA031181
Enter value for model: Lancer
Enter value for year: 1957
old 1: insert into car values('&reg_num','&model',&year)
new 1: insert into car values('KA031181','Lancer', 1957)

1 row created.

SQL>**commit;**

Commit complete.

SQL> select * from car;

```
REG_NUM      MODEL   YEAR
-------------------------- ----------
KA052250   Indica          1990
KA031181   Lancer 1957
KA095477   Toyota  1998
KA053408   Honda   2008
KA041702   Audi            2005
```

SQL> insert into accident values(&report_num,'&accident_date','&location');

```
Enter value for report_num: 11
Enter value for accident_date: 01-JAN-03
Enter value for location: Mysore Road
old   1: insert into accident values(&report_num,'&accident_date','&location')
new   1: insert into accident values(111,'01-JAN-03','Mysore Road')
```

1 row created.

SQL>**commit;**

Commit complete.
SQL> select * from accident;

```
 REPORT_NUM   ACCIDENT_DATE     LOCATION
------------------- ------------ --------------------
 11                01-JAN-03              Mysore Road
    12              02-FEB-04              Southend Circle
 13                21-JAN-03              Bulltemple Road
  14               17-FEB-08              Mysore Road
    15              04-MAR-05              Kanakpura Road
```

SQL> insert into owns values ('&driver_id','&reg_num');

```
Enter value for driver_id: A01
Enter value for reg_num: KA052250
old   1: insert into owns values('&driver_id','&reg_num')
new   1: insert into owns values('A01','KA052250')
1 row created.
```

SQL>**commit;**
Commit complete.

SQL> select * from owns;

```
DRIVER_ID   REG_NUM
--------------- ----------
A01               KA052250
A02               KA053408
```

A04 KA031181
A03 KA095477
A05 KA041702

SQL> insert into participated values ('&driver_id','&reg_num',&report_num, &damage_amount);

Enter value for driver_id: A01
Enter value for reg_num: KA052250
Enter value for report_num: 11
Enter value for damage_amount: 10000
old 1: insert into participated values ('&driver_id','&reg_num',&report_num,&damage_amount)
new 1: insert into participated values('A01','KA052250',11,10000)

1 row created.

SQL>/
Enter value for driver_id: A02
Enter value for reg_num: KA053408
Enter value for report_num: 12
Enter value for damage_amount: 50000
old 1: insert into participated values ('&driver_id','&reg_num', &report_num,& damage_amount)
new 1: insert into participated values('A02','KA053408',12,50000)

1 row created.

SQL>**commit;**
Commit complete.

SQL> select * from participated;

DRIVER_ID	REG_NUM	REPORT_NUM	DAMAGE_AMOUNT
A01	KA052250	11	10000
A02	KA053408	12	50000
A03	KA095477	13	25000
A04	KA031181	14	3000
A05	KA041702	15	5000

QUERY 3:

a) Update the damage amount to 25000 for the car with a specific reg_num (example 'KA053408') for which the accident report number was 12.

SQL> update participated set damage_amount=25000 where reg_num='KA053408' and report_num=12;

1 row updated.

SQL>**commit;**

Commit complete.

SQL>**select * from participated;**

```
DRIVER_ID   REG_NUM  REPORTNUM  DAMAGE_AMOUNT
---------- --------- ---------- ----------
A01        KA052250         11         10000
A02        KA053408         12         25000
A03        KA095477         13         25000
A04        KA031181         14          3000
A05        KA041702         15          5000
```

b) Add a new accident to the database.

SQL>**insert into accident values(16,'15-MAR-08','Domlur');**
1 row created.

SQL>**select * from accident;**

```
 REPORT_NUMACCIDENT_DATE     LOCATION
----------------- ------------  --------------------
 11               01-JAN-03           Mysore Road
 12               02-FEB-04           Southend Circle
 13               21-JAN-03           Bulltemple Road
 14               17-FEB-08           Mysore Road
 15               04-MAR-05           Kanakpura Road
 16               15-MAR-08           Domlur
```

6 rows selected.

QUERY 4: Find the total number of people who owned cars that were involved in accidents in 2008.

SQL>**select count(distinct driver_id) CNT from participated a, accident b where a.report_num=b.report_num and b.accident_date like '%08';**

```
       CNT
----------
         1
```

QUERY 5: Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.

SQL> select count(report_num) CNT from car c,participated p where c.reg_num=p.reg_num and model='Lancer';

```
       CNT
----------
         1
```

PROGRAM 5

DESCRIPTION:

The following relations keep track of a banking enterprise.

- BRANCH(branch-name:string, branch-city:string, assets:real)
- ACCOUNT(accno:int, branch-name:string, balance:real)
- DEPOSITOR(customer-name:string, accno:int)
- CUSTOMER(customer-name:string, customer-street:string, customer-city:string)
- LOAN(loan-number:int, branch-name:string, amount:real)
- BORROWER(customer-name:string, loan-number:int)

Queries:

Write each of the following queries in SQL.

1. Create the above tables by properly specifying the primary keys and the foreign keys
2. Enter at least five tuples for each relation.

Create:

```
CREATE TABLE BRANCH
     ( branch_name VARCHAR(15),
       branch_city VARCHAR(15),
       assets NUMBER(10,2),
       PRIMARY KEY(branch_name)
     );

     CREATE TABLE ACCOUNT
     ( accno INTEGER(8),
       branch_name VARCHAR(15),
       balance NUMBER(10,2),
       PRIMARY KEY(accno),
       FOREIGN KEY(branch_name) REFERENCES BRANCH(branch_name)ON DELETE CASCADE
     );

    CREATE TABLE CUSTOMER
    ( customer_name VARCHAR(15),
      customer_street VARCHAR(15),
      customer_city VARCHAR(15),
      PRIMARY KEY(customer_name)
    );

    CREATE TABLE LOAN
    ( loan_number INTEGER(8),
      branc_hname VARCHAR(15),
      amount NUMBER(10,2),
      PRIMARY KEY(loan_number),
      FOREIGN KEY(branch_name) REFERENCES BRANCH(branch_name)
    );

    CREATE TABLE DEPOSITOR
    ( customer_name VARCHAR(15),
      accno INTEGER,
      PRIMARY KEY(customer_name, accno),
      FOREIGN KEY(customer_name) REFERENCES CUSTOMER(customer_name),
      FOREIGN KEY(accno) REFERENCES ACCOUNT(accno)
    );

    CREATE TABLE BORROWER
    ( customer_name  VARCHAR(15),
      loan_number INTEGER(8),
      PRIMARY KEY(customer_name, loan_number),
      FOREIGN KEY(customer_name) REFERENCES CUSTOMER(customer_name),
      FOREIGN KEY(loan_number) REFERENCES LOAN(loan_number)
    );
```

INSERTIONS:

```
mysql> insert into branch values
    ->         ("b1","c1",10000),
    ->         ("b2","c2",20000),
    ->         ("b3","c3",30000),
    ->         ("b4","c4",40000),
    ->         ("b5","c5",50000);
Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings:0
```

```
mysql> select * from branch;

+-------------+-------------+--------+
| branch_name | branch_city | assets |
+-------------+-------------+--------+
| b1          | c1          |  10000 |
| b2          | c2          |  20000 |
| b3          | c3          |  30000 |
| b4          | c4          |  40000 |
| b5          | c5          |  50000 |
+-------------+-------------+--------+
5 rows in set (0.00 sec)
```

```
mysql> insert into account values
    ->         (12,"b1",3000),
    ->         (22,"b2",4000),
    ->         (32,"b3",5000),
    ->         (42,"b4",6000),
    ->         (52,"b5",7000);

Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from account;

+-------+-------------+---------+
| accno | branch_name | balance |
+-------+-------------+---------+
|    12 | b1          |    3000 |
|    22 | b2          |    4000 |
|    32 | b3          |    5000 |
|    42 | b4          |    6000 |
|    52 | b5          |    7000 |
+-------+-------------+---------+
5 rows in set (0.00 sec)
```

```
mysql> insert into customer values
    ->         ("cust1","cstreet1","ccity1"),
    ->         ("cust2","cstreet2","ccity2"),
    ->         ("cust3","cstreet3","ccity3"),
    ->      ("cust4","cstreet4","ccity4"),
    ->      ("cust5","cstreet5","ccity5");

Query OK, 5 rows affected (0.07 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from customer;
+---------------+-----------------+---------------+
| customer_name | customer_street | customer_city |
+---------------+-----------------+---------------+
| cust1         | cstreet1        | ccity1        |
| cust2         | cstreet2        | ccity2        |
| cust3         | cstreet3        | ccity3        |
| cust4         | cstreet4        | ccity4        |
| cust5         | cstreet5        | ccity5        |
+---------------+-----------------+---------------+
5 rows in set (0.00 sec)
```

```
mysql> insert into depositor values
    ->         ("cust1",12),
    ->         ("cust2",22),
    ->         ("cust3",32),
    ->         ("cust4",42),
    ->         ("cust5",52);

Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from depositor;

+---------------+-------+
| customer_name | accno |
+---------------+-------+
| cust1         |    12 |
| cust2         |    22 |
| cust3         |    32 |
| cust4         |    42 |
| cust5         |    52 |
+---------------+-------+
5 rows in set (0.00 sec)
```

```
mysql> insert into values loan
    -> (10,"b1",10000),
    -> (20,"b2",20000),
    -> (30,"b3",30000),
    -> (40,"b4",40000),
    -> (50,"b5",50000);

Query OK, 5 rows affected (0.06 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from loan;

+-------------+-------------+--------+
| loan_number | branch_name | amount |
+-------------+-------------+--------+
|          10 | b1          |  10000 |
|          20 | b2          |  20000 |
|          30 | b3          |  30000 |
|          40 | b4          |  40000 |
|          50 | b5          |  50000 |
+-------------+-------------+--------+
5 rows in set (0.00 sec)
```

```
mysql> insert into borrower values
    -> ("cust1",10),
    -> ("cust2",20),
    -> ("cust3",30),
    -> ("cust4",40),
    -> ("cust5",50);

Query OK, 5 rows affected (0.05 sec)
Records: 5  Duplicates: 0  Warnings: 0
```

```
mysql> select * from borrower;

+---------------+-------------+
| customer_name | loan_number |
+---------------+-------------+
| cust1         |          10 |
| cust2         |          20 |
| cust3         |          30 |
| cust4         |          40 |
| cust5         |          50 |
+---------------+-------------+
5 rows in set (0.00 sec)
```

QUERIES:

iii. Find all the customers who have at least two accounts at the Main branch.

```
mysql> SELECT customer_name FROM depositor d,account a WHERE
     d.accno=a.accno AND a.branch_name='Main'
     GROUP BY d.customer_name HAVING COUNT(d.customer_name)>=2;

Empty set (0.00 sec)
```

Note:Here we are getting empty set because in our 'account' table there is no branch_name with value 'Main' and also there are no customer who has two accounts at the Main branch.So we have to either update the table or else add the proper tuples so that we can get the proper outputs.

updating can be done with the following commands.

```
mysql> update account set branch_name='Main' where branch_name="b1";
mysql> update account set branch_name='Main' where branch_name="b2";
mysql> update account set customer_name='cust1' where customer_name="cust2";

+---------------+
| customer_name |
+---------------+
| cust1         |
+---------------+
1 row in set (0.00 sec)
```

Description: The query is selecting the customer's name such that the account number associated with name is in both the account table and depositor table and also the name of the branch in the account table is 'Main' and then the tuples are being grouped by customer name in the depositor table and also the customer name having count atleast equal to 2 are being selected.

iv. Find all the customers who have an account at all the branches located in a specific city.

```
 mysql> SELECT d.customer_name FROM account a,branch b,depositor d WHERE
       b.branch_name=a.branch_name AND a.accno=d.accno AND b.branch_city='c3'
       GROUP BY d.customer_name HAVING COUNT(distinct b.branch_name)=(select count(branch

+---------------+
| customer_name |
+---------------+
| cust3         |
+---------------+
1 row in set (0.00 sec)
```

Description: The query selects the customers from the the depositor table such that branch name is in both the branch table and also account table and the account number in the selected tuples is in both account table and in depositor table and also the name of the branch city is 'c3'. The selected tuples are gruoped by the customer name of the depositor table whose count should be equal to the count of the branch name in the branch table with brach city 'c3'.

v. Demonstrate how you delete all account tuples at every branch located in a specific city.

```
mysql> DELETE FROM account where branch_name IN(SELECT branch_name FROM branch WHERE bran

Query OK, 1 row affected (0.04 sec)

mysql>SELECT * FROM account;


+-------+-------------+---------+
| accno | branch_name | balance |
+-------+-------------+---------+
|    12 | b1          |    3000 |
|    22 | b2          |    4000 |
|    32 | b3          |    5000 |
|    42 | b4          |    6000 |
+-------+-------------+---------+
4 rows in set (0.00 sec)
```

Description:The inner query "SELECT branch_name FROM branch WHERE branch_city='c5' " selects the branch names from the branch table where the branch city is c5. The selected tuples are given as input to the outer query which deletes the tuples with the selected branch names.

PROGRAM 3: SUPPLIER DATABASE

Consider the following schema:

SUPPLIERS(sid: integer, sname: string, address: string)

PARTS(pid: integer, pname: string, color: string)

CATALOG(sid: integer, pid: integer, cost: real)

The Catalog relation lists the prices charged for parts by Suppliers.

Write the following queries in SQL:

i) Find the pnames of parts for which there is some supplier.

ii) Find the snames of suppliers who supply every part.

iii) Find the snames of suppliers who supply every red part.

iv) Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

v) Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

vi) For each part, find the sname of the supplier who charges the most for that part.

Schema Diagram

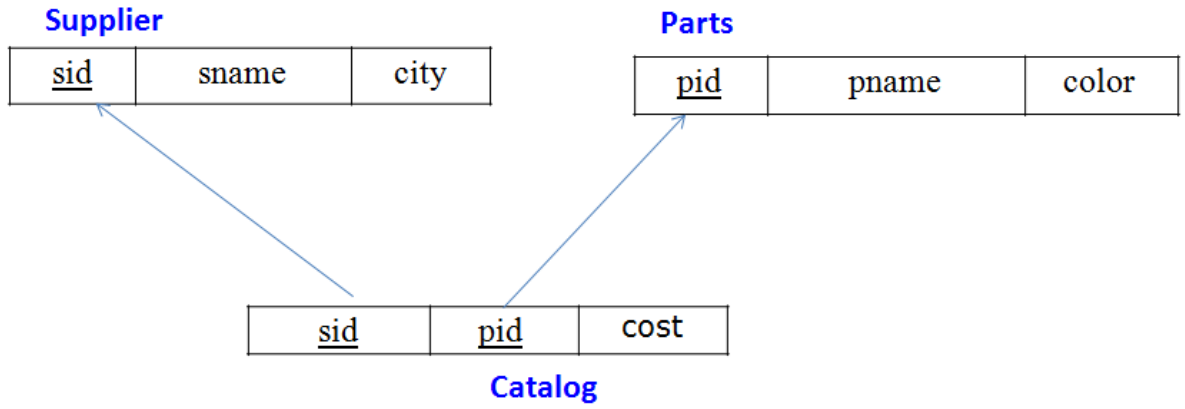


Table Data

SUPPLIERS

SID	SNAME	CITY
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi

PARTS

PID	PNAME	COLOR
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black

CATALOG

SID	PID	COST
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10001	20005	10
10002	20001	10
10002	20002	20
10003	20003	30
10004	20003	40

CREATION of Tables:

SQL> create table SUPPLIERS(sid number(5) primary key, sname varchar(20), city varchar(20));

Table created.

SQL> desc SUPPLIERS;

Name	Null?	Type
SID	NOT NULL	NUMBER(5)
SNAME		VARCHAR2(20)
CITY		VARCHAR2(20)

SQL> create table PARTS(pid number(5) primary key, pname varchar(20), color varchar(10));

Table created.

SQL> desc PARTS;

Name	Null?	Type
PID	NOT NULL	NUMBER(5)
PNAME		VARCHAR2(20)
COLOR		VARCHAR2(10)

SQL> create table CATALOG(sid number(5), pid number(5), foreign key(sid) references SUPPLIERS(sid), foreign key(pid) references PARTS(pid), cost float(6), primary key(sid, pid));

Table created.

SQL> desc CATALOG;

Name	Null?	Type
SID	NOT NULL	NUMBER(5)
PID	NOT NULL	NUMBER(5)
COST		FLOAT(6)

INSERTION OF DATA:

SQL> insert into suppliers values(&sid, '&sname','&city');

Enter value for sid: 10001

Enter value for sname: Acme Widget

Enter value for address: Bangalore

old 1: insert into suppliers values(&sid, '&sname','&city')

new 1: insert into suppliers values(10001, 'Acme Widget','Bangalore')

1 row created.

SQL> /

Enter value for sid: 10002

Enter value for sname: Johns

Enter value for address: Kolkata

old 1: insert into suppliers values(&sid, '&sname','&city')

new 1: insert into suppliers values(10002, 'Johns','Kolkata')

1 row created.

SQL> /
Enter value for sid: 10003
Enter value for sname: Vimal
Enter value for address: Mumbai
old 1: insert into suppliers values(&sid, '&sname','&city')
new 1: insert into suppliers values(10003, 'Vimal','Mumbai')

1 row created.

SQL> /
Enter value for sid: 10004
Enter value for sname: Reliance
Enter value for address: Delhi
old 1: insert into suppliers values(&sid, '&sname','&city')
new 1: insert into suppliers values(10004, 'Reliance','Delhi')

1 row created.

SQL> /
Enter value for sid: 10005
Enter value for sname: Mahindra
Enter value for address: Mumbai
old 1: insert into suppliers values(&sid, '&sname','&city')
new 1: insert into suppliers values(10005, 'Mahindra','Mumbai')

1 row created.

SQL> select * from SUPPLIERS;

SID	SNAME	CITY
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai

10004 Reliance Delhi

SQL> commit;

Commit complete.

SQL> insert into PARTS values(&pid, '&pname','&color');
Enter value for pid: 20001
Enter value for pname: Book
Enter value for color: Red
old 1: insert into PARTS values(&pid, '&pname','&color')
new 1: insert into PARTS values(20001, 'Book','Red')

1 row created.

SQL> /
Enter value for pid: 20002
Enter value for pname: Pen
Enter value for color: Red
old 1: insert into PARTS values(&pid, '&pname','&color')
new 1: insert into PARTS values(20002, 'Pen','Red')

1 row created.

SQL> /
Enter value for pid: 20003
Enter value for pname: Pencil
Enter value for color: Green
old 1: insert into PARTS values(&pid, '&pname','&color')
new 1: insert into PARTS values(20003, 'Pencil','Green')

1 row created.

SQL> /

Enter value for pid: 20004
Enter value for pname: Mobile
Enter value for color: Green
old 1: insert into PARTS values(&pid, '&pname','&color')
new 1: insert into PARTS values(20004, 'Mobile','Green')

1 row created.

SQL> /
Enter value for pid: 20005
Enter value for pname: Charger
Enter value for color: Black
old 1: insert into PARTS values(&pid, '&pname','&color')
new 1: insert into PARTS values(20005, 'Charger','Black')

1 row created.

SQL> select * from PARTS;

PID	PNAME	COLOR
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black

SQL> commit;

Commit complete.

SQL> insert into CATALOG values(&sid, '&pid','&cost');
Enter value for sid: 10001

```
Enter value for pid: 20001
Enter value for cost: 10
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10001, '20001','10')

1 row created.

SQL> /
Enter value for sid: 10001
Enter value for pid: 20002
Enter value for cost: 10
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10001, '20002','10')

1 row created.

SQL> /
Enter value for sid: 10001
Enter value for pid: 20003
Enter value for cost: 30
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10001, '20003','30')

1 row created.

SQL> /
Enter value for sid: 10001
Enter value for pid: 20004
Enter value for cost: 10
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10001, '20004','10')

1 row created.
```

```
SQL> /
Enter value for sid: 10001
Enter value for pid: 20005
Enter value for cost: 10
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10001, '20005','10')

1 row created.

SQL> /
Enter value for sid: 10002
Enter value for pid: 20001
Enter value for cost: 10
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10002, '20001','10')

1 row created.

SQL> /
Enter value for sid: 10002
Enter value for pid: 20002
Enter value for cost: 20
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10002, '20002','20')

1 row created.

SQL> /
Enter value for sid: 10003
Enter value for pid: 20003
Enter value for cost: 30
old   1: insert into CATALOG values(&sid, '&pid','&cost')
new   1: insert into CATALOG values(10003, '20003','30')
```

1 row created.

SQL> /
Enter value for sid: 10004
Enter value for pid: 20003
Enter value for cost: 40
old 1: insert into CATALOG values(&sid, '&pid','&cost')
new 1: insert into CATALOG values(10004, '20003','40')

1 row created.

SQL> select * from CATALOG;

SID	PID	COST
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10001	20005	10
10002	20001	10
10002	20002	20
10003	20003	30
10004	20003	40

9 rows selected.

i) <u>Find the pnames of parts for which there is some supplier.</u>

```
SQL> SELECT DISTINCT P.pname
  2  FROM Parts P, Catalog C
  3  WHERE P.pid = C.pid;
```

PNAME

Book

Charger

Mobile

Pen

Pencil

ii) <u>Find the snames of suppliers who supply every part.</u>

```
SQL> SELECT S.sname
  2  FROM Suppliers S
  3  WHERE NOT EXISTS ((SELECT P.pid  FROM Parts P)
  4         MINUS (SELECT C.pid FROM Catalog C
  5         WHERE C.sid = S.sid));
```

SNAME

Acme Widget

iii) Find the snames of suppliers who supply every red part.

```
SQL>SELECT S.sname
FROM Suppliers S
WHERE NOT EXISTS (( SELECT P.pid
                    FROM Parts P
                    WHERE P.color = 'Red' )
                    MINUS
                    ( SELECT C.pid
                    FROM Catalog C, Parts P
                    WHERE C.sid = S.sid AND
                    C.pid = P.pid AND P.color = 'Red' ));
```

SNAME

Acme Widget

Johns

iv) Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

```
SQL> select pname from parts where pid in (select pid from cataloge where sid =(
select sid from suppliers where sname='Acme widget') minus select pid from catal
oge where sid in (select sid from suppliers where sname <>'Acme widget'));

PNAME
---------------
Mobile
Charger
```

PNAME

Mobile

Charger

v) Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

```
SQL> SELECT DISTINCT C.sid FROM Catalog C
  2  WHERE C.cost > ( SELECT AVG (C1.cost)
  3  FROM Catalog C1
  4  WHERE C1.pid = C.pid );

      SID
----------
    10002
    10004
```

vi) For each part, find the sname of the supplier who charges the most for that part.

```
SQL>SELECT P.pid, S.sname
FROM Parts P, Suppliers S, Catalog C
WHERE C.pid = P.pid
AND C.sid = S.sid
AND C.cost = (SELECT MAX (C1.cost)
                FROM Catalog C1
                WHERE C1.pid = P.pid);
```

PID SNAME

---------- --------------------
20001 Acme Widget
20004 Acme Widget
20005 Acme Widget
20001 Johns
20002 Johns
20003 Reliance

6 rows selected.

PROGRAM 4: STUDENT FACULTY DATABASE

Consider the following database for student enrollment for course :

STUDENT(snum: integer, sname: string, major: string, lvl: string, age: integer)

CLASS(cname: string, meets at: time, room: string, fid: integer)

ENROLLED(snum: integer, cname: string)

FACULTY(fid: integer, fname: string, deptid: integer)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Level(lvl) is a two character code with 4 different values (example: Junior: JR etc)

Write the following queries in SQL. No duplicates should be printed in any of the answers.

i. Create above mentioned tables

ii. insert records into each of the tables

i. Find the names of all Juniors (level = JR) who are enrolled in a class taught by

ii. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.

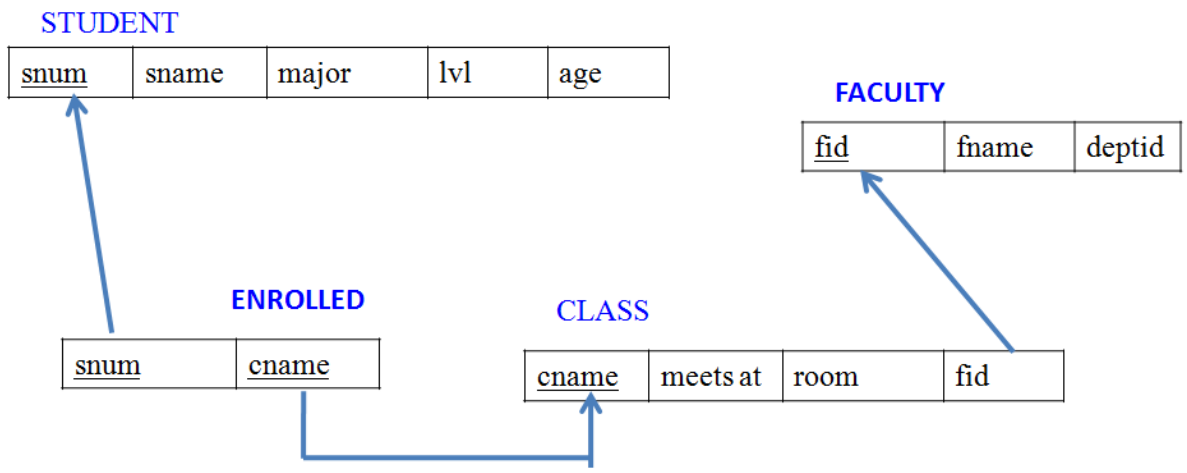
iii. Find the names of all students who are enrolled in two classes that meet at the same time.

iv. Find the names of faculty members who teach in every room in which some class is taught.

v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

vi. Find the names of students who are not enrolled in any class.

vii. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).



STUDENT

SNUM	SNAME	MA	LV	AGE
1	jhon	CS	Sr	19
2	Smith	CS	Jr	20
3	Jacob	CV	Sr	20
4	Tom	CS	Jr	20
5	Rahul	CS	Jr	20
6	Rita	CS	Sr	21

FACULTY

FID	FNAME	DEPTID
11	Harish	1000
12	MV	1000
13	Mira	1001
14	Shiva	1002
15	Nupur	1000

ENROLLED

SNUM	CNAME
1	class1
2	class1
3	class3
4	class3
5	class4
1	class5
2	class5
3	class5
4	class5
5	class5

CLASS

CNAME	METTS_AT	ROOM	FID
class1	12/11/15 10:15:16	R1	14
class10	12/11/15 10:15:16	R128	14
class2	12/11/15 10:15:20	R2	12
class3	12/11/15 10:15:25	R3	11
class4	12/11/15 20:15:20	R4	14
class5	12/11/15 20:15:20	R3	15
class6	12/11/15 13:20:20	R2	14
class7	12/11/15 10:10:10	R3	14

```
SQL> CREATE TABLE student(
  2     snum INT,
  3     sname VARCHAR(10),
  4     major VARCHAR(2),
  5     lvl VARCHAR(2),
  6     age INT, primary key(snum));
```

Table created.

SQL> desc student;

Name	Null?	Type
SNUM	NOT NULL	NUMBER(38)
SNAME		VARCHAR2(10)
MAJOR		VARCHAR2(2)
LVL		VARCHAR2(2)
AGE		NUMBER(38)

```
SQL> CREATE TABLE faculty(
  2  fid INT,fname VARCHAR(20),
  3  deptid INT,
  4  PRIMARY KEY(fid));
```

Table created.

SQL> desc faculty;

Name	Null?	Type
FID	NOT NULL	NUMBER(38)
FNAME		VARCHAR2(20)
DEPTID		NUMBER(38)

```
SQL>  CREATE TABLE class(
  2  cname VARCHAR(20),
  3  metts_at TIMESTAMP,
  4  room VARCHAR(10),
5  fid INT,
  6  PRIMARY KEY(cname),
  7  FOREIGN KEY(fid) REFERENCES faculty(fid));
```

Table created.

SQL> DESC class;

Name	Null?	Type
CNAME	NOT NULL	VARCHAR2(20)
METTS_AT		TIMESTAMP(6)
ROOM		VARCHAR2(10)
FID		NUMBER(38)

```
SQL> CREATE TABLE enrolled(
  2  snum INT,
  3  cname VARCHAR(20),
  4  PRIMARY KEY(snum,cname),
  5  FOREIGN KEY(snum) REFERENCES student(snum),
  6  FOREIGN KEY(cname) REFERENCES class(cname));
```

Table created.

SQL> desc enrolled;

Name	Null?	Type
SNUM	NOT NULL	NUMBER(38)
CNAME	NOT NULL	VARCHAR2(20)

SQL> commit;

Commit complete.

INSERTION OF VALUES:

SQL> INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age);

```
Enter value for snum: 1
Enter value for sname: jhon
Enter value for major: CS
Enter value for lvl: Sr
Enter value for age: 19
old   1: INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age)
new   1: INSERT INTO STUDENT VALUES(1, 'jhon', 'CS', 'Sr', 19)

1 row created.

SQL> /
Enter value for snum: 2
Enter value for sname: Smith
Enter value for major: CS
Enter value for lvl: Jr
Enter value for age: 20
old   1: INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age)
new   1: INSERT INTO STUDENT VALUES(2, 'Smith', 'CS', 'Jr', 20)

1 row created.

SQL> /
Enter value for snum: 3
Enter value for sname: Jacob
Enter value for major: CV
Enter value for lvl: Sr
Enter value for age: 20
old   1: INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age)
new   1: INSERT INTO STUDENT VALUES(3 , 'Jacob', 'CV', 'Sr', 20)

1 row created.

SQL> /
```

Enter value for snum: 4
Enter value for sname: Tom
Enter value for major: CS
Enter value for lvl: Jr
Enter value for age: 20
old 1: INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age)
new 1: INSERT INTO STUDENT VALUES(4, 'Tom ', 'CS', 'Jr', 20)

1 row created.

SQL> /
Enter value for snum: 5
Enter value for sname: Rahul
Enter value for major: CS
Enter value for lvl: Jr
Enter value for age: 20
old 1: INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age)
new 1: INSERT INTO STUDENT VALUES(5, 'Rahul', 'CS', 'Jr', 20)

1 row created.

SQL> /
Enter value for snum: 6
Enter value for sname: Rita
Enter value for major: CS
Enter value for lvl: Sr
Enter value for age: 21
old 1: INSERT INTO STUDENT VALUES(&snum, '&sname', '&major', '&lvl', &age)
new 1: INSERT INTO STUDENT VALUES(6, 'Rita', 'CS', 'Sr', 21)

1 row created.

SQL> select * from student;

SNUM	SNAME	MA	LV	AGE
1	jhon	CS	Sr	19
2	Smith	CS	Jr	20
3	Jacob	CV	Sr	20
4	Tom	CS	Jr	20
5	Rahul	CS	Jr	20
6	Rita	CS	Sr	21

6 rows selected.

SQL> INSERT INTO FACULTY VALUES(&FID, '&FNAME', &DEPTID);
Enter value for fid: 11
Enter value for fname: Harish
Enter value for deptid: 1000
old 1: INSERT INTO FACULTY VALUES(&FID, '&FNAME', &DEPTID)

new 1: INSERT INTO FACULTY VALUES(11, 'Harish', 1000)

1 row created.

SQL> /
Enter value for fid: 12
Enter value for fname: MV
Enter value for deptid: 1000
old 1: INSERT INTO FACULTY VALUES(&FID, '&FNAME', &DEPTID)
new 1: INSERT INTO FACULTY VALUES(12, 'MV', 1000)

1 row created.

SQL> /
Enter value for fid: 13
Enter value for fname: Mira
Enter value for deptid: 1001
old 1: INSERT INTO FACULTY VALUES(&FID, '&FNAME', &DEPTID)
new 1: INSERT INTO FACULTY VALUES(13 , 'Mira', 1001)

1 row created.

SQL> /
Enter value for fid: 14
Enter value for fname: Shiva
Enter value for deptid: 1002
old 1: INSERT INTO FACULTY VALUES(&FID, '&FNAME', &DEPTID)
new 1: INSERT INTO FACULTY VALUES(14, 'Shiva', 1002)

1 row created.

SQL> /
Enter value for fid: 15
Enter value for fname: Nupur
Enter value for deptid: 1000
old 1: INSERT INTO FACULTY VALUES(&FID, '&FNAME', &DEPTID)
new 1: INSERT INTO FACULTY VALUES(15, 'Nupur', 1000)

1 row created.

SQL> commit;

Commit complete.

SQL> select * from faculty;

```
       FID FNAME                DEPTID
---------- -------------------- ----------
        11 Harish                1000
        12 MV                    1000
        13 Mira                  1001
        14 Shiva                 1002
        15 Nupur                 1000
```

SQL> commit;

Commit complete.

SQL> alter session set nls_timestamp_format = 'RR/MM/DD HH24:MI:SSXFF';

Session altered.

SQL> alter session set nls_date_language ='ENGLISH';

Session altered.

SQL> insert into class values('&cname', '&meets_at', '&room', &fid);
Enter value for cname: class1
Enter value for meets_at: 12/11/15 10:15:16
Enter value for room: R1
Enter value for fid: 14
old 1: insert into class values('&cname', '&meets_at', '&room', &fid)
new 1: insert into class values('class1', '12/11/15 10:15:16', 'R1', 14)

1 row created.

Enter value for cname: class10
Enter value for meets_at: 12/11/15 10:15:16
Enter value for room: R128
Enter value for fid: 14
old 1: insert into class values('&cname', '&meets_at', '&room', &fid)
new 1: insert into class values('class10', '12/11/15 10:15:16', 'R128', 14)

1 row created.

SQL> /
Enter value for cname: class2
Enter value for meets_at: 12/11/15 10:15:20
Enter value for room: R2
Enter value for fid: 12
old 1: insert into class values('&cname', '&meets_at', '&room', &fid)
new 1: insert into class values('class2', '12/11/15 10:15:20', 'R2', 12)

```
1 row created.

SQL> insert into class values('&cname', '&meets_at', '&room', &fid);
Enter value for cname: class3
Enter value for meets_at: 12/11/15 10:15:25
Enter value for room: R3
Enter value for fid: 11
old   1: insert into class values('&cname', '&meets_at', '&room', &fid)
new   1: insert into class values('class3', '12/11/15 10:15:25', 'R3', 12)

1 row created.

SQL> /
Enter value for cname: class4
Enter value for meets_at: 12/11/15 20:15:20
Enter value for room: R4
Enter value for fid: 14
old   1: insert into class values('&cname', '&meets_at', '&room', &fid)
new   1: insert into class values('class4', '12/11/15 20:15:20', 'R4', 14)

1 row created.

SQL> /
Enter value for cname: class5
Enter value for meets_at: 12/11/15 20:15:20
Enter value for room: R3
Enter value for fid: 15
old   1: insert into class values('&cname', '&meets_at', '&room', &fid)
new   1: insert into class values('class5', '12/11/15 20:15:20', 'R3', 15)

1 row created.

SQL> /
Enter value for cname: class6
Enter value for meets_at: 12/11/15 13:20:20
Enter value for room: R2
Enter value for fid: 14
old   1: insert into class values('&cname', '&meets_at', '&room', &fid)
new   1: insert into class values('class6', '12/11/15 13:20:20', 'R2', 14)

1 row created.

SQL> /
Enter value for cname: class7
Enter value for meets_at: 12/11/15 10:10:10
Enter value for room: R3
Enter value for fid: 14
old   1: insert into class values('&cname', '&meets_at', '&room', &fid)
new   1: insert into class values('class7', '12/11/15 10:10:10', 'R3', 14)
```

1 row created.

SQL> select * from class;

CNAME

METTS_AT

ROOM FID
---------- ----------
class1
12/11/15 10:15:16.000000
R1 14

class10
12/11/15 10:15:16.000000
R128 14

CNAME

METTS_AT

ROOM FID
---------- ----------

class2
12/11/15 10:15:20.000000
R2 12

class3
12/11/15 10:15:25.000000

CNAME

METTS_AT

ROOM FID
---------- ----------
R3 11

class4
12/11/15 20:15:20.000000
R4 14

class5

CNAME

METTS_AT

ROOM FID
---------- ----------
12/11/15 20:15:20.000000
R3 15

class6
12/11/15 13:20:20.000000
R2 14

CNAME

METTS_AT

ROOM FID
---------- ----------
class7
12/11/15 10:10:10.000000
R3 14

8 rows selected.

SQL> commit;

Commit complete.

SQL> insert into enrolled values(&snum, '&cname');
Enter value for snum: 1
Enter value for cname: class1
old 1: insert into enrolled values(&snum, '&cname')
new 1: insert into enrolled values(1, 'class1')

1 row created.

SQL> /
Enter value for snum: 2
Enter value for cname: class1
old 1: insert into enrolled values(&snum, '&cname')
new 1: insert into enrolled values(2, 'class1')

1 row created.

SQL> /
Enter value for snum: 3
Enter value for cname: class3
old 1: insert into enrolled values(&snum, '&cname')
new 1: insert into enrolled values(3, 'class3')

1 row created.

```
SQL> /
Enter value for snum: 4
Enter value for cname: class3
old   1: insert into enrolled values(&snum, '&cname')
new   1: insert into enrolled values(4, 'class3')

1 row created.

SQL> /
Enter value for snum: 5
Enter value for cname: class4
old   1: insert into enrolled values(&snum, '&cname')
new   1: insert into enrolled values(5, 'class4')

1 row created.

SQL> /
Enter value for snum: 1
Enter value for cname: class5
old   1: insert into enrolled values(&snum, '&cname')
new   1: insert into enrolled values(1, 'class5')

1 row created.

SQL> /
Enter value for snum: 2
Enter value for cname: class5
old   1: insert into enrolled values(&snum, '&cname')
new   1: insert into enrolled values(2, 'class5')

1 row created.

SQL> /
Enter value for snum: 3
Enter value for cname: class5
old   1: insert into enrolled values(&snum, '&cname')
new   1: insert into enrolled values(3, 'class5')

1 row created.

SQL> /
Enter value for snum: 4
Enter value for cname: class5
old   1: insert into enrolled values(&snum, '&cname')
new   1: insert into enrolled values(4, 'class5')

1 row created.

SQL> /
```

Enter value for snum: 5
Enter value for cname: class5
old 1: insert into enrolled values(&snum, '&cname')
new 1: insert into enrolled values(5, 'class5')

1 row created.

SQL> select * from enrolled;

```
      SNUM CNAME
---------- --------------------
         1 class1
         2 class1
         3 class3
         4 class3
         5 class4
         1 class5
         2 class5
         3 class5
         4 class5
         5 class5

      10  rows selected.
```

iii. Find the names of all Juniors (level(lvl) = Jr) who are enrolled in a class taught by Harish.

SELECT DISTINCT S.Sname
FROM Student S, Class C, Enrolled E, Faculty F
WHERE S.snum = E.snum AND E.cname = C.cname AND C.fid = F.fid AND F.fname = 'Harish' AND S.lvl = 'Jr';

```
SNAME
----------
Tom
```

vi. Find the names of all classes that either meet in room R128 or have five or more Students enrolled.

```
SQL>SELECT C.cname
FROM Class C
WHERE C.room = 'R128'
OR C.cname IN (SELECT E.cname
        FROM Enrolled E
        GROUP BY E.cname
        HAVING COUNT (*) >= 5);

CNAME
--------------------
class10
class5
```

v. Find the names of all students who are enrolled in two classes that meet at the same time.

```
SQL>SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum IN (SELECT E1.snum
                FROM Enrolled E1, Enrolled E2, Class C1, Class C2
                WHERE E1.snum = E2.snum AND E1.cname <> E2.cname
                AND E1.cname = C1.cname
                AND E2.cname = C2.cname AND C1.meets_at =
C2.meets_at);
SNAME
----------
Rahul
```

vi. Find the names of faculty members who teach in every room in which some class is taught.

```
SELECT DISTINCT F.fname
FROM Faculty F
```

```
WHERE NOT EXISTS ((SELECT C.roomFROM Class C )
                  MINUS
                  (SELECTC1.room
                  FROM Class C1
                  WHERE C1.fid = F.fid ));
```

```
FNAME
--------------------
Shiva
```

vii. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less than five.

```
SQL>SELECT DISTINCT F.fname
FROM Faculty F
WHERE 5 > (SELECT COUNT (E.snum)
          FROM Class C, Enrolled E
          WHERE C.cname = E.cname
          AND C.fid = F.fid)
```

```
     FNAME
--------------------
Harish
MV
Mira
Shiva
```

viii. Find the names of students who are not enrolled in any class.

```
SELECT DISTINCT S.sname
FROM Student S
WHERE S.snum NOT IN (SELECT E.snum
                     FROM Enrolled E );
```

SNAME

Rita

ix. For each age value that appears in Students, find the level value that appears most often. For example, if there are more FR level students aged 18 than SR, JR, or SO students aged 18, you should print the pair (18, FR).

```
SELECT S.age, S.lvl
FROM Student S
GROUP BY S.age, S.lvl
HAVING S.lvl IN (SELECT S1.lvl FROM Student S1
                 WHERE S1.age = S.age
                 GROUP BY S1.lvl, S1.age
                 HAVING COUNT (*) >= ALL (SELECT COUNT (*)
                                          FROM Student S2
                                          WHERE s1.age = S2.age
                                          GROUP BY S2.lvl,
                                          S2.age));
```

AGE LV

---------- --

19 Sr

20 Jr

21 Sr

PROGRAM 2

DESCRIPTION:

The following relations keep track of airline flight information:

- FLIGHTS (no:integer,from:string,to:string,distance:integer,departs:time,arrives:time,price:real)
- AIRCRAFT (aid:integer,aname:string,cruisingrange:integer)
- CERTIFIED (eid:integer,aid:integer)
- EMPLOYEES (eid:integer,ename:string,salary:integer)

NOTE that the EMPLOYEES relation describes pilots and other kinds of employees as well;Every pilot is certified for some aircraft,and only pilots are certified to fly.

Queries:

```
Write each of the following queries in SQL.

1. Find the names of aircraft such that all pilots certified to operate them have salarie
2. For each pilot who is certified for more than three aircrafts,find the eid and the max
3. Find the names of all pilots whose salary is less than the price of the cheapest route
4. For all aircrafts with cruisingrange over 1000 kms,find the name of the aircraft and t
5. Find the names of pilots certified for some Boeing aircraft.
6. Find the aid's of all aircraft that can be used on routes from Bangalore to Delhi.
```

Create:

```
mysql> create database flights;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> use flights;
Database changed
mysql> create table flight(
    -> no int,
    -> frm varchar(20),
    -> too varchar(20),
    -> distance int,
    -> departs varchar(20),
    -> arrives varchar(20),
    -> price real,
    -> primary key (no) );
Query OK, 0 rows affected (0.17 sec)
```

```
mysql> desc flight;
+----------+-------------+------+-----+---------+-------+
| Field    | Type        | Null | Key | Default | Extra |
+----------+-------------+------+-----+---------+-------+
| no       | int(11)     | NO   | PRI | 0       |       |
| frm      | varchar(20) | YES  |     | NULL    |       |
| too      | varchar(20) | YES  |     | NULL    |       |
| distance | int(11)     | YES  |     | NULL    |       |
| departs  | varchar(20) | YES  |     | NULL    |       |
| arrives  | varchar(20) | YES  |     | NULL    |       |
| price    | double      | YES  |     | NULL    |       |
+----------+-------------+------+-----+---------+-------+
7 rows in set (0.00 sec)
```

```
mysql> create table aircraft(
    -> aid int,
    -> aname varchar(20),
    -> cruisingrange int,
    -> primary key (aid) );
Query OK, 0 rows affected (0.19 sec)
```

```
mysql> desc aircraft;
+---------------+-------------+------+-----+---------+-------+
| Field         | Type        | Null | Key | Default | Extra |
+---------------+-------------+------+-----+---------+-------+
| aid           | int(11)     | NO   | PRI | 0       |       |
| aname         | varchar(20) | YES  |     | NULL    |       |
| cruisingrange | int(11)     | YES  |     | NULL    |       |
+---------------+-------------+------+-----+---------+-------+
3 rows in set (0.01 sec)
```

```
mysql> create table employees(
    -> eid int,
    -> ename varchar(20),
    -> salary int,
    -> primary key (eid) );
Query OK, 0 rows affected (0.29 sec)
```

```
mysql> desc employees;
+--------+-------------+------+-----+---------+-------+
| Field  | Type        | Null | Key | Default | Extra |
+--------+-------------+------+-----+---------+-------+
| eid    | int(11)     | NO   | PRI | 0       |       |
| ename  | varchar(20) | YES  |     | NULL    |       |
| salary | int(11)     | YES  |     | NULL    |       |
+--------+-------------+------+-----+---------+-------+
3 rows in set (0.00 sec)
```

```
mysql> create table certified(
    -> eid int,
    -> aid int,
    -> primary key (eid,aid),
    -> foreign key (eid) references employees (eid),
    -> foreign key (aid) references aircraft (aid) );
Query OK, 0 rows affected (0.43 sec)
```

```
mysql> desc certified;
+-------+---------+------+-----+---------+-------+
| Field | Type    | Null | Key | Default | Extra |
+-------+---------+------+-----+---------+-------+
| eid   | int(11) | NO   | PRI | 0       |       |
| aid   | int(11) | NO   | PRI | 0       |       |
+-------+---------+------+-----+---------+-------+
2 rows in set (0.00 sec)
```

Insertion:

```
mysql> insert into flight values (1,'Bangalore','Mangalore',360,'10:45:00','12:00:00',10(
Query OK, 7 rows affected (0.06 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

```
mysql> select * from flight;
+----+-----------+-----------+----------+----------+----------+-------+
| no | frm       | too       | distance | departs  | arrives  | price |
+----+-----------+-----------+----------+----------+----------+-------+
|  1 | Bangalore | Mangalore |      360 | 10:45:00 | 12:00:00 | 10000 |
|  2 | Bangalore | Delhi     |     5000 | 12:15:00 | 04:30:00 | 25000 |
|  3 | Bangalore | Mumbai    |     3500 | 02:15:00 | 05:25:00 | 30000 |
|  4 | Delhi     | Mumbai    |     4500 | 10:15:00 | 12:05:00 | 35000 |
|  5 | Delhi     | Frankfurt |    18000 | 07:15:00 | 05:30:00 | 90000 |
|  6 | Bangalore | Frankfurt |    19500 | 10:00:00 | 07:45:00 | 95000 |
|  7 | Bangalore | Frankfurt |    17000 | 12:00:00 | 06:30:00 | 99000 |
+----+-----------+-----------+----------+----------+----------+-------+
7 rows in set (0.00 sec)
```

```
mysql> insert into aircraft values (123,'Airbus',1000),(302,'Boeing',5000),(306,'Jet01',5
Query OK, 7 rows affected (0.07 sec)
Records: 7  Duplicates: 0  Warnings: 0

mysql> select * from aircraft;
+-----+------------+---------------+
| aid | aname      | cruisingrange |
+-----+------------+---------------+
| 123 | Airbus     |          1000 |
| 302 | Boeing     |          5000 |
| 306 | Jet01      |          5000 |
| 378 | Airbus380  |          8000 |
| 456 | Aircraft   |           500 |
| 789 | Aircraft02 |           800 |
| 951 | Aircraft03 |          1000 |
+-----+------------+---------------+
7 rows in set (0.00 sec)
```

```
mysql> insert into employees values(1,'Ajay',30000),(2,'Ajith',85000),(3,'Arnab',50000),(
Query OK, 7 rows affected (0.29 sec)
Records: 7  Duplicates: 0  Warnings: 0
```

```
mysql> select * from employees;
+-----+-------+--------+
| eid | ename | salary |
+-----+-------+--------+
|   1 | Ajay  |  30000 |
|   2 | Ajith |  85000 |
|   3 | Arnab |  50000 |
|   4 | Harry |  45000 |
|   5 | Ron   |  90000 |
|   6 | Josh  |  75000 |
|   7 | Ram   | 100000 |
+-----+-------+--------+
7 rows in set (0.00 sec)
```

```
mysql> insert into certified values (1,123),(2,123),(1,302),(5,302),(7,302),(1,306),(2,30
Query OK, 17 rows affected (0.30 sec)
Records: 17  Duplicates: 0  Warnings: 0
```

```
mysql> select * from certified;
+-----+-----+
| eid | aid |
+-----+-----+
|   1 | 123 |
|   2 | 123 |
|   1 | 302 |
|   5 | 302 |
|   7 | 302 |
|   1 | 306 |
|   2 | 306 |
|   1 | 378 |
|   2 | 378 |
|   4 | 378 |
|   3 | 456 |
|   6 | 456 |
|   1 | 789 |
|   5 | 789 |
|   6 | 789 |
|   1 | 951 |
|   3 | 951 |
+-----+-----+
17 rows in set (0.00 sec)
```

Queries:

1.Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs

80,000.

```
mysql> select distinct a.aname
    -> from aircraft a,certified c,employees e
    -> where a.aid=c.aid
    -> and c.eid=e.eid
    -> and not exists
    -> (select *
    -> from employees e1
    -> where e1.eid=e.eid
    -> and e1.salary<80000);
+------------+
| aname      |
+------------+
| Airbus     |
| Boeing     |
| Jet01      |
| Airbus380  |
| Aircraft02 |
+------------+
5 rows in set (0.00 sec)
```

2.For each pilot who is certified for more than three aircrafts,find the eid and the maximum cruisingrange of the aircraft for which he/she is certified.

```
mysql> select c.eid,max(cruisingrange)
    -> from certified c,aircraft a
    -> where c.aid=a.aid
    -> group by c.eid
    -> having count(*)>3;
+-----+--------------------+
| eid | max(cruisingrange) |
+-----+--------------------+
|   1 |               8000 |
+-----+--------------------+
1 row in set (0.00 sec)
```

3.Find the names of all pilots whose salary is less than the price of the cheapest route from Bangalore to Frankfurt.

```
mysql> select distinct e.ename
    -> from employees e
    -> where e.salary<
    -> (select min(f.price)
    -> from flight f
    -> where f.frm='Bangalore'
    -> and f.too='Frankfurt');
+-------+
| ename |
+-------+
| Ajay  |
| Ajith |
| Arnab |
| Harry |
| Ron   |
| Josh  |
+-------+
6 rows in set (0.00 sec)
```

4.For all aircrafts with cruisingrange over 1000 kms,find the name of the aircraft and the average salary of all pilots certified for this aircraft.

```
mysql> select a.aid,a.aname,avg(e.salary)
    -> from aircraft a,certified c,employees e
    -> where a.aid=c.aid
    -> and c.eid=e.eid
    -> and a.cruisingrange>1000
    -> group by a.aid,a.aname;
+-----+-----------+---------------+
| aid | aname     | avg(e.salary) |
+-----+-----------+---------------+
| 302 | Boeing    |    73333.3333 |
| 306 | Jet01     |    57500.0000 |
| 378 | Airbus380 |    53333.3333 |
+-----+-----------+---------------+
3 rows in set (0.01 sec)
```

5.Find the names of pilots certified for some Boeing aircraft.

```
mysql> select distinct e.ename
    -> from employees e,aircraft a,certified c
    -> where e.eid=c.eid
    -> and c.aid=a.aid
    -> and a.aname='Boeing';
+-------+
| ename |
+-------+
| Ajay  |
| Ron   |
| Ram   |
+-------+
3 rows in set (0.00 sec)
```

6.Find the aid's of all aircraft that can be used on routes from Bangalore to Delhi.

```
mysql> select a.aid
    -> from aircraft a
    -> where a.cruisingrange>
    -> (select min(f.distance)
    -> from flight f
    -> where f.frm='Bangalore'
    -> and f.too='Delhi');
+-----+
| aid |
+-----+
| 378 |
+-----+
1 row in set (0.00 sec)
```

6 . Order Processing Database

CUSTOMER (CUST #: INT, CNAME: STRING, CITY: STRING)

ORDER (ORDER #: INT, ODATE: DATE, CUST #: INT, ORD-AMT: INT)

ITEM (ITEM #: INT, UNIT PRICE: INT)

ORDER - ITEM (ORDER #: INT, ITEM #: INT, QTY: INT)

SHIPMENT (ORDER #: INT, WAREHOUSE#: INT, SHIP-DATE: DATE)

WAREHOUSE (WAREHOUSE #: INT, CITY: STRING)

1. **Create the above tables by mentioning the foreign keys**

```
CREATE TABLE CUSTOMER
(
CUST#   NUMBER(5),
CNAME VARCHAR2(15) NOT NULL,
CITY VARCHAR2(15),
PRIMARY KEY(CUST#)
);

CREATE TABLE CUSTORDER
(
 ORDER# NUMBER(5) CONSTRAINT PK1 PRIMARY KEY,
 ODATE DATE,
 CUST# NUMBER(5) REFERENCES CUSTOMER(CUST#),
 ORD_AMT NUMBER(5)
 );

CREATE TABLE ITEM
(
 ITEM  NUMBER(5) CONSTRAINT PK2 PRIMARY KEY,
UNITPRICE NUMBER(9,2) NOT NULL
 );

CREATE TABLE ORDER_ITEM
(
ORDER# NUMBER (5),
ITEM NUMBER (5),
QTY NUMBER(4),
FOREIGN KEY(ORDER#) REFERENCES CUSTORDER(ORDER#),
FOREIGN KEY(ITEM) REFERENCES ITEM(ITEM),
PRIMARY KEY(ORDER#,ITEM)
 );


 CREATE TABLE WAREHOUSE
 (
 WAREHOUSE#  NUMBER(5),
CITY VARCHAR2(15),
 PRIMARY KEY(WAREHOUSE#)
  );

CREATE TABLE SHIPMENT
 (
ORDER# NUMBER(5),
WAREHOUSE# NUMBER(5),
SHIP_DATE DATE,
FOREIGN KEY(ORDER#) REFERENCES CUSTORDER(ORDER#),
FOREIGN KEY(WAREHOUSE#) REFERENCES
                WAREHOUSE(WAREHOUSE#),
```

```
PRIMARY KEY(ORDER#,WAREHOUSE#)
 );
```

2) Insert the records into the relations

```
INSERT INTO CUSTOMER VALUES(&CUSTNO,'&CNAME','&CITY');
INSERT INTO CUSTORDER
        VALUES(&ORDER_NO,'&ODATE',&CUSTNO,&ORD_AMT);
INSERT INTO ITEM VALUES(&ITEM_NO,&UNITPRICE);
INSERT INTO ORDER_ITEM VALUES(&ORDER_NO,&ITEM_NO,&QTY);
INSERT INTO WAREHOUSE VALUES(&WAREHOUSE_NO,'&CITY');
INSERT INTO SHIPMENT
        VALUES(&ORDER_NO,&WAREHOUSE_NO,'&SHIPDATE');
```

3) VIEW THE RELATIONS

CUSTOMER

CUST#	CNAME	CITY
1	ABC	BLORE
2	DEF	KOLAR
3	GHI	BLORE
4	JKL	CHITTOR
5	MNO	MYSORE

ORDER

ORDER#	ODATE	CUST#	ORD_AMT
1	01-JAN-06	2	5000.5
2	26-APR-06	3	2500
3	27-APR-06	3	1000
4	30-APR-06	5	1000
5	25-MAY-06	1	5000

ITEM

ITEM#	UNITPRICE
1	2500
2	5000
3	1000
4	5
5	200

ORDER_ITEM

ORDER#	ITEM#	QTY
1	2	1
1	4	1
2	1	1
3	5	5
4	2	2

WAREHOUSE

WAREHOUSE#	CITY
1	BLORE
2	KOLAR
3	CHITTOR
4	MLORE
5	MYSORE

SHIPMENT

ORDER#	WAREHOUSE#	SHIPDATE
1	1	30-APR-06
2	2	29-APR-06
3	2	24-APR-06
4	5	30-APR-06
5	3	01-JUN-06
6	1	01-JUN-06

3) Producing the listing: custname, order#, avg_order_amt, where the middle column is the total no of orders made by the customer, and the last column is the average order amount for the customer

```
SELECT C.CNAME, COUNT(*),AVG(CO.ORD_AMT)
FROM CUSTOMER C, ORDER O
WHERE
C.CUST#=O.CUST# GROUP BY C.CNAME;
```

OUTPUT

CNAME	COUNT(*)	AVG
GHI	2	1750
MNO	1	10000
ABC	1	5000
DEF	1	5000.5

4) List the order# for orders that where shipped from all the warehouse that the company has in specific city.

```
SELECT  ORDER#
 FROM WAREHOUSE W,  SHIPMENT S
WHERE W.WAREHOUSE#=S.WAREHOUSE# AND CITY='BLORE';
```

ORDER#
1

5) Demonstrate how to delete item 10 from the item table and make that field null in the ORDER-ITEM table

```
SELECT CONSTRAINT_NAME,CONSTRAINT_TYPE
FROM USER_CONSTRAINTS
WHERE TABLE_NAME='ORDER_ITEM';
```

CONSTRAINT_NAME	C
SYS_C002734	P
SYS_C002735	R
SYS_C002736	R

```
ALTER TABLE ORDER_ITEM DROP CONSTRAINT SYS_C002736;

ALTER TABLE ORDER_ITEM ADD CONSTRAINT FK1 FOREIGN KEY(ITEM#)
REFERENCES ITEM(ITEM#) ON DELETE SET NULL;
```

DELETE FROM ITEM WHERE ITEM#=5;

SELECT * FROM ITEM

ITEM#	UNITPRICE
1	2500
2	5000
3	1000
4	-5

SELECT * FROM ORDER_ITEM

ORDER#	ITEM#	QTY
1	**2**	**1**
1	**4**	**1**
2	**1**	**1**
3		**5**
4	**2**	**2**

4. BOOK DEALER DATABASE

AUTHOR (AUTHOR-ID: INT, NAME: STRING, CITY: STRING, COUNTRY: STRING)
PUBLISHER (PUBLISHER-ID: INT, NAME: STRING, CITY: STRING, COUNTRY: STRING)
CATALOG (BOOK-ID: INT, TITLE: STRING, AUTHOR-ID: INT, PUBLISHER-ID: INT, CATEGORY-ID: INT,
YEAR: INT, PRICE: INT)
CATEGORY (CATEGORY-ID: INT, DESCRIPTION: STRING)
ORDER-DETAILS (ORDER-NO: INT, BOOK-ID: INT, QUANTITY: INT)

1) **Create the above tables by properly specifying the primary keys and the foreign keys**

```
CREATE TABLE AUTHOR
(
 AUTHORID NUMBER(5),
 NAME VARCHAR(15),
 CITY VARCHAR2(15),
 ACOUNTRY VARCHAR(15),
 PRIMARY KEY(AUTHORID)
);

CREATE TABLE PUBLISHER
(
 PUBLISHERID NUMBER(5),
NAMEVARCHAR(15),
 CITY VARCHAR2(15),
 COUNTRY VARCHAR(15),
 PRIMARY KEY(PUBLISHERID)
 );

CREATE TABLE CATEGORY
(
 CATEGORYID NUMBER(5),
 DESCRIPTION VARCHAR(15),
 PRIMARY KEY(CATEGORYID)
 );

 CREATE TABLE CATALOG
 (
  BOOKID NUMBER(5),
 TITLE VARCHAR2(15),
 AUTHORID NUMBER(5),
 PUBLISHERID NUMBER(5),
 CATEGORYID NUMBER(5),
 YEAR NUMBER(5),
 PRICE NUMBER(10,3),
 PRIMARY KEY(BOOKID),
 FOREIGN KEY(AUTHORID) REFERENCES AUTHOR(AUTHORID),
 FOREIGN KEY(PUBLISHERID) REFERENCES PUBLISHER(PUBLISHERID),
 FOREIGN KEY(CATEGORYID) REFERENCES CATEGORY(CATEGORYID)
 );

 CREATE TABLE ORDER_DETAILS
 (
  ORDERNO NUMBER(5),
  BOOKID NUMBER(5),
 QUANTITY NUMBER(5),
 PRIMARY KEY(ORDERNO,BOOKID),
 FOREIGN KEY(BOOKID) REFERENCES CATALOG(BOOKID)
```

);

2) INSERT THE RECORDS INTO THE RELATIONS

```
INSERT INTO AUTHOR
        VALUES(&AUTHORID,'&NAME','&CITY','&ACOUNTRY');
INSERT INTO PUBLISHER
        VALUES(&PUBLISHERID,'&NAME','&CITY','&COUNTRY');
INSERT INTO CATEGORY VALUES(&CATEGORYID,'&DESCRIPTION');
INSERT INTO CATALOG VALUES
       (&BOOKID,'&TITLE',&AUTHORID,&PUBLISHERID,&CATEGORYID,&YEAR,
        &PRICE);
INSERT INTO ORDER_DETAILS VALUES(&ORDERNO,&BOOKID,&QUANTITY);
```

3) VIEW THE RELATIONS

AUTHOR

AUTHORID	NAME	CITY	COUNTRY
101	ABC	DELHI	INDIA
102	TONY	HAYHOOD	USA
103	GHI	PATNA	INDIA
104	JKL	BELM	SRILANKA
105	MND	BANGALORE	INDIA

PUBLISHER

PUBLISHERID	NAME	CITY	COUNTRY
1001	pbp	blore	INDIA
1002	palk	slaugh	england
1003	press	tata	INDIA
1004	rathe	angakara	srilanka
1005	pbp	blore	india

CATEGORY

CATEGORYID	DESCRIPTIONM
10001	cs
10002	med
10003	bio
10004	meteor
10005	mech

CATALOG

BOOKID	TITLE	AUTHORID	PUBLISHERID	CATEGORYID	YEAR	PRICE
1000001	dbms	101	1001	10001	1998	235
1000002	or	101	1002	10003	1997	255
1000003	cn	102	1003	10002	2001	352
1000004	se	102	1003	10001	2002	523
1000005	ada	103	1004	10004	2003	124

ORDER_DETAILS

ORDER-NO	BOOK-ID	QUANTITY
1	1000001	12
1	1000002	2
2	1000002	15
3	1000003	23
4	1000003	14
5	1000005	7

4) GIVE THE DETAILS OF THE AUTHORS WHO HAVE TWO OR MORE BOOKS IN THE CATALOG AND THE PRICE OF THE BOOKS IS GREATER THAN THE AVERAGE PRICE OF THE BOOKS IN THE CATALOG & THE YEAR OF PUBLICATION IS AFTER 2000.

```
SELECT * FROM AUTHOR A WHERE A.AUTHORID IN
        (SELECT C.AUTHORID FROM CATALOG C
                WHERE YEAR>2000 AND
                    C.PRICE > (SELECT AVG (PRICE) FROM CATALOG)
                                GROUP BY C.AUTHORID HAVING COUNT (AUTHOR-ID)>=2);
```

Output

AUTHOR-ID	NAME	CITY	COUNTRY
102	DEF	JAMES	INDIA

5) FIND THE AUTHOR OF THE BOOK WHICH HAS MAXIMUN SALES

```
SELECT NAME FROM AUTHOR WHERE AUTHORID IN
  (
```

SELECT AUTHORID FROM CATALOG , ORDER_DETAILS O WHERE O.BOOK-ID=CATALOG.BOOK-ID AND QUANTITY=9SELECT MAX(QUANTITY) FROM OREDER-DETAILS));

NAME

DEF

6) DEMONSTRATE HOW YOU INCREASE THE PRICE OF BOOK PUBLISHED BY A SPECIFIC PUBLISHER BY 10%.

```
UPDATE CATALOG SET PRICE=PRICE+PRICE*0.10
  WHERE PUBLISHERID IN (SELECT P.PUBLISHERID
                                    FROM PUBLISHER P
                                        WHERE P.NAME=''PBP');
```

Output

SQL>SELECT * FROM CATALOG;

BOOKID	TITLE	AUTHORID	PUBLISHERID	CATEGORYID	YEAR	PRICE
1000001	dbms	101	1001	10001	1998	258
1000002	Or	101	1002	10003	1997	255
1000003	Cn	102	1003	10002	2001	352
1000004	Se	102	1003	10001	2002	523
1000005	ada	103	1004	10004	2003	124

8. STUDENT ENROLLMENT DATABASE

STUDENT (REGNO: STRING, NAME: STRING, MAJOR: STRING, BDATE :DATE)

COURSE (COURSE #: INT, CNAME: STRING, DEPT: STRING)

ENROLL (REGNO: STRING, COURSE#: INT, SEM: INT, MARKS: INT)

BOOK - ADOPTION (COURSE#: INT, SEM: INT, BOOK-ISBN: INT)

TEXT (BOOK-ISBN: INT, BOOK-TITLE: STRING, PUBLISHER:STRING, AUTHOR: STRING)

1) Create the above tables by properly specifying the primary keys and the foreign keys

CREATE TABLE STUDENT

```
(
 REGNO VARCHAR2(10),
 NAME VARCHAR2(14),
 MAJOR VARCHAR2(10),
 BDATE DATE,
 PRIMARY KEY(REGNO)
);

CREATE TABLE COURSE
(
 COURSE# NUMBER(4),
 CNAME VARCHAR2(14),
 DEPT VARCHAR2(10),
 PRIMARY KEY(COURSE#)
);

CREATE TABLE ENROLL
(
 REGNO VARCHAR2(10),
 COURSE# NUMBER(4),
 SEM NUMBER(4),
 MARKS NUMBER(3),
 PRIMARY KEY(REGNO,COURSE#,SEM),
 FOREIGN KEY(REGNO) REFERENCES STUDENT(REGNO),
 FOREIGN KEY(COURSE#) REFERENCES COURSE(COURSE#)
);
CREATE TABLE TEXT
(
 ISBN NUMBER(5),
 BOOK_TITLE VARCHAR2(13) NOT NULL,
 PUBLISHER VARCHAR2(12),
 AUTHOR VARCHAR2(12),
 PRIMARY KEY(ISBN)
);
```

CREATE TABLE BOOK_ADOPTION

```
(
 COURSE# NUMBER(5),
 SEM NUMBER(3),
```

```
 ISBN NUMBER(4),
 PRIMARY KEY(COURSE#,SEM),
 FOREIGN KEY(COURSE#) REFERENCES COURSE(COURSE#),
 FOREIGN KEY(ISBN) REFERENCES TEXT(ISBN)
);
```

2)INSERTION OF RECORDS INTO THE RELATIONS

- INSERT INTO STUDENT VALUES('&REGNO','&NAME','&MAJOR','&BDATE');
- INSERT INTO COURSE VALUES(&COURSE#,'&CNAME','&DEPT');
- INSERT INTO ENROLL VALUES('&REGNO',&COURSE#,&SEM,&MARKS);
- INSERT INTO TEXT VALUES(&ISBN,'&BOOK_TITLE','&PUBLISHER','&AUTHOR');
- INSERT INTO BOOK_ADOPTION VALUES(&COURSE#,&SEM,&ISBN);

VIEW THE RECORDS OF THE RELATIONS

STUDENT

REGNO	NAME	MAJOR	BDATE
1DA05CS045	A	BIOLOGY	25-DEC-84
1DA05CS062	B	CHE	23-JAN-86
1DA05CS015	C	PHYSICS	20-JUN-86
1DA05CS025	D	MA	30-MAR-84
1DA05IS405	E	BIO	06-APR-84

COURSE

COURSE#	CNAME	DEPT
1	MCA	MANAG
2	MBA	MANAG
3	ISE	CS
4	CSE	CS
5	CIV	CIVIL

ENROLL

REGNO	COURSE#	SEM	MARKS
1DA05CS062	2	3	85
1DA05CS015	3	5	57
1DA05CS025	4	8	92
1DA05CS025	4	4	91
1DA05IS405	5	2	85

TEXT

BOOK_ISBN	BOOK_TITLE	PUBLISHER	AUTHOR
111001	DBMS	TATA	NAVATHE
111002	CN	PDR	TENENBAUM
111003	DS	MC-MIILAN	GALVIN
111004	ADA	PERSON	ULLMAN
111005	SE	PRESS	PRESSMAN

BOOK_ADOPTION

COURSE#	SEM	BOOK_ISBN
1	1	111001
2	2	111004
3	4	111003
4	1	111005
5	5	111006
6	4	111003
3	5	111001

3) DEMONSTRATE HOW YOU ADD NEW TEXT BOOK TO THE DATABASE AND MAKE THAT BOOK IS ADOPTED BY SOME DEPARTMENT.

```
INSERT INTO TEXT VALUES(111006,'AMP','MHP,'BERY);
```

```
INSERT INTO BOOK_ADOPTION VALUES(5,5,111006);
```

4) PRODUCE A LIST OF TEXTBOOKS(INCLUDE COURSE,BOOK_ISBN,BOOK_TITLE) IN THE ALPHABETIC ORDER FOR COURSES OFFERED BY THE 'CSE' DEPT THAT USE MORE THAN TWO BOOKS.

```
SELECT C.COURSE#, B.ISBN,BOOK_TITLE
    FROM COURSE C,BOOK_ADOPTION B,TEXT T
      WHERE C.COURSE#=B.COURSE# AND B.ISBN=T.ISBN  AND
        C.COURSE# IN   (SELECT C1.COURSE# FROM COURSE
            C1,BOOK_ADOPTION B1
                  WHERE
              C1.COURSE#=B1.COURSE# AND DEPT='CSE'
                GROUP BY C1.COURSE#  HAVING COUNT(*)>2)
               ORDER BY CNAME;
```

OUTPUT

COURSE#	BOOK_ISBN	BOOK_TITLE
3	111003	CS
3	111001	DBMS
4	111005	SE
4	111003	OS
4	111006	AMP

5) LIST ANY DEPARTMENT THAT HAS ALL ITS ADOPTED BOOKS PUBLISHED BY A SPECIFIC PUBLISHER

```
SELECT DISTINCT C.DEPT
 FROM COURSE C, BOOK_ADOPT B, TEXT T
  WHERE T.BOOK_ISBN=B.BOOK_ISBN
   AND B.COURSE#=C.COURSE#
                AND T.PUBLISHER='TATA';
```

OUTPUT

DEPT
CS

9. MOVIE DATABASE

ACTOR (*Act_id, Act_Name, Act_Gender*)
DIRECTOR (*Dir_id, Dir_Name, Dir_Phone*)
MOVIES (*Mov_id, Mov_Title, Mov_Year, Mov_Lang, Dir_id*)
MOVIE_CAST (*Act_id, Mov_id, Role*)
RATING (*Mov_id, Rev_Stars*)

Write SQL queries to

1. **List the titles of all movies directed by 'Hitchcock'.**
2. **Find the movie names where one or more actors acted in two or more movies.**
3. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**
4. **Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**
5. **Update rating of all movies directed by 'Steven Spielberg' to 5.**

Schema Diagram

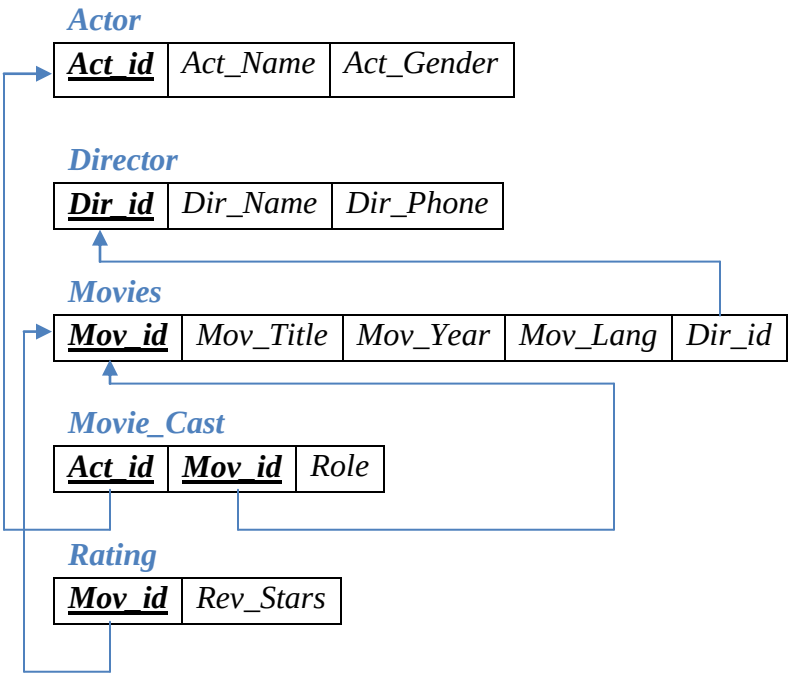


Table Creation

```
CREATE TABLE ACTOR (
ACT_ID NUMBER (3),
ACT_NAME  VARCHAR (20),
ACT_GENDER CHAR (1),
PRIMARY KEY (ACT_ID));

CREATE TABLE DIRECTOR (
DIR_ID NUMBER (3),
DIR_NAME VARCHAR (20),
DIR_PHONE  NUMBER (10),
PRIMARY KEY (DIR_ID));

CREATE TABLE MOVIES (
MOV_ID NUMBER (4),
MOV_TITLE  VARCHAR (25),
MOV_YEAR  NUMBER (4),
MOV_LANG  VARCHAR (12),
DIR_ID NUMBER (3),
PRIMARY KEY (MOV_ID),
FOREIGN KEY (DIR_ID) REFERENCES DIRECTOR (DIR_ID));
```

```
CREATE TABLE MOVIE_CAST (
ACT_ID NUMBER (3),
MOV_ID NUMBER (4),
ROLE  VARCHAR (10),
PRIMARY KEY (ACT_ID, MOV_ID),
FOREIGN KEY (ACT_ID) REFERENCES ACTOR (ACT_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));

CREATE TABLE RATING (
MOV_ID NUMBER (4),
REV_STARS VARCHAR (25),
PRIMARY KEY (MOV_ID),
FOREIGN KEY (MOV_ID) REFERENCES MOVIES (MOV_ID));
```

Table Descriptions

DESC ACTOR;

```
SQL> DESC ACTOR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACT_ID                                    NOT NULL NUMBER(3)
 ACT_NAME                                           VARCHAR2(20)
 ACT_GENDER                                         CHAR(1)
```

DESC DIRECTOR;

```
SQL> DESC DIRECTOR;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DIR_ID                                    NOT NULL NUMBER(3)
 DIR_NAME                                           VARCHAR2(20)
 DIR_PHONE                                          NUMBER(10)
```

DESC MOVIES;

```
SQL> DESC MOVIES;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MOV_ID                                    NOT NULL NUMBER(4)
 MOV_TITLE                                          VARCHAR2(25)
 MOV_YEAR                                           NUMBER(4)
 MOV_LANG                                           VARCHAR2(12)
 DIR_ID                                             NUMBER(3)
```

DESC MOVIE_CAST;

```
SQL> DESC MOVIE_CAST;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 ACT_ID                                    NOT NULL NUMBER(3)
 MOV_ID                                    NOT NULL NUMBER(4)
 ROLE                                               VARCHAR2(10)
```

DESC RATING;

```
SQL> DESC RATING;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 MOV_ID                                    NOT NULL NUMBER(4)
 REV_STARS                                          VARCHAR2(25)
```

Insertion of Values to Tables

INSERT INTO ACTOR VALUES (301,‘ANUSHKA‘,‘F‘);
INSERT INTO ACTOR VALUES (302,‘PRABHAS‘,‘M‘);
INSERT INTO ACTOR VALUES (303,‘PUNITH‘,‘M‘);
INSERT INTO ACTOR VALUES (304,‘JERMY‘,‘M‘);

INSERT INTO DIRECTOR VALUES (60,‘RAJAMOULI‘, 8751611001);
INSERT INTO DIRECTOR VALUES (61,‘HITCHCOCK‘, 7766138911);
INSERT INTO DIRECTOR VALUES (62,‘FARAN‘, 9986776531);
INSERT INTO DIRECTOR VALUES (63,‘STEVEN SPIELBERG‘, 8989776530);

INSERT INTO MOVIES VALUES (1001,‘BAHUBALI-2‘, 2017, ‗TELAGU‘, 60);
INSERT INTO MOVIES VALUES (1002,‘BAHUBALI-1‘, 2015, ‗TELAGU‘, 60);
INSERT INTO MOVIES VALUES (1003,‘AKASH‘, 2008, ‗KANNADA‘, 61);
INSERT INTO MOVIES VALUES (1004,‘WAR HORSE‘, 2011, ‗ENGLISH‘, 63);

INSERT INTO MOVIE_CAST VALUES (301, 1002, ‗HEROINE‘);
INSERT INTO MOVIE_CAST VALUES (301, 1001, ‗HEROINE‘);
INSERT INTO MOVIE_CAST VALUES (303, 1003, ‗HERO‘);
INSERT INTO MOVIE_CAST VALUES (303, 1002, ‗GUEST‘);
INSERT INTO MOVIE_CAST VALUES (304, 1004, ‗HERO‘);

INSERT INTO RATING VALUES (1001, 4);
INSERT INTO RATING VALUES (1002, 2);

INSERT INTO RATING VALUES (1003, 5);
INSERT INTO RATING VALUES (1004, 4);

SELECT * FROM ACTOR;

```
SQL> SELECT * FROM ACTOR;

    ACT_ID ACT_NAME             A
---------- -------------------- -
       301 ANUSHKA              F
       302 PRABHAS              M
       303 PUNITH               M
       304 JERMY                M
```

SELECT * FROM DIRECTOR;

```
SQL> SELECT * FROM DIRECTOR;

    DIR_ID DIR_NAME              DIR_PHONE
---------- -------------------- ----------
        60 RAJAMOULI            8751611001
        61 HITCHCOCK            7766138911
        62 FARAN                9986776531
        63 STEVEN SPIELBERG     8989776530
```

SELECT * FROM MOVIES;

```
SQL> SELECT * FROM MOVIES;

    MOV_ID MOV_TITLE                MOV_YEAR MOV_LANG         DIR_ID
---------- ---------------------- ---------- ------------ ----------
      1001 BAHUBALI-2                   2017 TELAGU               60
      1002 BAHUBALI-1                   2015 TELAGU               60
      1003 AKASH                        2008 KANNADA              61
      1004 WAR HORSE                    2011 ENGLISH              63
```

SELECT * FROM MOVIE_CAST;

```
SQL> SELECT * FROM MOVIE_CAST;

    ACT_ID     MOV_ID ROLE
---------- ---------- ----------
       301       1002 HEROINE
       301       1001 HEROINE
       303       1003 HERO
       303       1002 GUEST
       304       1004 HERO
```

SELECT * FROM RATING;

```
SQL> SELECT * FROM RATING;

    MOV_ID REV_STARS
---------- -----------
      1001 4
      1002 2
      1003 5
      1004 4
```

Queries:

1. **List the titles of all movies directed by 'Hitchcock'.**

```
SELECT MOV_TITLE
FROM MOVIES
WHERE DIR_ID IN (SELECT DIR_ID
                 FROM DIRECTOR
                 WHERE DIR_NAME = _HITCHCOCK');
```

```
MOV_TITLE
-------------------------
AKASH
```

2. **Find the movie names where one or more actors acted in two or more movies.**

```
SELECT MOV_TITLE
FROM MOVIES M, MOVIE_CAST MV
WHERE M.MOV_ID=MV.MOV_ID AND ACT_ID IN (SELECT ACT_ID
                                        FROM MOVIE_CAST GROUP BY ACT_ID
                                        HAVING COUNT (ACT_ID)>1)
GROUP BY MOV_TITLE
HAVING COUNT (*)>1;
```

```
MOV_TITLE
------------------------
BAHUBALI-1
```

3. **List all actors who acted in a movie before 2000 and also in a movie after 2015 (use JOIN operation).**

```
SELECT ACT_NAME, MOV_TITLE, MOV_YEAR
```

```
FROM ACTOR A
JOIN MOVIE_CAST C
        ON A.ACT_ID=C.ACT_ID
JOIN MOVIES   M
        ON C.MOV_ID=M.MOV_ID
WHERE M.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

OR

```
SELECT A.ACT_NAME, A.ACT_NAME, C.MOV_TITLE, C.MOV_YEAR
FROM ACTOR A, MOVIE_CAST B, MOVIES C
WHERE A.ACT_ID=B.ACT_ID
AND B.MOV_ID=C.MOV_ID
AND C.MOV_YEAR NOT BETWEEN 2000 AND 2015;
```

ACT_NAME	MOV_TITLE	MOV_YEAR
ANUSHKA	BAHUBALI-2	2017

4. **Find the title of movies and number of stars for each movie that has at least one rating and find the highest number of stars that movie received. Sort the result by movie title.**

```
SELECT MOV_TITLE, MAX (REV_STARS)
FROM MOVIES
INNER JOIN RATING USING (MOV_ID)
GROUP BY MOV_TITLE
HAVING MAX (REV_STARS)>0
ORDER BY MOV_TITLE;
```

MOV_TITLE	MAX(REV_STARS)
AKASH	5
BAHUBALI-1	2
BAHUBALI-2	4
WAR HORSE	4

5. Update rating of all movies directed by 'Steven Spielberg' to 5

KL

```
UPDATE RATING
SET REV_STARS=5
WHERE MOV_ID IN (SELECT MOV_ID FROM MOVIES
                 WHERE DIR_ID IN (SELECT DIR_ID
                                  FROM DIRECTOR
                                  WHERE DIR_NAME = _STEVEN
                                  SPIELBERG'));
```

```
SQL> SELECT * FROM RATING;

    MOV_ID REV_STARS
---------- ------------------------
      1001 4
      1002 2
      1003 5
      1004 5
```

---

10. COLLEGE DATABASE

STUDENT (*USN, SName, Address, Phone, Gender*)
SEMSEC (*SSID, Sem, Sec*)
CLASS (*USN, SSID*)
SUBJECT (*Subcode, Title, Sem, Credits*)
IAMARKS (*USN, Subcode, SSID, Test1, Test2, Test3, FinalIA*)

Write SQL queries to :

1. **List all the student details studying in fourth semester 'C' section.**
2. **Compute the total number of male and female students in each semester and in each section.**
3. **Create a view of Test1 marks of student USN '1BI15CS101' in all subjects.**
4. **Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.**
5. **Categorize students based on the following criterion:**
   If FinalIA = 17 to 20 then CAT = 'Outstanding'
   If FinalIA = 12 to 16 then CAT = 'Average'
   If FinalIA< 12 then CAT = 'Weak'
   Give these details only for 8th semester A, B, and C section students.

Schema Diagram

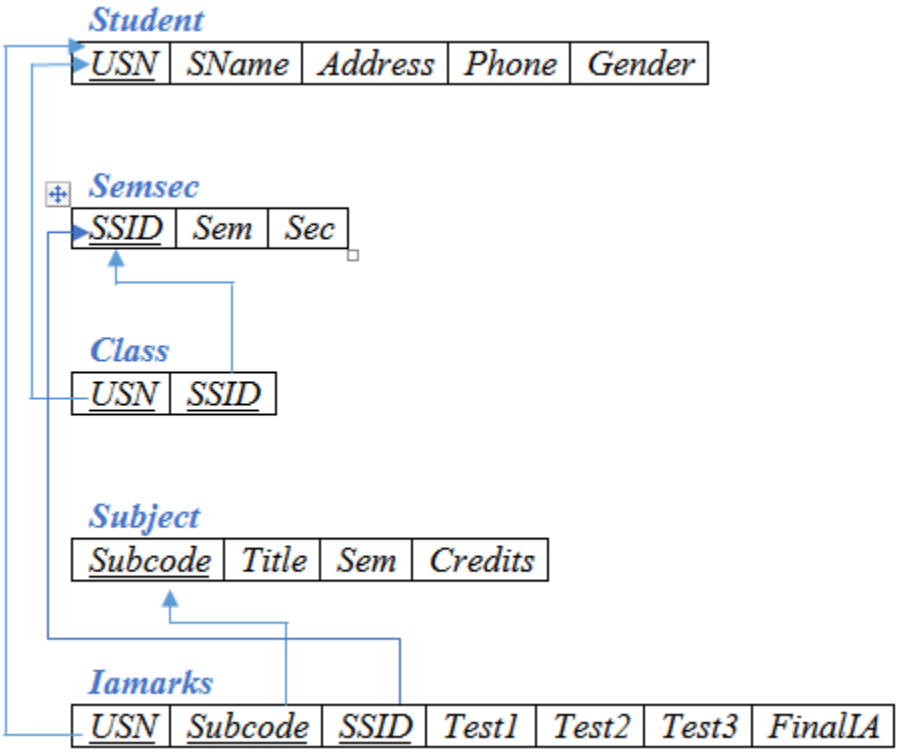


Table Creation

```
CREATE TABLE STUDENT (
 USN VARCHAR (10) PRIMARY KEY,
 SNAME VARCHAR (25),
 ADDRESS VARCHAR (25),
 PHONE NUMBER (10),
 GENDER CHAR (1));

CREATE TABLE SEMSEC (
SSID VARCHAR (5) PRIMARY KEY,
SEM NUMBER (2),
SEC CHAR (1));

CREATE TABLE CLASS (
USN VARCHAR (10),
SSID VARCHAR (5),
PRIMARY KEY (USN, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN),
FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
```

```
CREATE TABLE SUBJECT (
SUBCODE VARCHAR (8),
TITLE VARCHAR (20),
SEM NUMBER (2),
CREDITS NUMBER (2),
PRIMARY KEY (SUBCODE));

 CREATE TABLE IAMARKS (
 USN VARCHAR (10),
 SUBCODE VARCHAR (8),
 SSID VARCHAR (5),
 TEST1 NUMBER (2),
 TEST2 NUMBER (2),
 TEST3 NUMBER (2),
 FINALIA NUMBER (2),
 PRIMARY KEY (USN, SUBCODE, SSID),
FOREIGN KEY (USN) REFERENCES STUDENT (USN),
 FOREIGN KEY (SUBCODE) REFERENCES SUBJECT (SUBCODE),
 FOREIGN KEY (SSID) REFERENCES SEMSEC (SSID));
```

Table Descriptions

DESC STUDENT;

```
Name
--------------------------------------------------------------------------------------------
USN
SNAME
ADDRESS
PHONE
GENDER
```

DESC SEMSEC;

```
SQL> DESC SEMSEC;
 Name
 ------------------------------------------------------------------------------
 SSID
 SEM
 SEC
```

DESC CLASS;

```
SQL> DESC CLASS;
 Name
 -----------------------------------------
 USN
 SSID
```

DESC SUBJECT;

```
SQL> DESC SUBJECT1;
 Name
 ----------------------------------
 SUBCODE
 TITLE
 SEM
 CREDITS
```

DESC IAMARKS;

```
SQL> DESC IAMARKS;
 Name
 -----------------------------------------------
 USN
 SUBCODE
 SSID
 TEST1
 TEST2
 TEST3
 FINALIA
```

Insertion of values to tables

INSERT INTO STUDENT VALUES ('1RN13CS020','AKSHAY','BELAGAVI', 8877881122,'M');
INSERT INTO STUDENT VALUES ('1RN13CS062','SANDHYA','BENGALURU', 7722829912,'F');
INSERT INTO STUDENT VALUES ('1RN13CS091','TEESHA','BENGALURU', 7712312312,'F');
INSERT INTO STUDENT VALUES ('1RN13CS066','SUPRIYA','MANGALURU', 8877881122,'F');
INSERT INTO STUDENTVALUES ('1RN14CS010','ABHAY','BENGALURU', 9900211201,'M');
INSERT INTO STUDENT VALUES ('1RN14CS032','BHASKAR','BENGALURU', 9923211099,'M');
INSERT INTO STUDENTVALUES ('1RN14CS025','ASMI','BENGALURU', 7894737377,'F');
INSERT INTO STUDENT VALUES ('1RN15CS011','AJAY','TUMKUR', 9845091341,'M');

```
INSERT INTO STUDENT VALUES ('1RN15CS029','CHITRA','DAVANGERE',
7696772121,'F');
INSERT INTO STUDENT VALUES ('1RN15CS045','JEEVA','BELLARY', 9944850121,'M');
INSERT INTO STUDENT VALUES ('1RN15CS091','SANTOSH','MANGALURU',
8812332201,'M');
INSERT INTO STUDENT VALUES ('1RN16CS045','ISMAIL','KALBURGI',
9900232201,'M');
INSERT INTO STUDENT VALUES ('1RN16CS088','SAMEERA','SHIMOGA',
9905542212,'F');
INSERT INTO STUDENT VALUES ('1RN16CS122','VINAYAKA','CHIKAMAGALUR',
8800880011,'M');

INSERT INTO SEMSEC VALUES ('CSE8A', 8,'A');
INSERT INTO SEMSEC VALUES (_CSE8B', 8,'B');
INSERT INTO SEMSEC VALUES (_CSE8C‘, 8,‘C‘);

INSERT INTO SEMSEC VALUES ('CSE7A', 7,‘A‘);
INSERT INTO SEMSEC VALUES (_CSE7B‘, 7,'B‘);
INSERT INTO SEMSEC VALUES ('CSE7C', 7,'C');

INSERT INTO SEMSEC VALUES (_CSE6A', 6,'A');
INSERT INTO SEMSEC VALUES (_CSE6B‘, 6,‘B‘);
INSERT INTO SEMSEC VALUES ('CSE6C‘, 6,‘C‘);

INSERT INTO SEMSEC VALUES (_CSE5A‘, 5,'A‘);
INSERT INTO SEMSEC VALUES ('CSE5B', 5,'B');
INSERT INTO SEMSEC VALUES (_CSE5C', 5,'C');

INSERT INTO SEMSEC VALUES (_CSE4A‘, 4,‘A‘);
INSERT INTO SEMSEC VALUES ('CSE4B', 4,‘B‘);
INSERT INTO SEMSEC VALUES (_CSE4C‘, 4,'C‘);

INSERT INTO SEMSEC VALUES ('CSE3A', 3,'A');
INSERT INTO SEMSEC VALUES (_CSE3B', 3,'B');
INSERT INTO SEMSEC VALUES (_CSE3C‘, 3,‘C‘);

INSERT INTO SEMSEC VALUES ('CSE2A', 2,‘A‘);
INSERT INTO SEMSEC VALUES (_CSE2B‘, 2,'B‘);
INSERT INTO SEMSEC VALUES ('CSE2C', 2,'C');
INSERT INTO SEMSEC VALUES (_CSE1A', 1,'A');
```

```
INSERT INTO SEMSEC VALUES (_CSE1B‘, 1,‘B‘);
INSERT INTO SEMSEC VALUES ('CSE1C', 1,‘C‘);

INSERT INTO CLASS VALUES (_1RN13CS020‘,‘CSE8A‘);
INSERT INTO CLASS VALUES (_1RN13CS062‘,‘CSE8A‘);
INSERT INTO CLASS VALUES (_1RN13CS066‘,‘CSE8B‘);
INSERT INTO CLASS VALUES (_1RN13CS091‘,‘CSE8C‘);

INSERT INTO CLASS VALUES (_1RN14CS010‘,‘CSE7A‘);
INSERT INTO CLASS VALUES (_1RN14CS025‘,‘CSE7A‘);
INSERT INTO CLASS VALUES (_1RN14CS032‘,‘CSE7A‘);

INSERT INTO CLASS VALUES (_1RN15CS011‘,‘CSE4A‘);
INSERT INTO CLASS VALUES (_1RN15CS029‘,‘CSE4A‘);
INSERT INTO CLASS VALUES (_1RN15CS045‘,‘CSE4B‘);
INSERT INTO CLASS VALUES (_1RN15CS091‘,‘CSE4C‘);

INSERT INTO CLASS VALUES (_1RN16CS045‘,‘CSE3A‘);
INSERT INTO CLASS VALUES (_1RN16CS088‘,‘CSE3B‘);
INSERT INTO CLASS VALUES (_1RN16CS122‘,‘CSE3C‘);

INSERT INTO SUBJECT VALUES ('10CS81','ACA', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS82','SSM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS83','NM', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS84','CC', 8, 4);
INSERT INTO SUBJECT VALUES ('10CS85','PW', 8, 4);

INSERT INTO SUBJECT VALUES ('10CS71','OOAD', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS72','ECS', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS73','PTW', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS74','DWDM', 7, 4);
INSERT INTO SUBJECT VALUES (_10CS75','JAVA', 7, 4);
INSERT INTO SUBJECT VALUES ('10CS76','SAN', 7, 4);

INSERT INTO SUBJECT VALUES ('15CS51', 'ME', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS52','CN', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS53','DBMS', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS54','ATC', 5, 4);
INSERT INTO SUBJECT VALUES ('15CS55','JAVA', 5, 3);
INSERT INTO SUBJECT VALUES ('15CS56','AI', 5, 3);
```

```
INSERT INTO SUBJECT VALUES ('15CS41','M4', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS42','SE', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS43','DAA', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS44','MPMC', 4, 4);
INSERT INTO SUBJECT VALUES ('15CS45','OOC', 4, 3);
INSERT INTO SUBJECT VALUES ('15CS46','DC', 4, 3);

INSERT INTO SUBJECT VALUES ('15CS31','M3', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS32','ADE', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS33','DSA', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS34','CO', 3, 4);
INSERT INTO SUBJECT VALUES ('15CS35','USP', 3, 3);
INSERT INTO SUBJECT VALUES ('15CS36','DMS', 3, 3);

INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS81','CSE8C', 15, 16, 18);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS82','CSE8C', 12, 19, 14);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS83','CSE8C', 19, 15, 20);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS84','CSE8C', 20, 16, 19);
INSERT INTO IAMARKS (USN, SUBCODE, SSID, TEST1, TEST2, TEST3) VALUES
('1RN13CS091','10CS85','CSE8C', 15, 15, 12);

SELECT * FROM STUDENT;
```

```
SQL> SELECT * FROM STUDENT1;

USN        SNAME                     ADDRESS                        PHONE G
---------- ------------------------- ------------------------- ---------- -
1RN13CS020 AKSHAY                    BELAGAVI                  8877881122 M
1RN13CS062 SANDHYA                   BENGALURU                 7722829912 F
1RN13CS091 TEESHA                    BENGALURU                 7712312312 F
1RN13CS066 SUPRIYA                   MANGALURU                 8877881122 F
1RN14CS010 ABHAY                     BENGALURU                 9900211201 M
1RN14CS032 BHASKAR                   BENGALURU                 9923211099 M
1RN15CS011 AJAY                      TUMKUR                    9845091341 M
1RN15CS029 CHITRA                    DAVANGERE                 7696772121 F
1RN15CS045 JEEVA                     BELLARY                   9944850121 M
1RN15CS091 SANTOSH                   MANGALURU                 8812332201 M
1RN16CS045 ISMAIL                    KALBURGI                  9900232201 M
1RN16CS088 SAMEERA                   SHIMOGA                   9905542212 F
1RN16CS122 VINAYAKA                  CHIKAMAGALUR              8800880011 M
1RN14CS025 ASMI                      BENGALURU                 7894737377 F
```

SELECT * FROM SEMSEC;

```
SQL> SELECT * FROM SEMSEC;

SSID         SEM S
----- ---------- -
CSE8A          8 A
CSE8B          8 B
CSE8C          8 C
CSE7A          7 A
CSE7B          7 B
CSE7C          7 C
CSE6A          6 A
CSE6B          6 B
CSE6C          6 C
CSE5A          5 A
CSE5B          5 B
CSE5C          5 C
CSE4A          4 A
CSE4B          4 B
CSE4C          4 C
CSE3A          3 A
CSE3B          3 B
CSE3C          3 C
CSE2A          2 A
CSE2C          2 C
CSE2B          2 B
CSE1A          1 A
CSE1B          1 B
CSE1C          1 C
```

SELECT * FROM CLASS;

```
SQL> SELECT * FROM CLASS;

USN        SSID
---------- -----
1RN13CS020 CSE8A
1RN13CS062 CSE8A
1RN13CS066 CSE8B
1RN13CS091 CSE8C
1RN14CS010 CSE7A
1RN14CS025 CSE7A
1RN14CS032 CSE7A
1RN15CS011 CSE4A
1RN15CS029 CSE4A
1RN15CS045 CSE4B
1RN15CS091 CSE4C
1RN16CS045 CSE3A
1RN16CS088 CSE3B
1RN16CS122 CSE3C

14 rows selected.
```

SELECT * FROM SUBJECT;

SUBCODE	TITLE	SEM	CREDITS
10CS81	ACA	8	4
10CS82	SSM	8	4
10CS83	NM	8	4
10CS84	CC	8	4
10CS85	PW	8	4
10CS71	OOAD	7	4
10CS72	ECS	7	4
10CS73	PTW	7	4
10CS74	DWDM	7	4
10CS75	JAVA	7	4
10CS76	SAN	7	4
15CS51	ME	5	4
15CS52	CN	5	4
15CS53	DBMS	5	4
15CS54	ATC	5	4
15CS55	JAVA	5	3
15CS56	AI	5	3
15CS41	M4	4	4
15CS42	SE	4	4
15CS43	DAA	4	4
15CS44	MPMC	4	4
15CS45	OOC	4	3
15CS46	DC	4	3
15CS31	M3	3	4
15CS32	ADE	3	4
15CS33	DSA	3	4
15CS34	CO	3	4
15CS35	USP	3	3
15CS36	DMS	3	3

SELECT * FROM IAMARKS;

```
SQL> SELECT * FROM IAMARKS;
```

USN	SUBCODE	SSID	TEST1	TEST2	TEST3	FINALIA
1RN13CS091	10CS81	CSE8C	15	16	18	
1RN13CS091	10CS82	CSE8C	12	19	14	
1RN13CS091	10CS83	CSE8C	19	15	20	
1RN13CS091	10CS84	CSE8C	20	16	19	
1RN13CS091	10CS85	CSE8C	15	15	12	

Queries:

1. **List all the student details studying in fourth semester 'C' section.**

```
SELECT S.*, SS.SEM, SS.SEC
 FROM STUDENT S, SEMSEC SS, CLASS C
 WHERE S.USN = C.USN AND
 SS.SSID = C.SSID AND
 SS.SEM = 4 AND
```

SS.SEc=‘C‘;

USN	SNAME	ADDRESS	PHONE	G	SEM	S
1RN15CS091	SANTOSH	MANGALURU	8812332201	M	4	C

2. **Compute the total number of male and female students in each semester and in each section.**

```
SELECT SS.SEM, SS.SEC, S.GENDER, COUNT (S.GENDER) AS COUNT
FROM STUDENT S, SEMSEC SS, CLASS C
WHERES.USN = C.USN AND
SS.SSID = C.SSID
GROUP BY SS.SEM, SS.SEC, S.GENDER
ORDER BY SEM;
```

SEM	S	G	COUNT
3	A	M	1
3	B	F	1
3	C	M	1
4	A	F	1
4	A	M	1
4	B	M	1
4	C	M	1
7	A	F	1
7	A	M	2
8	A	F	1
8	A	M	1
8	B	F	1
8	C	F	1

3. **Create a view of Test1 marks of student USN ‘1BI15CS101’ in all subjects.**

```
CREATE VIEW STU_TEST1_MARKS_VIEW
AS
SELECT TEST1, SUBCODE
FROM IAMARKS
WHERE USN = '1RN13CS091';
```

TEST1	SUBCODE
15	10CS81
12	10CS82
19	10CS83
20	10CS84
15	10CS85

4. Calculate the FinalIA (average of best two test marks) and update the corresponding table for all students.

```
  CREATE OR REPLACE PROCEDURE AVGMARKS
  IS
  CURSOR C_IAMARKS IS
SELECT GREATEST(TEST1,TEST2) AS A, GREATEST(TEST1,TEST3) AS B,
GREATEST(TEST3,TEST2) AS C
FROM IAMARKS
WHERE FINALIA IS NULL
FOR UPDATE;

  C_A NUMBER;
  C_B NUMBER;
  C_C NUMBER;
  C_SM NUMBER;
  C_AV NUMBER;

BEGIN
  OPEN C_IAMARKS;
  LOOP
  FETCH C_IAMARKS INTO C_A, C_B, C_C;
    EXIT WHEN C_IAMARKS%NOTFOUND;
    --DBMS_OUTPUT.PUT_LINE(C_A || ' ' || C_B || ' ' || C_C);
    IF (C_A != C_B) THEN
 C_SM:=C_A+C_B;
    ELSE
 C_SM:=C_A+C_C;
    END IF;

    C_AV:=C_SM/2;
    --DBMS_OUTPUT.PUT_LINE('SUM = '||C_SM);
    --DBMS_OUTPUT.PUT_LINE('AVERAGE = '||C_AV);
    UPDATE IAMARKS SET FINALIA=C_AV  WHERE CURRENT OF C_IAMARKS;

  END LOOP;
  CLOSE C_IAMARKS;
END;
/
```

Note: Before execution of PL/SQL procedure, IAMARKS table contents are:

SELECT * FROM IAMARKS;

```
SQL> SELECT * FROM IAMARKS;

USN        SUBCODE  SSID       TEST1      TEST2      TEST3    FINALIA
---------- -------- ----- ---------- ---------- ---------- ----------
1RN13CS091 10CS81   CSE8C          15         16         18
1RN13CS091 10CS82   CSE8C          12         19         14
1RN13CS091 10CS83   CSE8C          19         15         20
1RN13CS091 10CS84   CSE8C          20         16         19
1RN13CS091 10CS85   CSE8C          15         15         12
```

Below SQL code is to invoke the PL/SQL stored procedure from the command line:

```
BEGIN
AVGMARKS;
END;
```

```
SQL> select * from IAMARks;

USN        SUBCODE  SSID       TEST1      TEST2      TEST3    FINALIA
---------- -------- ----- ---------- ---------- ---------- ----------
1RN13CS091 10CS81   CSE8C          15         16         18         17
1RN13CS091 10CS82   CSE8C          12         19         14         17
1RN13CS091 10CS83   CSE8C          19         15         20         20
1RN13CS091 10CS84   CSE8C          20         16         19         20
1RN13CS091 10CS85   CSE8C          15         15         12         15
```

5. Categorize students based on the following criterion:
If FinalIA = 17 to 20 then CAT = 'Outstanding'
If FinalIA = 12 to 16 then CAT = 'Average'
If FinalIA< 12 then CAT = 'Weak'
Give these details only for 8th semester A, B, and C section students.

```
SELECT S.USN,S.SNAME,S.ADDRESS,S.PHONE,S.GENDER,
    (CASE
      WHEN IA.FINALIA BETWEEN 17 AND 20 THEN 'OUTSTANDING'
      WHEN IA.FINALIA BETWEEN 12 AND 16 THEN 'AVERAGE'
      ELSE 'WEAK'
    END) AS CAT
FROM STUDENT S, SEMSEC SS, IAMARKS IA, SUBJECT SUB
WHERE S.USN = IA.USN AND
SS.SSID = IA.SSID AND
SUB.SUBCODE = IA.SUBCODE AND
SUB.SEM = 8;
```

USN	SNAME	ADDRESS	PHONE	G	CAT
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	OutStanding
1RN13CS091	TEESHA	BENGALURU	7712312312	F	Average