DBMS RECORD

Rahul Prakasha
1BM18CS078
DBMS LAB
19CS4PCDBM

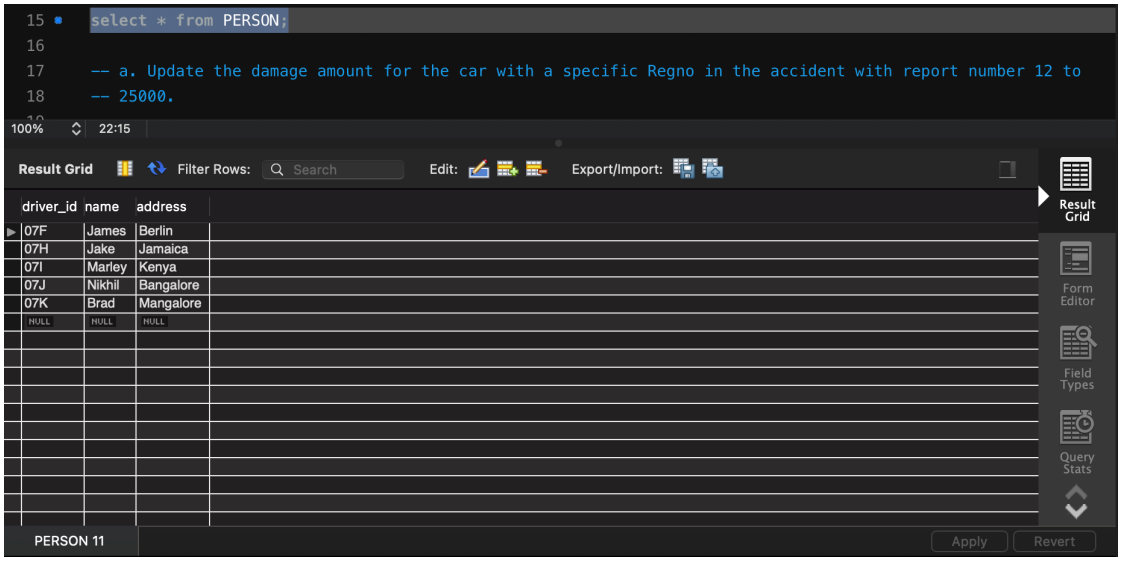
PROGRAM 1: INSURANCE DATABASE

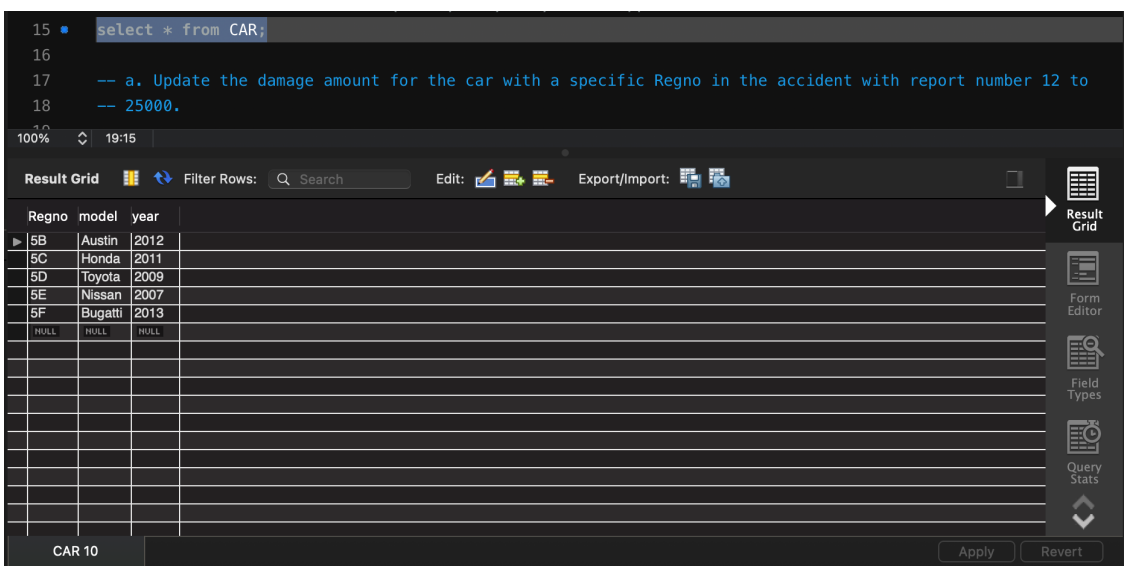
```
show databases;
create database Insurance01;
use Insurance01;
create table PERSON(driver_id varchar(30) primary key, name varchar(30), address
varchar(30));
create table CAR(Regno varchar(30) primary key, model varchar(30), year int);
create table ACCIDENT(report_number int primary key, adate date, location varchar(30));
create table OWNS(driver_id varchar(30), Regno varchar(30), primary key(driver_id,Regno),
foreign key(driver_id) references PERSON(driver_id), foreign key(Regno) references
CAR(Regno));
create table PARTICIPATED(driver_id varchar(30), Regno varchar(30), report_number int,
damage_amount int, primary key(driver_id, Regno), foreign key(driver_id, Regno) references
OWNS(driver_id, Regno));
show tables;
insert into PERSON values('07K','Brad','Mangalore');
insert into CAR values('5F','Bugatti','2013');
insert into ACCIDENT values('12','2007/1/21','Mangalore');
insert into OWNS values('07K','5F');
insert into PARTICIPATED values('07K','5F','12','555555');
select * from ACCIDENT;

-- a. Update the damage amount for the car with a specific Regno in the accident with report
number 12 to
-- 25000.

update PARTICIPATED set damage_amount='25000' where Regno='5F' AND
report_number='12';
select * from PARTICIPATED;


-- b.Add a new accident to the database.

insert into ACCIDENT values('14','1967/09/02','cuba');
select * from ACCIDENT;

-- c.Find the total number of people who owned cars that involved in accidents in 2008.

select count(*) from ACCIDENT where adate>'2007/12/31' AND adate<'2009/01/01';
```

-- d.Find the number of accidents in which cars belonging to a specific model were involved.

select count(ACCIDENT.report_number) from ACCIDENT,PARTICIPATED,CAR where ACCIDENT.report_number=PARTICIPATED.report_number AND PARTICIPATED.Regno=CAR.Regno AND CAR.model='Honda';

Output:





```
select * from ACCIDENT;

-- a. Update the damage amount for the car with a specific Regno in the accident with report number 12 to
-- 25000.
```

100% 24:15

Result Grid Filter Rows: Search Edit: Export/Import:

report_number	adate	location
8	1995-12-10	kenya
9	1998-10-01	Atlanta
10	2000-05-02	Jammu
11	2002-07-17	Singapore
12	2007-01-21	Mangalore
13	1967-09-02	Bermuda
NULL	NULL	NULL

ACCIDENT 9 Apply Revert

```
select * from OWNS;

-- a. Update the damage amount for the car with a specific Regno in the accident with report number 12 to
-- 25000.
```

100% 20:15

Result Grid Filter Rows: Search Edit: Export/Import:

driver_id	Regno
07F	5B
07H	5C
07I	5D
07J	5E
07K	5F
NULL	NULL

OWNS 8 Apply Revert

Result Grid

driver_id	Regno	report_number	damage_amount
07F	5B	8	999999
07H	5C	9	888888
07I	5D	10	777777
07J	5E	11	666666
07K	5F	12	555555
NULL	NULL	NULL	NULL

PARTICIPATED 34 Apply

Queries

```
select * from PARTICIPATED;
```

100% 28:15

Result Grid Filter Rows: Search Edit: Ex

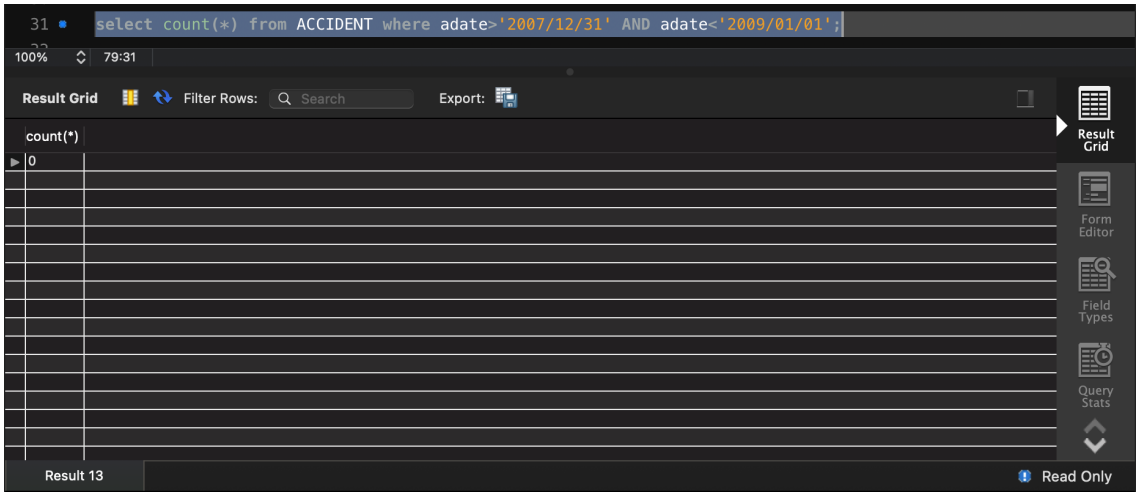
driver_id	Regno	report_numb...	damage_amount
07F	5B	8	999999
07H	5C	9	888888
07I	5D	10	777777
07J	5E	11	666666
07K	5F	12	25000
NULL	NULL	NULL	NULL

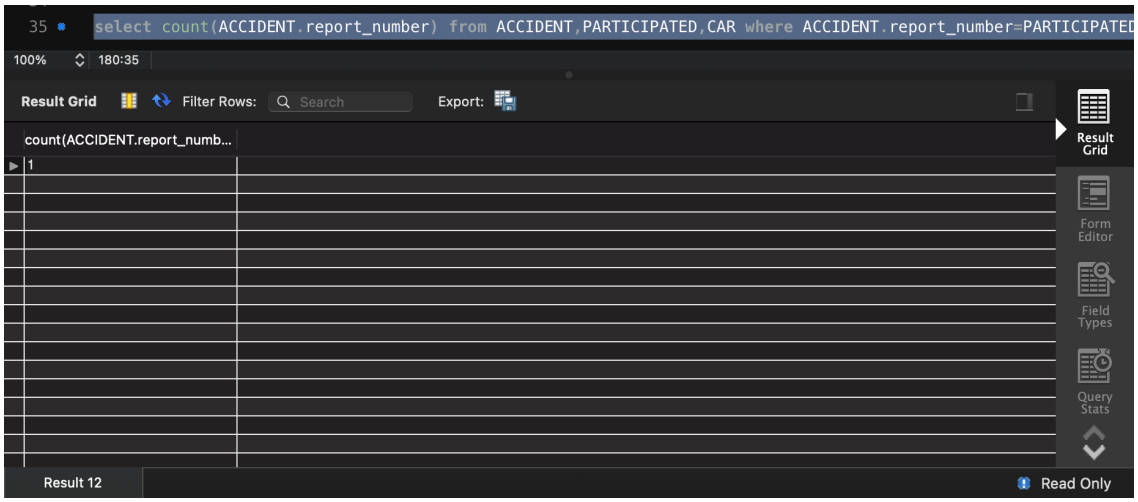
```
-- Add a new accident to the database.

insert into ACCIDENT values('14','1967/09/02','cuba');
select * from ACCIDENT;

-- Find the total number of people who owned cars that involv
```

100% 18:35

Result Grid Filter Rows: Search Edit: Export/Import:

report_numb...	adate	location
8	1995-12-10	kenya
9	1998-10-01	Atlanta
10	2000-05-02	Jammu
11	2002-07-17	Singapore
12	2007-01-21	Mangalore
13	1967-09-02	Bermuda
14	1967-09-02	cuba
NULL	NULL	NULL





PROGRAM 2. BANKING ENTERPRISE DATABASE

show databases;
create database Banking;
use Banking;
create table BRANCH(Branch_Name varchar(25) primary key, Branch_City varchar(25), Assets real);
create table BANKACCOUNTS(Acc_no int, Branch_Name varchar(25), Balance real, primary key(Acc_no), foreign key(Branch_Name) references BRANCH(Branch_Name) on delete cascade);
create table BANKCUSTOMER(CustomerName varchar(30), CustomerStreet varchar(30), CustomerCity varchar(30), primary key(CustomerName));
create table DEPOSITER(CustomerName varchar(30), Acc_no Int, primary key(CustomerName, Acc_no), foreign key(CustomerName) references

```
BANKCUSTOMER(CustomerName) on delete cascade, foreign key(Acc_no) references
BANKACCOUNTS(Acc_no) on delete cascade);
create table LOAN(LoanNumber int, Branch_Name varchar(30), Amount real, primary
key(loanNumber), foreign key(Branch_Name) references BRANCH(Branch_Name) on delete
cascade);
show tables;
insert into BRANCH values('SBI_RAJ','Mangalore','700000');
insert into BANKACCOUNTS values('0004','SBI_MG','11000');
insert into BANKCUSTOMER values('Amit','RT_Nagar','Bombay');
insert into DEPOSITER values('Amit','0005');
insert into LOAN values('05','SBI_IND','900000');
select * from BANKACCOUNTS;

-- Find all the customers who have at least two deposits at the same branch (Ex. 'SBI_RAJ')

select C.CustomerName
 from BANKCUSTOMER C
 where exists
 (select D.CustomerName, count(D.CustomerName)
 from DEPOSITER D, BANKACCOUNTS BA
 where
 D.Acc_no = BA.Acc_no AND
 C.CustomerName = D.CustomerName AND
 BA.Branch_Name = 'SBI_RAJ'
 group by D.CustomerName
 having count(D.CustomerName)>=2);

 -- Find all the customers who have an account at all the branches located in a specific city (Ex.
Delhi).

select BC.CustomerName
 from BANKCUSTOMER BC
 where not exists
 (select Branch_Name from BRANCH
 where Branch_City = 'Delhi'
 -
 (select BA.Branch_Name from
 DEPOSITER D, BANKACCOUNTS BA
 where D.Acc_no = BA.Acc_no and
 BC.CustomerName = D.CustomerName));

-- Demonstrate how you delete all account tuples at every branch located in a specific city (Ex.
Bomay).
```

```
delete from BANKACCOUNTS
 where Branch_Name IN (
 select Branch_Name
 from BRANCH
 where Branch_City='Bombay');
```

Output:

```
15  select * from BRANCH;
16
```

100% 22:15

Result Grid Filter Rows: Search

Branch_Name	Branch_City	Assets
SBI_CHM	Bangalore	100000
SBI_IND	Delhi	900000
SBI_KOR	Mysore	2000000
SBI_MG	Bombay	500000
SBI_RAJ	Mangalore	700000
NULL	NULL	NULL

```
15  select * from BANKACCOUNTS;
16
```

100% 28:15

Result Grid Filter Rows: Search Edit:

Acc_no	Branch_Name	Balance
1	SBI_CHM	7000
2	SBI_RAJ	4000
3	SBI_KOR	2000
4	SBI_MG	11000
5	SBI_IND	1000
NULL	NULL	NULL

```
select * from BANKCUSTOMER;
```

100% 28:15

Result Grid Filter Rows: Search Edit:

CustomerName	CustomerStreet	CustomerCity
Amit	RT_Nagar	Bombay
Arjun	Rajajinagar	Bangalore
Corona	JP_Nagar	Mangalore
James	MG_Road	Mysore
Sam	Indranagar	Delhi
NULL	NULL	NULL

```
select * from DEPOSITER;
```

100% 25:15

Result Grid Filter Rows: Search Edit:

CustomerName	Acc_no
James	1
Corona	2
Arjun	3
James	4
Amit	5
NULL	NULL

```
select * from LOAN;
```

100% 20:15

Result Grid Filter Rows: Search Edit:

LoanNumber	Branch_Name	Amount
1	SBI_CHM	120000
2	SBI_RAJ	120000
3	SBI_KOR	150000
4	SBI_MG	70000
5	SBI_IND	900000
NULL	NULL	NULL

Queries

```
select C.CustomerName
 from BANKCUSTOMER C
 where exists
 (select D.CustomerName, count(D.CustomerName)
 from DEPOSITER D, BANKACCOUNTS BA
 where
 D.Acc_no = BA.Acc_no AND
 C.CustomerName = D.CustomerName AND
 BA.Branch_Name = 'SBI_RAJ'
 group by D.CustomerName
 having count(D.CustomerName)>=2);

 -- Find all the customers who have an account at
```

100% 35:29

Result Grid Filter Rows: Search Edit:

CustomerName
NULL

```
select BC.CustomerName
 from BANKCUSTOMER BC
 where not exists
 (select Branch_Name from BRANCH
 where Branch_City = 'Delhi'
 -
 (select BA.Branch_Name from
 DEPOSITER D, BANKACCOUNTS BA
 where D.Acc_no = BA.Acc_no and
 BC.CustomerName = D.CustomerName));
```

100% 37:42

Result Grid Filter Rows: Search Edit: Export/Import

CustomerName
Sam
NULL

```
delete from BANKACCOUNTS
where Branch_Name IN (
select Branch_Name
from BRANCH
where Branch_City='Bombay');
```

100% 28:15

Result Grid Filter Rows: Search Edit: Export/Impo

Acc_no	Branch_Name	Balance
1	SBI_CHM	7000
2	SBI_RAJ	4000
3	SBI_KOR	2000
5	SBI_IND	1000
NULL	NULL	NULL

PROGRAM 3. SUPPLIER DATABASE

```
create database Supplier;
use Supplier;
create table SUPPLIERS(sid int(5) primary key, sname varchar(20), city varchar(20));
create table PARTS(pid int(5) primary key, pname varchar(20), color varchar(10));
create table CATALOG(sid int(5), pid int(5),  foreign key(sid) references SUPPLIERS(sid),
foreign key(pid) references PARTS(pid), cost float(6), primary key(sid, pid));
show tables;
insert into SUPPLIERS values('105','BELURY','Delhi');
insert into PARTS values('205','Charger','Black');
insert into CATALOG values('104','203','40');
select * from PARTS;

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

SELECT DISTINCT P.pname
   FROM PARTS P, CATALOG C
   WHERE P.pid = C.pid;

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

SELECT S.sname
 FROM SUPPLIERS S
 WHERE NOT EXISTS((SELECT P.pid FROM PARTS P)
```

```
 EXCEPT
 (SELECT C.pid FROM CATALOG C
 WHERE C.sid = S.sid));
```

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

```
SELECT S.sname
 FROM SUPPLIERS S
 WHERE NOT EXISTS ((SELECT P.pid
 FROM PARTS P
 WHERE P.color = 'Red')
 EXCEPT
(SELECT C.pid
 FROM CATALOG C, PARTS P
 WHERE C.sid = S.sid AND
 C.pid = P.pid AND P.color = 'Red'));
```

-- iv. Find the pnames of parts supplied by ABIBAS Suppliers and by no one else

```
SELECT P.pname
FROM PARTS P, CATALOG C, SUPPLIERS S
WHERE P.pid = C.pid AND C.sid = S.sid
AND S.sname = 'ABIBAS'
AND NOT EXISTS ( SELECT *
                 FROM CATALOG C1, SUPPLIERS S1
                 WHERE P.pid = C1.pid AND C1.sid = S1.sid AND
                 S1.sname <> 'ABIBAS' );
```

-- 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).

```
SELECT DISTINCT C.sid
FROM CATALOG C
WHERE C.cost > ( SELECT AVG (C1.cost)
                 FROM CATALOG C1
                 WHERE C1.pid = C.pid );
```

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

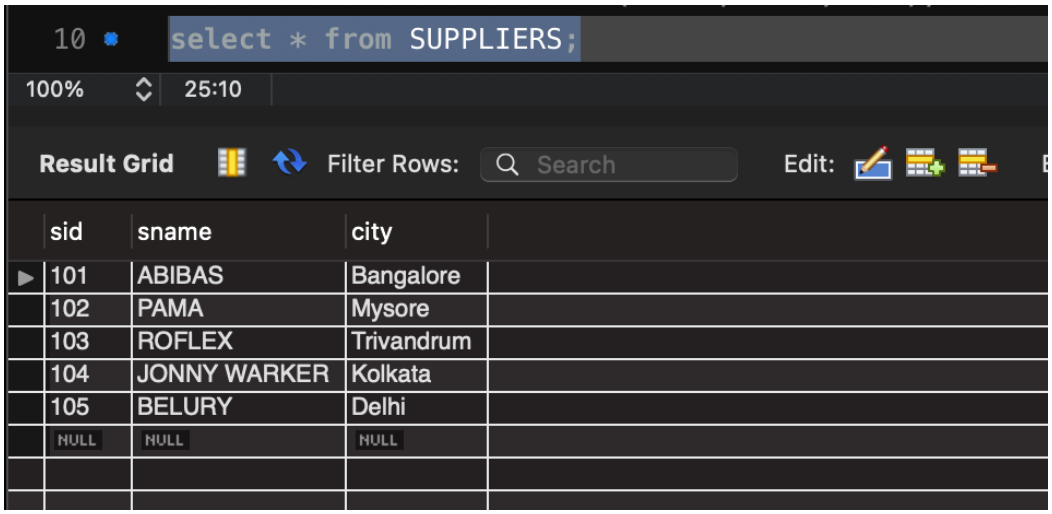
```
SELECT P.pid, S.sname
FROM PARTS P, SUPPLIERS S, CATALOG C
WHERE C.pid = P.pid
AND C.sid = S.sid
AND C.cost = (SELECT MAX(C1.cost)
```

```
                    FROM CATALOG C1
                    WHERE C1.pid = P.pid);
```

-- vii. Find the sids of suppliers who supply only red parts.

```
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);
```

Output:

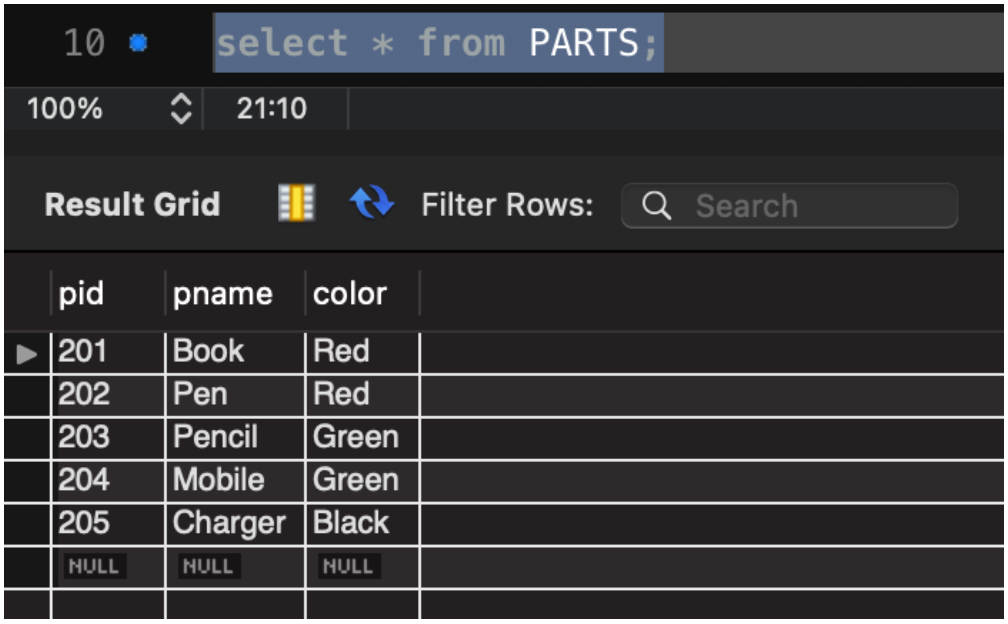


```
10  select * from SUPPLIERS;
```

100% 25:10

Result Grid Filter Rows: Search Edit:

sid	sname	city
101	ABIBAS	Bangalore
102	PAMA	Mysore
103	ROFLEX	Trivandrum
104	JONNY WARKER	Kolkata
105	BELURY	Delhi
NULL	NULL	NULL



```
10  select * from PARTS;
```

100% 21:10

Result Grid Filter Rows: Search

pid	pname	color
201	Book	Red
202	Pen	Red
203	Pencil	Green
204	Mobile	Green
205	Charger	Black
NULL	NULL	NULL

```
select * from CATALOG;
```

100% 23:10

Result Grid Filter Rows: Search Edit:

sid	pid	cost
101	201	10
101	202	10
101	203	30
101	204	10
101	205	10
102	201	10
102	202	20
103	203	30
104	203	40
NULL	NULL	NULL

Queries

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

SELECT DISTINCT P.pname
    FROM PARTS P, CATALOG C
    WHERE P.pid = C.pid;
```

100% 25:16

Result Grid Filter Rows: Search Export:

pname
Book
Pen
Pencil
Mobile
Charger

```sql
-- iv. Find the pnames of parts supplied by ABIBAS Suppliers and by no one else

SELECT P.pname
FROM PARTS P, CATALOG C, SUPPLIERS S
WHERE P.pid = C.pid AND C.sid = S.sid
AND S.sname = 'ABIBAS'
AND NOT EXISTS ( SELECT *
        FROM CATALOG C1, SUPPLIERS S1
        WHERE P.pid = C1.pid AND C1.sid = S1.sid AND
        S1.sname <> 'ABIBAS' );
```

100% 26:50

Result Grid Filter Rows: Search Export:

pname
Mobile
Charger

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

SELECT DISTINCT C.sid
FROM CATALOG C
WHERE C.cost > ( SELECT AVG (C1.cost)
                FROM CATALOG C1
                WHERE C1.pid = C.pid );

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

SELECT P.pid, S.sname
```

100% 28:58

Result Grid Filter Rows: Search Export:

sid
102
104

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

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);
```

100% 25:68

Result Grid Filter Rows: Search Export:

pid	sname
201	ABIBAS
201	PAMA
202	PAMA
203	JONNY WARKER
204	ABIBAS
205	ABIBAS

```
-- vii. Find the sids of suppliers who supply only red parts.

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);
```

100% 25:78

Result Grid Filter Rows: Search Export:

pid	sname
201	ABIBAS
201	PAMA
202	PAMA
203	JONNY WARKER
204	ABIBAS
205	ABIBAS

PROGRAM 4. STUDENT FACULTY DATABASE

```
create database Student_Faculty;
use Student_Faculty;
CREATE TABLE STUDENT(snum INT, sname VARCHAR(10), major VARCHAR(2), lvl
VARCHAR(2), age INT, primary key(snum));
CREATE TABLE FACULTY(fid INT,fname VARCHAR(20), deptid INT, PRIMARY KEY(fid));
CREATE TABLE CLASS(cname VARCHAR(20), metts_at TIMESTAMP, room VARCHAR(10),
fid INT, PRIMARY KEY(cname), FOREIGN KEY(fid) REFERENCES faculty(fid));
CREATE TABLE ENROLLED(snum INT, cname VARCHAR(20), PRIMARY KEY(snum,cname),
FOREIGN KEY(snum) REFERENCES student(snum), FOREIGN KEY(cname) REFERENCES
class(cname));
SHOW TABLES;
INSERT INTO STUDENT VALUES('6','Corona','CS','Sr','19');
INSERT INTO FACULTY VALUES('15','Jhonny Bravo','1000');
INSERT INTO CLASS VALUES('class7','12/11/15 10:10:10.000000','R3','14');
INSERT INTO ENROLLED VALUES('5','class5');
SELECT * FROM CLASS;

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

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 = 'Jagdesh' AND S.lvl = 'Jr';

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

```
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);
```

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

```
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.metts_at = C2.metts_at);
```

-- iv. 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.room FROM Class C)
-
(SELECT C1.room
FROM Class C1
WHERE C1.fid = F.fid );
```

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

```
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);
```

-- vi. 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);
```

-- 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).

```
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));
```

Output:

```
12    SELECT * FROM STUDENT;
100%    23:12    1 error found
Result Grid    Filter Rows:    Search
```

snum	sname	major	lvl	age
1	John	CS	Sr	20
2	Bond	CS	Jr	20
3	Rita	IS	Sr	19
4	Rambo	CS	Jr	21
5	Jack	CS	Jr	20
6	Corona	CS	Sr	19
NULL	NULL	NULL	NULL	NULL

```
12  SELECT * FROM FACULTY;
```

100% 23:12 1 error found

Result Grid Filter Rows: Search

fid	fname	deptid
11	Jagdesh	1000
12	Kumar	1000
13	Salmon Khan	1001
14	Mira	1002
15	Jhonny Bravo	1000
NULL	NULL	NULL

```
12  SELECT * FROM CLASS;
```

100% 21:12 1 error found

Result Grid Filter Rows: Search

cname	metts_at	room	fid
class1	2012-11-15 10:15:16	R12	14
class10	2012-11-15 10:15:16	R128	14
class2	2012-11-15 10:15:20	R2	12
class3	2012-11-15 10:15:25	R3	11
class4	2012-11-15 20:15:20	R4	14
class5	2012-11-15 20:15:20	R3	15
class6	2012-11-15 13:20:20	R3	14
class7	2012-11-15 10:10:10	R3	14
NULL	NULL	NULL	NULL

```
SELECT * FROM ENROLLED;
```

100% 24:12 1 error found

Result Grid Filter Rows: Search

snum	cname
1	class1
2	class1
3	class3
4	class3
5	class4
1	class5
2	class5
3	class5
4	class5
5	class5
NULL	NULL

Queries

```
-- i. Find the names of all Juniors (level(lvl) = Jr) who are enrolled in a class taught by Jagdesh.

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 = 'Jagdesh' AND S.lvl = 'Jr';
```

100% 38:19 1 error found

Result Grid Filter Rows: Search Export:

Sname
Rambo

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

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);
```

100% 25:29 1 error found

Result Grid Filter Rows: Search Edit: Export/Import:

cname
class10
class5
NULL

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

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.metts_at = C2.metts_at);
```

100% 1:40 1 error found

Result Grid Filter Rows: Search Export:

sname
Jack

```sql
-- v. Find the names of faculty members for whom the combined enrollment of the courses that they teach is less

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);
```

100% 20:58

Result Grid Filter Rows: Search Export:

fname
Jagdesh
Kumar
Salmon Khan
Mira

```sql
-- vi. 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);

-- vii. For each age value that appears in Students, find the level value that ar
```

100% 1:66

Result Grid Filter Rows: Search Export:

sname

```sql
-- vii. For each age value that appears in Students, find the level value that appears most often. For example,

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));
```

100% 1:79

Result Grid Filter Rows: Search Export:

age	lvl
20	Jr
19	Sr
21	Jr

PROGRAM 5. AIRLINE FLIGHT DATABASE

```
create database flight_airlines;
use flight_airlines;
create table flights(
   flno integer not null,
   ffrom varchar(20) not null,
   fto varchar(20) not null,
   distance int not null,
   departs time not null,
   arrives time not null,
   price int not null,
   primary key(flno)
);
create table aircraft(
   aid int not null,
   aname varchar(20) not null,
   cruisingrange int not null,
   primary key(aid)
);
create table employee(
   eid int not null,
   ename varchar(20) not null,
   salary int not null,
   primary key(eid)
);
create table certified(
   eid int not null,
   aid int not null,
   foreign key(eid) REFERENCES employee(eid) on delete cascade on update cascade,
foreign key(aid) references aircraft(aid) on delete cascade on update cascade);
show tables;

 INSERT INTO flights VALUES('7','Bangalore','Frankfurt','17000','12:00:00','06:30:00','99000');
 INSERT INTO aircraft (aid,aname,cruisingrange) values('951','Aircraft03','1000');
 INSERT INTO employee (eid,ename,salary) VALUES('7','Ram','100000');
 INSERT INTO certified (eid,aid) VALUES('1','789');
 select * from certified;

-- i. Find the names of aircraft such that all pilots certified to operate them have salaries more
than Rs.80,000.
```

```
 select distinct aname from aircraft where aid in (select aid from certified where eid in (select eid from employee
 where salary > 80000));

-- ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.

select c.eid, max(cruisingrange) from certified c, aircraft a where c.aid = a.aid group by c.eid having count(*) > 3;

-- iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt

select e.ename from employee e where exists (select * from certified c where c.eid = e.eid) and e.salary
< (select min(price) from flights where ffrom = 'Bangalore' and fto = 'Frankfurt');

-- iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of
--  all pilots certified for this aircraft.

select a.aname, avg(e.salary) from aircraft a, certified c, employee e where c.aid = a.aid and c.eid = e.eid
and a.cruisingrange > 1000 group by a.aname;

-- v. Find the names of pilots certified for some Boeing aircraft.

select e.ename from employee e, certified c, aircraft a where a.aname like '%Boeing%' and a.aid = c.aid
and c.eid = e.eid

-- vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

select aid from aircraft where cruisingrange > (select distance from flights where ffrom = 'Bangalore'
and fto = 'Delhi');

-- vii.Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.

select e1.ename, e1.salary from employee e1 where e1.salary > (select avg(e.salary) from employee e where e.eid in
(select eid from certified)) and not exists(select * from certified c where c.eid = e1.eid)
```

Output:

```
36  select * from flights;
37
38
```

100% 24:36

Result Grid Filter Rows: Search Edit:

flno	ffrom	fto	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
NULL	NULL	NULL	NULL	NULL	NULL	NULL

```
36  select * from aircraft;
37
```

100% 25:36

Result Grid Filter Rows: Search

aid	aname	cruisingrange
123	Airbus	1000
302	Boeing	5000
306	Jet01	5000
378	Airbus380	8000
456	Aircraft	500
789	Aircraft02	800
951	Aircraft03	1000
NULL	NULL	NULL

```
36  select * from certified;
37
```

100% 2:36

Result Grid Filter Rows: Search

eid	aid
1	123
1	123
2	123
1	302
5	302
7	302
1	306
2	306
1	378
2	378

```
select * from employee;
```

100% 25:36

Result Grid Filter Rows: Search

eid	ename	salary
1	Ajay	30000
2	Ajith	85000
3	Arnab	50000
4	Harry	45000
5	Ron	90000
6	Josh	75000
7	Ram	100000
NULL	NULL	NULL

Queries

```
-- i. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

select distinct aname from aircraft where aid in (select aid from certified where eid in (select eid from employee
where salary > 80000));
```

100% 25:41

Result Grid Filter Rows: Search Export:

aname
Airbus
Boeing
Jet01
Airbus380
Aircraft02

```
-- ii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range

select c.eid, max(cruisingrange) from certified c, aircraft a where c.aid = a.aid group by c.eid having count(*) >
```

100% 118:45

Result Grid Filter Rows: Search Export:

eid	max(cruisingrang...
1	8000

```
-- iii. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt

select e.ename from employee e where exists (select * from certified c where c.eid = e.eid) and e.salary
< (select min(price) from flights where ffrom = 'Bangalore' and fto = 'Frankfurt');
```

100% 1:50

Result Grid Filter Rows: Search Export:

eid	max(cruisingrang...
1	8000

```
-- iv. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of
--  all pilots certified for this aircraft.

select a.aname, avg(e.salary) from aircraft a, certified c, employee e where c.aid = a.aid and c.eid = e.eid
and a.cruisingrange > 1000 group by a.aname;
```

100% 45:56

Result Grid Filter Rows: Search Export:

aname	avg(e.salary)
Boeing	73333.3333
Jet01	57500.0000
Airbus380	53333.3333

```
-- v. Find the names of pilots certified for some Boeing aircraft.

select e.ename from employee e, certified c, aircraft a where a.aname like '%Boeing%' and a.aid = c.aid
and c.eid = e.eid
```

100% 18:61

Result Grid Filter Rows: Search Export:

ename
Ajay
Ron
Ram

```
-- vi. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

select aid from aircraft where cruisingrange > (select distance from flights where ffrom = 'Bangalore'
and fto = 'Delhi');
```

100% 1:66 1 error found

Result Grid Filter Rows: Search Edit: Export/Import:

aid
378
NULL

```
-- vii.Print the name and salary of every non-pilot whose salary is more than the average salary for pilots.

select e1.ename, e1.salary from employee e1 where e1.salary > (select avg(e.salary) from employee e where e.eid in
(select eid from certified)) and not exists(select * from certified c where c.eid = e1.eid)
```

100% 92:71 1 error found

Result Grid Filter Rows: Search Export:

ename	salary