VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems

(23CS3PCDBM)

Submitted by

NAGARAJ G M (1BM24CS176)

in partial fulfillment for the award of the degree of
BACHELOR OF ENGINEERING
in
COMPUTER SCIENCE AND ENGINEERING

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019
Sep-2024 to Jan-2025

B.M.S. College of Engineering,
Bull Temple Road, Bangalore 560019
(Affiliated To Visvesvaraya Technological University, Belgaum)
Department of Computer Science and Engineering

CERTIFICATE

This is to certify that the Lab work entitled "Database Management Systems (23CS3PCDBM)" carried out by **NAGARAJ G M (1BM24CS176),** who is bonafide student of **B.M.S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Divyashree S Assistant Professor Department of CSE, BMSCE	DR. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Insurance Database

Question

(Week 1)

Consider the Insurance database given below.

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. Display Accident date and location

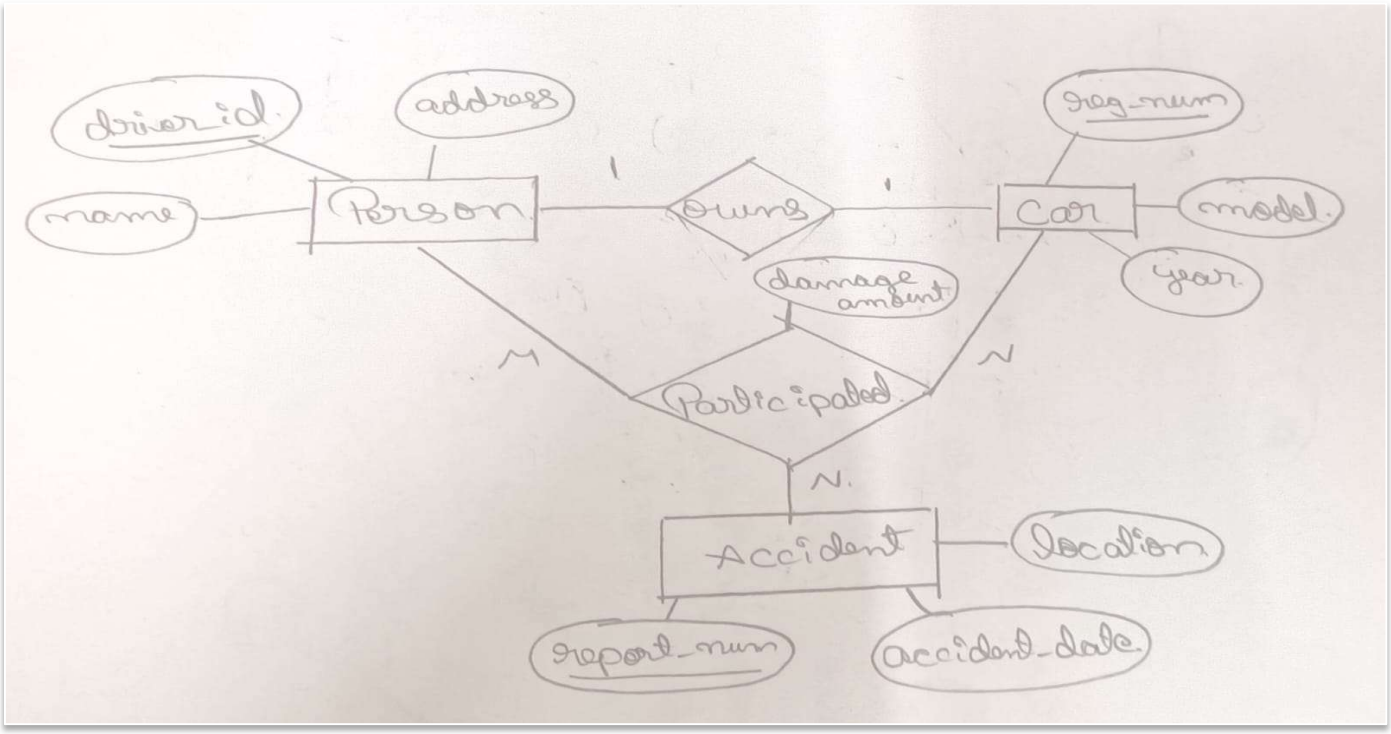
iv. Update the damage amount to 25000 for the car with a specific reg_num (example 'K A053408') for which the accident report number was 12.

v. Add a new accident to the database.

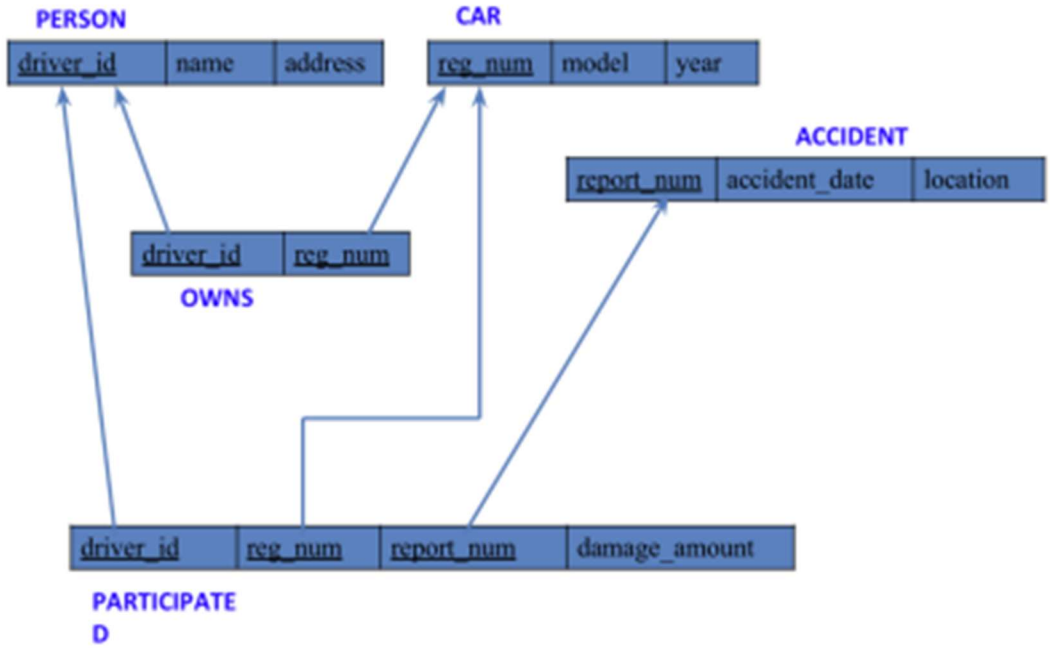
vi. Display Accident date and location

vii. Display driver id who did accident with damage amount greater than or equal to Rs.25000

Entity-Relationship Diagram:



Schema Diagram



Create database

```
create database insurance;
use insurance;
```

Create table

```
create table person(
driverid varchar(10) primary key,
name varchar(50),
address varchar(100)
);
create table car(
regnum varchar(20) primary key,
model varchar(40),
year int
);
create table accident(
reportnum int primary key,
accidentdate varchar(10),
location varchar(40)
);
create table owns(
driver_id varchar(10),
reg_num varchar(40),
foreign key (driver_id) references person(driverid),
foreign key (reg_num) references car(regnum)
);
```

```
create table participated (
driver_id varchar(10),
reg_num varchar(15),
report_num int,
damage_amount int,
foreign key (driver_id) references person(driver_id),
foreign key (reg_num) references car(reg_num),
foreign key (report_num) references accident(report_num)
);
```

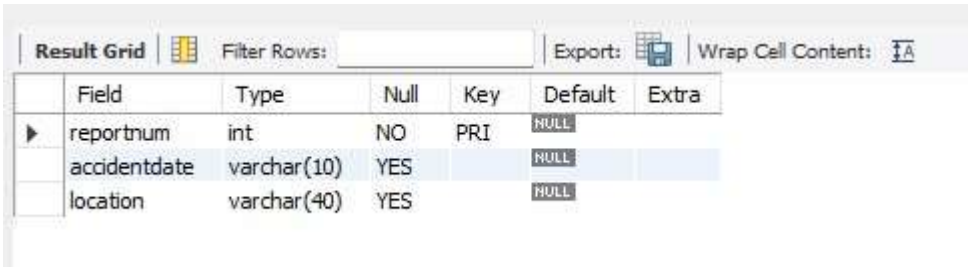
Structure of the table

desc person;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
driverid	varchar(10)	NO	PRI	NULL	
name	varchar(50)	YES		NULL	
address	varchar(100)	YES		NULL	

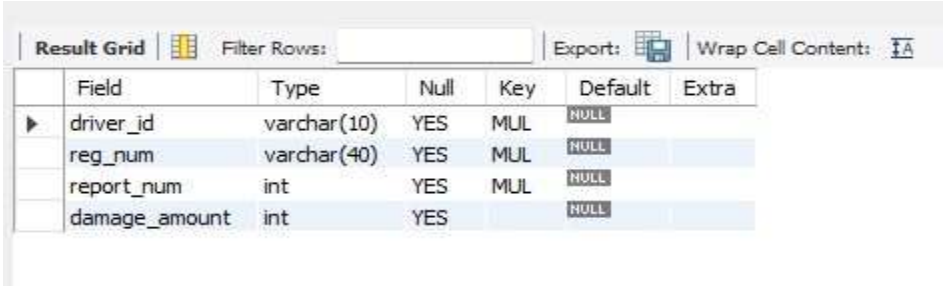
desc accident;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
reportnum	int	NO	PRI	NULL	
accidentdate	varchar(10)	YES		NULL	
location	varchar(40)	YES		NULL	

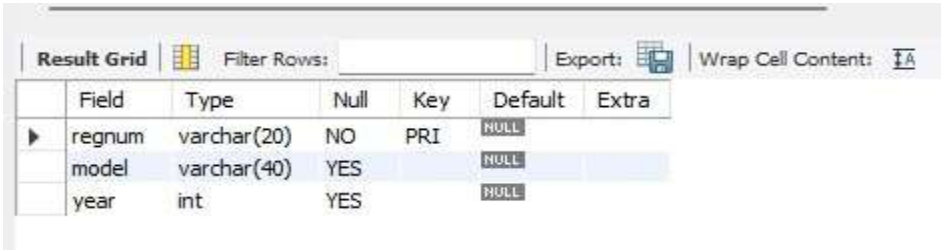
desc participated;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
driver_id	varchar(10)	YES	MUL	NULL	
reg_num	varchar(40)	YES	MUL	NULL	
report_num	int	YES	MUL	NULL	
damage_amount	int	YES		NULL	

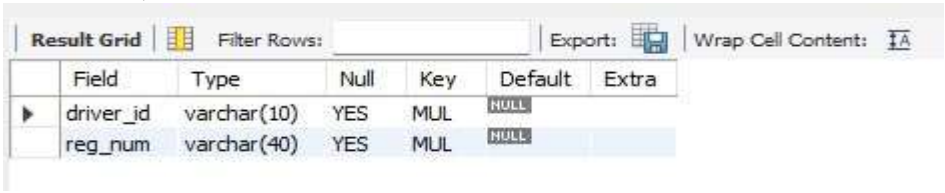
desc car;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
regnum	varchar(20)	NO	PRI	NULL	
model	varchar(40)	YES		NULL	
year	int	YES		NULL	

desc owns;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
driver_id	varchar(10)	YES	MUL	NULL	
reg_num	varchar(40)	YES	MUL	NULL	

Inserting Values to the table

insert into person

values

("A01", "Richard", "Srinvasnagar"),

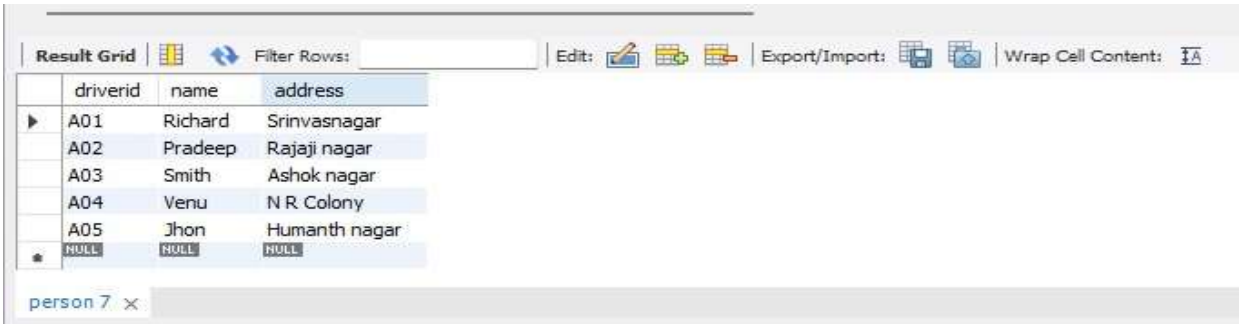
("A02", "Pradeep", "Rajaji nagar"),

("A03", "Smith", "Ashok nagar"),

("A04", "Venu", "N R Colony"),

("A05", "Jhon", "Humanth nagar");

select * from person;



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

driverid	name	address
A01	Richard	Srinvasnagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R Colony
A05	Jhon	Humanth nagar
NULL	NULL	NULL

person 7 ×

insert into car

values

("KA052250", "Indica", 1990),

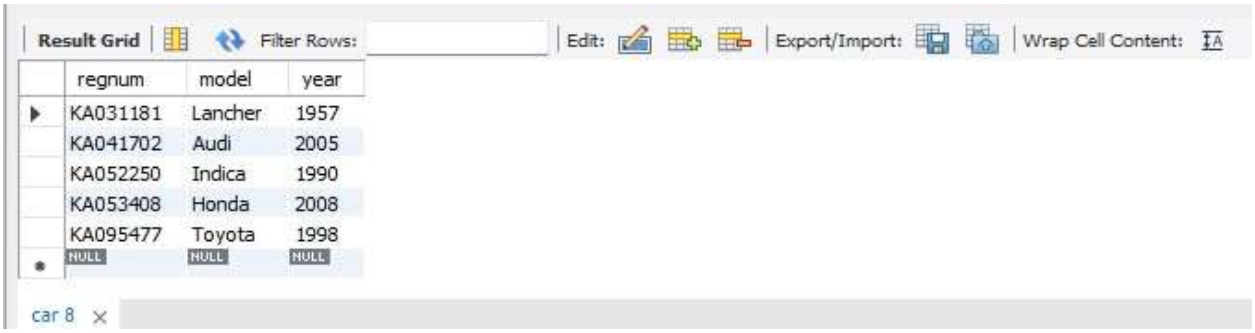
("KA031181", "Lancher", 1957),

("KA095477", "Toyota", 1998),

("KA053408", "Honda", 2008),

("KA041702", "Audi", 2005);

select * from car;



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

regnum	model	year
KA031181	Lancher	1957
KA041702	Audi	2005
KA052250	Indica	1990
KA053408	Honda	2008
KA095477	Toyota	1998
NULL	NULL	NULL

car 8

insert into owns

values

("A01", "KA052250"),

("A02", "KA053408"),

("A03", "KA031181"),

("A04", "KA095477"),

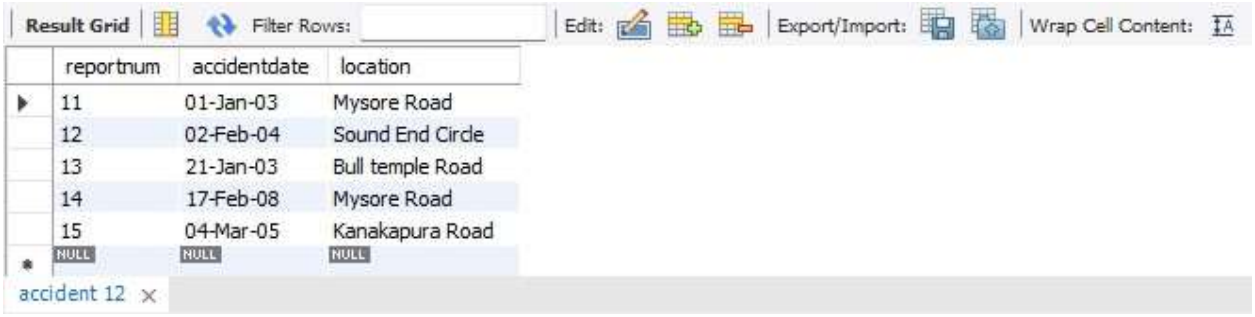
("A05", "KA041702");

select * from owns;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

```
insert into accident
 values
(11, "01-Jan-03", "Mysore Road"),
(12, "02-Feb-04", "Sound End Circle"),
(13, "21-Jan-03", "Bull temple Road"),
(14, "17-Feb-08", "Mysore Road"),
(15, "04-Mar-05", "Kanakapura Road");
select * from accident;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

reportnum	accidentdate	location
11	01-Jan-03	Mysore Road
12	02-Feb-04	Sound End Circle
13	21-Jan-03	Bull temple Road
14	17-Feb-08	Mysore Road
15	04-Mar-05	Kanakapura Road
NULL	NULL	NULL

accident 12 ×

```
insert into participated
values
("A01", "KA052250", 11, 10000),
("A02", "KA053408", 12, 50000),
("A03", "KA031181", 13, 25000),
("A04", "KA095477", 14, 3000),
("A05", "KA041702", 15, 5000);
select * from participated;
```

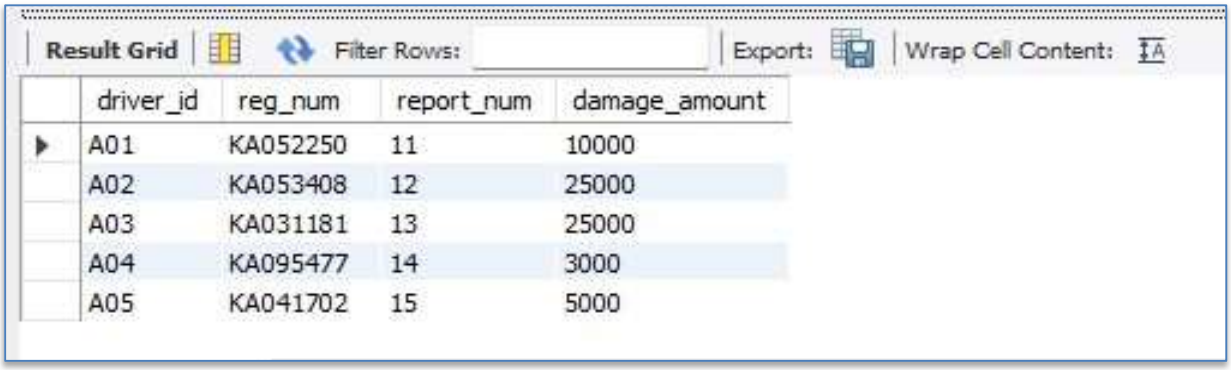
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

10000

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

```
update participated
set damage_amount = 25000
where reg_num = "KA053408" and report_num = 12;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

Display Accident date and location

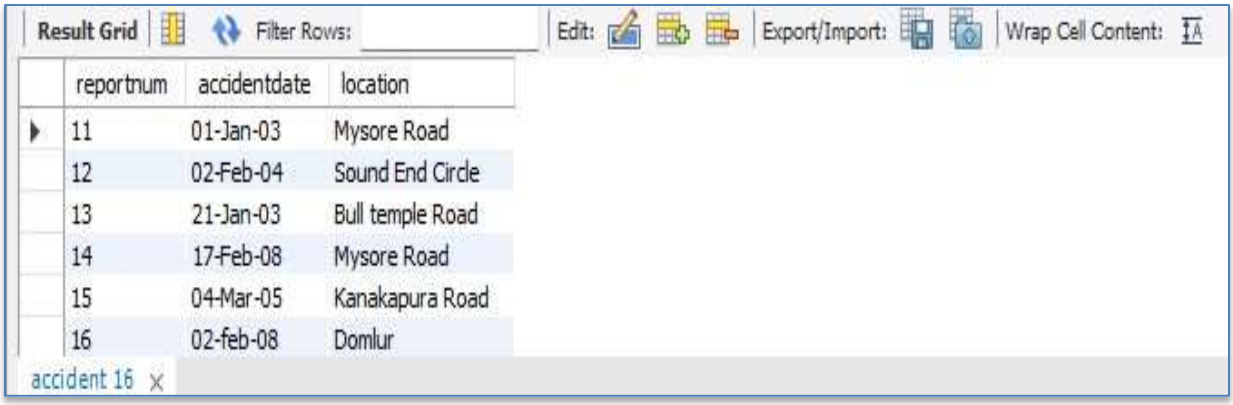
```
select accidentdate, location
from accident;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

accidentdate	location
01-Jan-03	Mysore Road
02-Feb-04	Sound End Circle
21-Jan-03	Bull temple Road
17-Feb-08	Mysore Road
04-Mar-05	Kanakapura Road

Add a new accident to the database.

```
insert into accident (reportnum, accidentdate, location)
values (16,"02-feb-08","Domlur");
select * from accident;
```



Display driver id who did accident with damage amount greater than or equal to Rs.25000

```
select driver_id
from participated
where damage_amount >= 25000;
```

More Queries on Insurance Database

Question

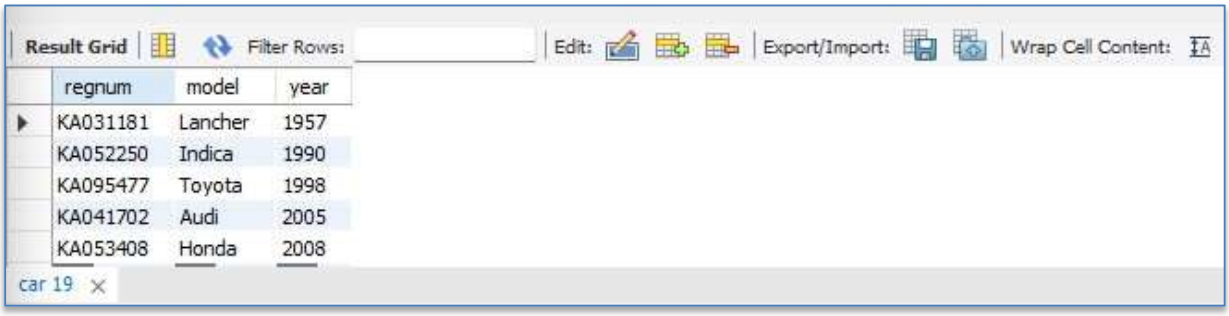
(Week 2)

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)
Create the above tables by properly specifying the primary keys and the foreign keys as done in “Program-1”week’s lab and Enter at least five tuples for each relation.

i. Display the entire CAR relation in the ascending order of manufacturing year.
ii. Find the number of accidents in which cars belonging to a specific model (example 'Lancher') were involved.
iii. Find the total number of people who owned cars that involved in accidents in 2008.
iv. List the entire participated relation in the Descending Order of Damage Amount.
v. Find the Average Damage Amount.
vi. Delete the tuple whose Damage Amount is below the Average Damage Amount
vii. List the name of drivers whose Damage is Greater than the Average Damage Amount.
viii. Find Maximum Damage Amount.

Display the entire CAR relation in the ascending order of manufacturing year.

```
select *
from car
order by year asc;
```



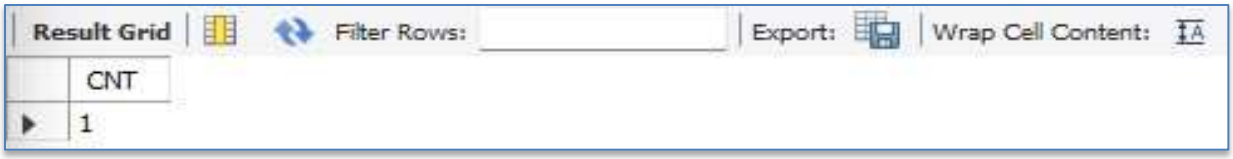
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

car 19

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

```
select count(report_num) CNT
from car c,participated p
where c.regnum=p.reg_num and model='Lancher';
```

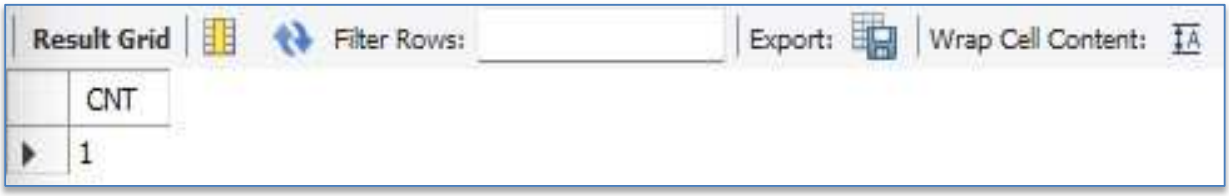


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CNT
1

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

```
select count(distinct driver_id) CNT
from participated a, accident b
where a.report_num = b.reportnum and b.accidentdate like '%08';
```

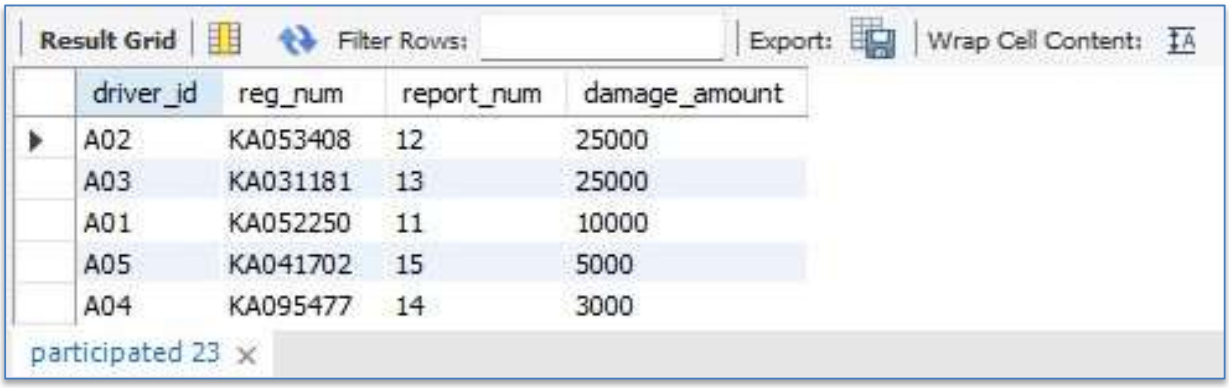


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

CNT
1

List the entire participated relation in the Descending Order of Damage Amount.

```
select *
from participated
order by damage_amount desc;
```



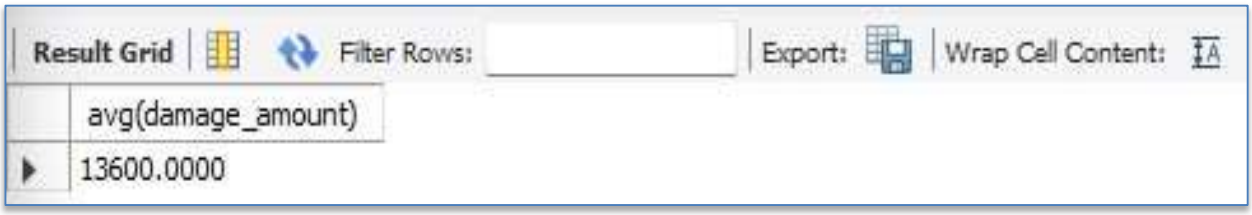
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

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

participated 23

Find the Average Damage Amount.

```
select avg(damage_amount)
from participated;
```



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

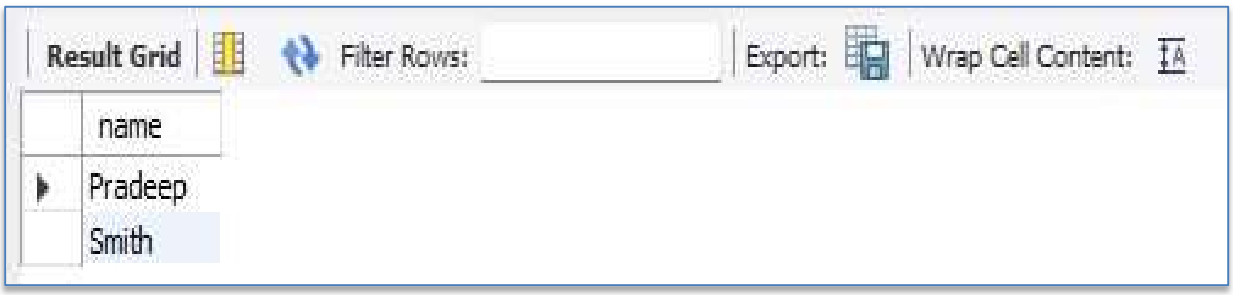
avg(damage_amount)
13600.0000

Delete the tuple whose Damage Amount is below the Average Damage Amount

```
delete from participated
where damage_amount < (select avg(damage_amount) from participated);
```

List the name of drivers whose Damage is Greater than the Average Damage Amount

```
select name
from person a, participated b
where a.driverid = b.driver_id and
damage_amount > (select avg(damage_amount) from participated);
```



Find Maximum Damage Amount.

```
select max(damage_amount)
from participated;
```

Bank Database

Question

(Week 3)

Branch (branch-name: String, branch-city: String, assets: real)

BankAccount(accno: int, branch-name: String, balance: real)

BankCustomer (customer-name: String, customer-street: String, customer-city: String)

Depositer(customer-name: String, accno: int)

LOAN (loan-number: int, branch-name: String, amount: real)

i. Create the above tables by properly specifying the primary keys and the foreign keys.

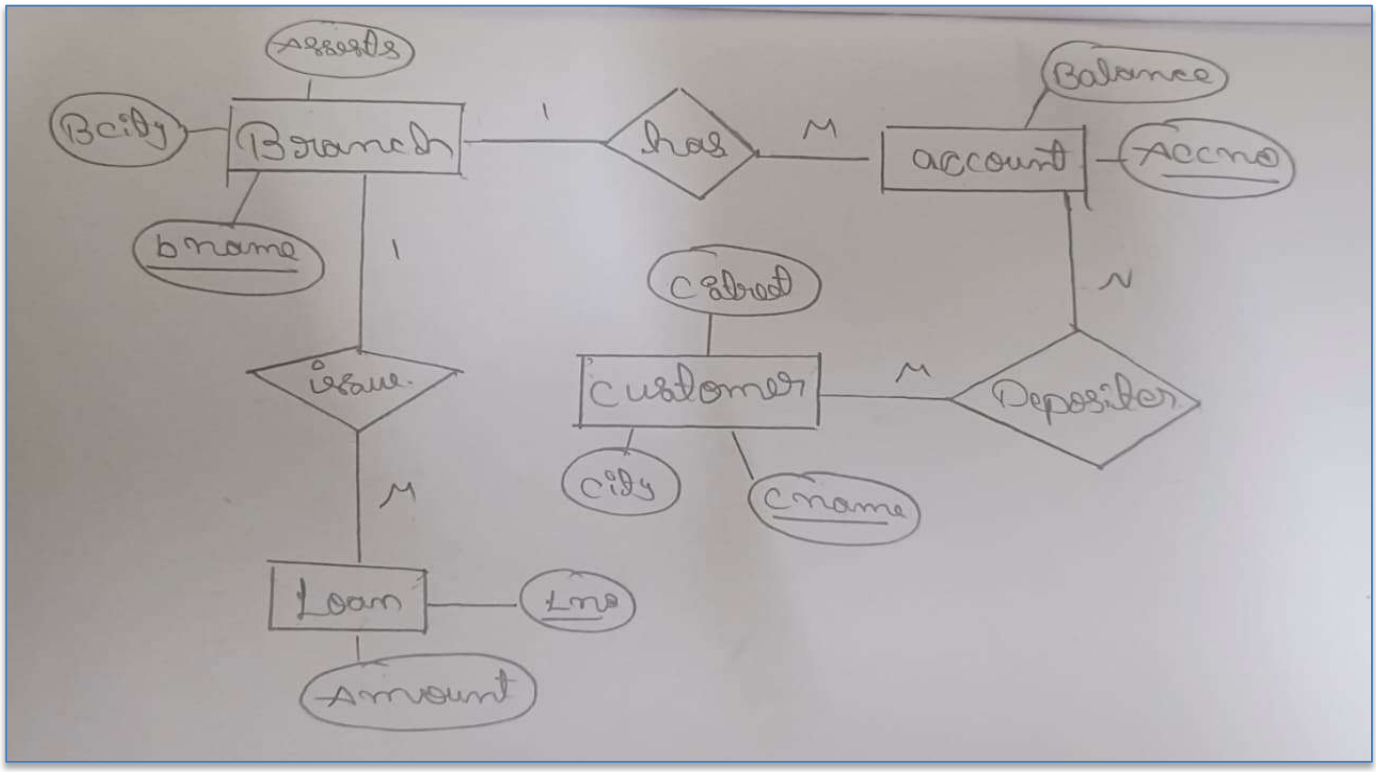
ii. Enter at least five tuples for each relation.

iii. Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.

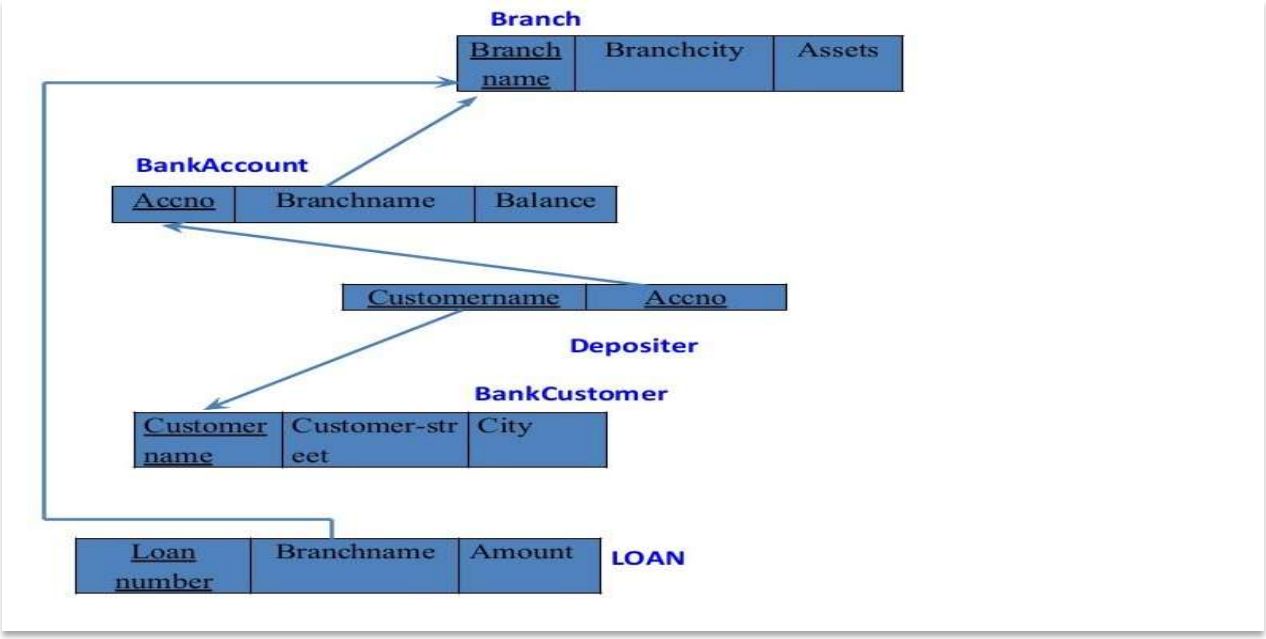
iv. Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).

v. Create a view which gives each branch the sum of the amount of all the loans at the branch.

Entity-Relationship Diagram:



Schema Diagram



Create database

```
create database bank;
use bank;
```

Create table

```
create table branch(
branch_name varchar(30) primary key,
branchcity varchar(20),
assets int
);
create table bankcustomer(
customer_name varchar(20) primary key,
customer_street varchar(30),
customer_city varchar(20)
);
create table bankaccount(
accno int primary key,
branch_name varchar(30),
balance double,
foreign key (branch_name) references branch(branch_name)
);
```

```
create table depositer(
customer_name varchar(20),
accno int,
foreign key (accno) references bankaccount(accno),
foreign key (customer_name) references bankcustomer(customer_name)
);
create table loan(
loan_number int primary key,
branch_name varchar(30),
amount double,
foreign key (branch_name) references branch(branch_name)
);
```

Structure of the table

desc branch;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
branch_name	varchar(30)	NO	PRI	NULL	
branchcity	varchar(20)	YES		NULL	
assets	int	YES		NULL	

desc bankcustomer;

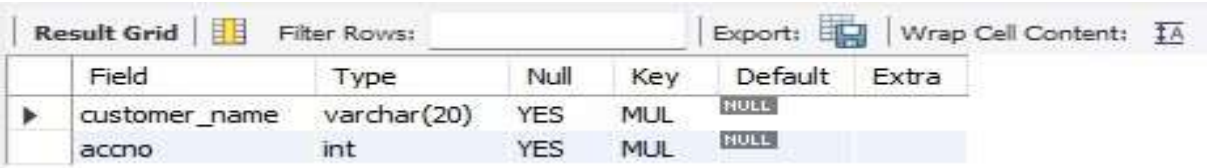
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
customer_street	varchar(30)	YES		NULL	
customer_city	varchar(20)	YES		NULL	

desc bankaccount;

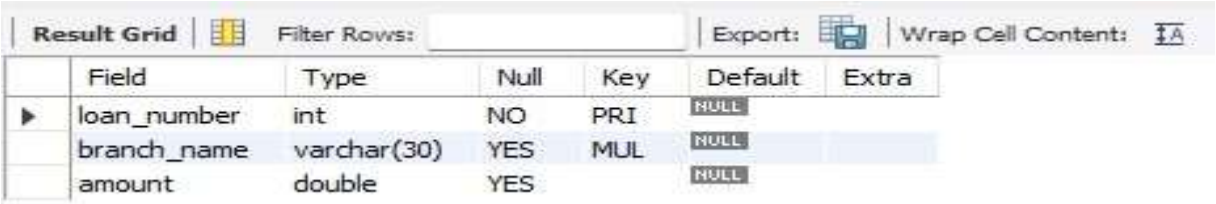
Field	Type	Null	Key	Default	Extra
accno	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
balance	double	YES		NULL	

desc depositer;



Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	YES	MUL	NULL	
accno	int	YES	MUL	NULL	

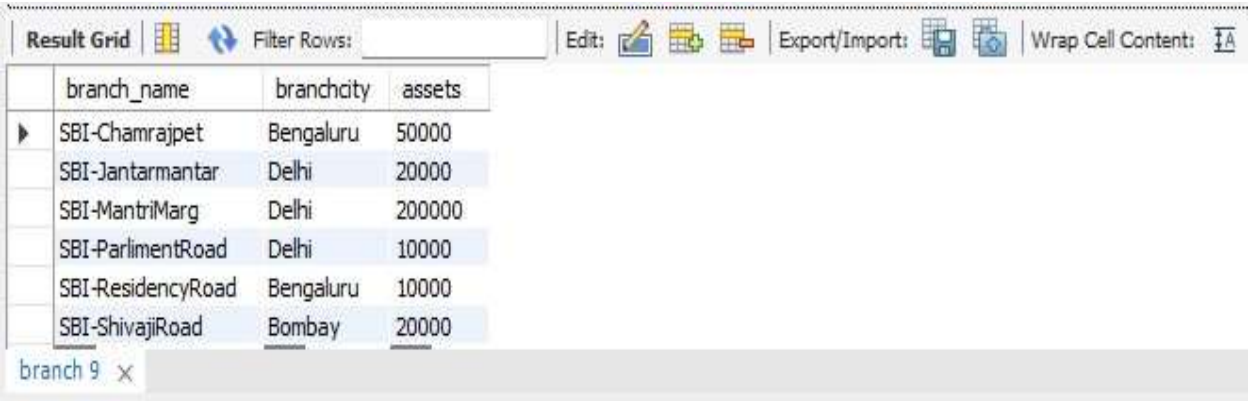
desc loan;



Field	Type	Null	Key	Default	Extra
loan_number	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
amount	double	YES		NULL	

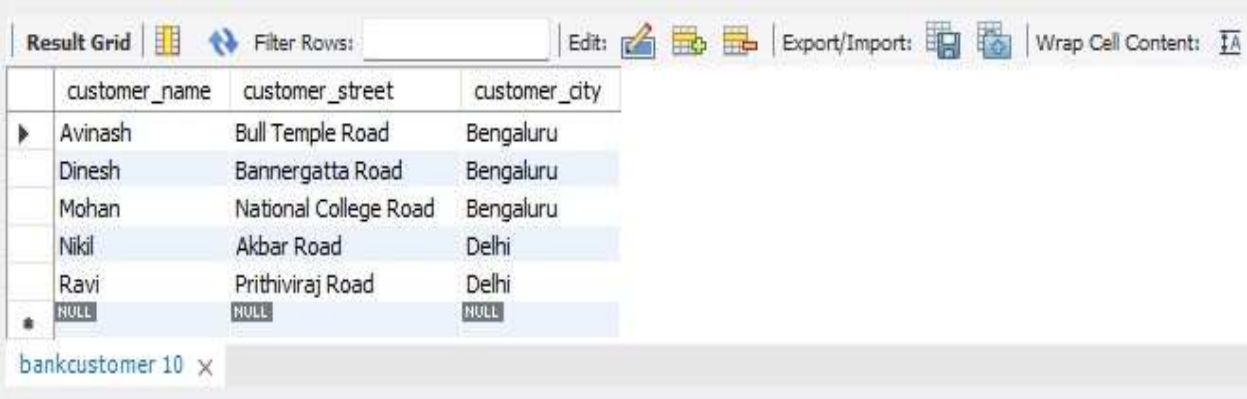
Inserting Values to the table

insert into branch
values
("SBI-Chamrajpet", "Bengaluru", 50000),
("SBI-ResidencyRoad", "Bengaluru", 10000),
("SBI-ShivajiRoad", "Bombay", 20000),
("SBI-ParlimentRoad", "Delhi", 10000),
("SBI-Jantarmantar", "Delhi", 20000),
("SBI-MantriMarg", "Delhi", 200000);
select * from branch;



branch_name	branchcity	assets
SBI-Chamrajpet	Bengaluru	50000
SBI-Jantarmantar	Delhi	20000
SBI-MantriMarg	Delhi	200000
SBI-ParlimentRoad	Delhi	10000
SBI-ResidencyRoad	Bengaluru	10000
SBI-ShivajiRoad	Bombay	20000

```
insert into bankcustomer
values
("Avinash", "Bull Temple Road", "Bengaluru"),
("Dinesh", "Bannergatta Road", "Bengaluru"),
("Mohan", "National College Road", "Bengaluru"),
("Nikil", "Akbar Road", "Delhi"),
("Ravi", "Prithiviraj Road", "Delhi");
select * from bankcustomer;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

customer_name	customer_street	customer_city
Avinash	Bull Temple Road	Bengaluru
Dinesh	Bannergatta Road	Bengaluru
Mohan	National College Road	Bengaluru
Nikil	Akbar Road	Delhi
Ravi	Prithiviraj Road	Delhi
NULL	NULL	NULL

bankcustomer 10

```
insert into bankaccount (accno, branch_name, balance)
values
(1, "SBI-Chamrajpet", 2000),
(2, "SBI-ResidencyRoad", 5000),
(3, "SBI-ShivajiRoad", 6000),
(4, "SBI-ParlimentRoad", 9000),
(5, "SBI-Jantarmantar", 8000),
(6, "SBI-ShivajiRoad", 4000),
(8, "SBI-ResidencyRoad", 4000),
(9, "SBI-ParlimentRoad", 3000),
(10, "SBI-ResidencyRoad", 5000),
(11, "SBI-Jantarmantar", 2000),
(12, "SBI-MantriMarg", 2000);
select * from bankaccount;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

accno	branch_name	balance
1	SBI-Chamrajpet	2000
2	SBI-ResidencyRoad	5000
3	SBI-ShivajiRoad	6000
4	SBI-ParlimentRoad	9000
5	SBI-Jantarmantar	8000
6	SBI-ShivajiRoad	4000
8	SBI-ResidencyRoad	4000
9	SBI-ParlimentRoad	3000
10	SBI-ResidencyRoad	5000
11	SBI-Jantarmantar	2000
12	SBI-MantriMarg	2000
NULL	NULL	NULL

bankaccount 11

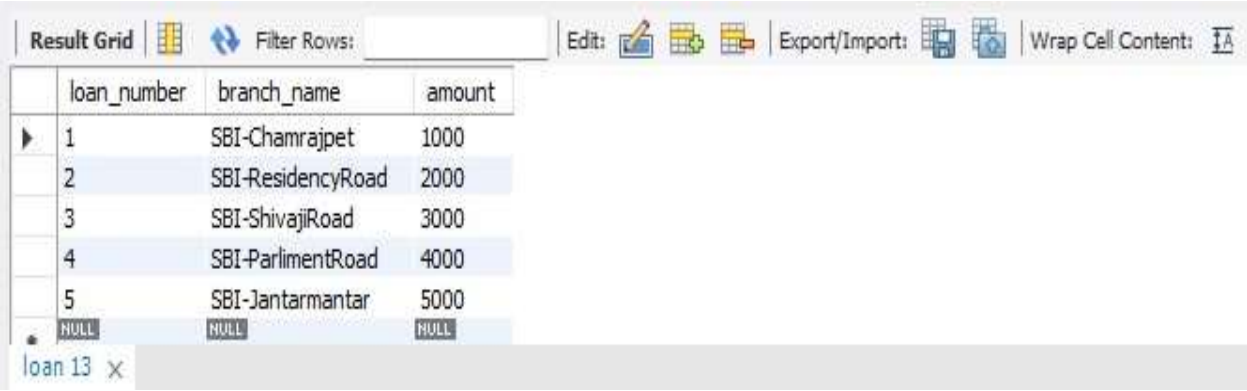
```
insert into depositer
values
("Avinash", 1),
("Dinesh", 2),
("Nikil", 4),
("Ravi", 5),
("Avinash", 8),
("Nikil", 9),
("Dinesh", 10),
("Nikil", 11),
("Nikil", 12);
select * from depositer;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	accno
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
Nikil	12

depositer 12

```
insert into loan
values
(1,"SBI-Chamrajpet", 1000),
(2,"SBI-ResidencyRoad", 2000),
(3,"SBI-ShivajiRoad", 3000),
(4,"SBI-ParlimentRoad", 4000),
(5,"SBI-Jantarmantar", 5000);
select * from loan;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

loan_number	branch_name	amount
1	SBI-Chamrajpet	1000
2	SBI-ResidencyRoad	2000
3	SBI-ShivajiRoad	3000
4	SBI-ParlimentRoad	4000
5	SBI-Jantarmantar	5000
NULL	NULL	NULL

loan 13

QUERIES

Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.

```
select branch_name, CONCAT(assets/100000, "lakhs" )
assets_in_lakhs
from branch;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch_name	assets_in_lakhs
SBI-Chamrajpet	0.5000lakhs
SBI-Jantarmantar	0.2000lakhs
SBI-MantriMarg	2.0000lakhs
SBI-ParlimentRoad	0.1000lakhs
SBI-ResidencyRoad	0.1000lakhs
SBI-ShivajiRoad	0.2000lakhs

Result 14

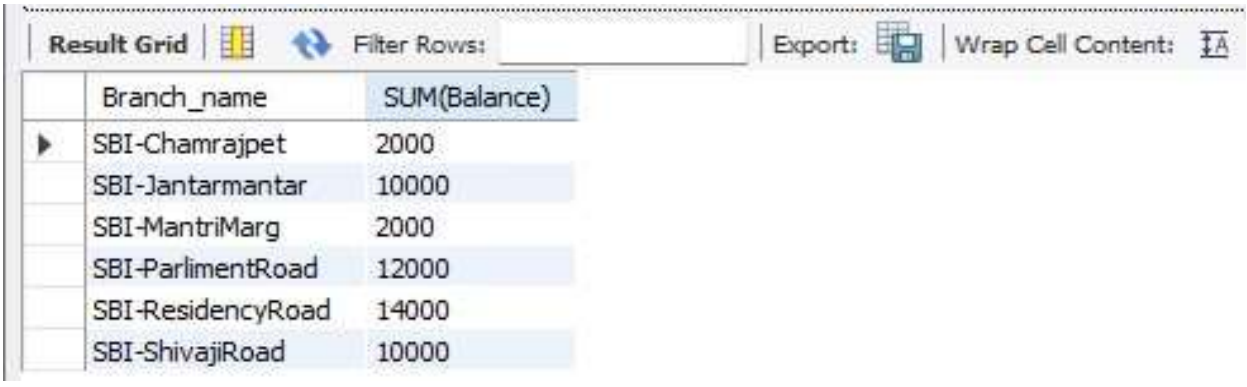
Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).

```
select d.customer_name
from depositer d,bankaccount b
where b.branch_name = "SBI-ResidencyRoad" and d.accno = b.accno
group by d.customer_name
having count(d.accno) >= 2;
```

customer_name
Dinesh

Create a view which gives each branch the sum of the amount of all the loans at the branch.

```
create view sum_of_loan
as select Branch_name, SUM(Balance)
from BankAccount
group by Branch_name;

select * from sum_of_loan;
```



Branch_name	SUM(Balance)
SBI-Chamrajpet	2000
SBI-Jantarmantar	10000
SBI-MantriMarg	2000
SBI-ParlimentRoad	12000
SBI-ResidencyRoad	14000
SBI-ShivajiRoad	10000

More Queries on Bank Database

Question

(Week 4)

Branch (branch-name: String, branch-city: String, assets: real)

BankAccount(accno: int, branch-name: String, balance: real)

BankCustomer (customer-name: String, customer-street: String, customer-city: String)

Depositer(customer-name: String, accno: int)

LOAN (loan-number: int, branch-name: String, amount: real)

Borrower (customer-name: String, loan-number: int)

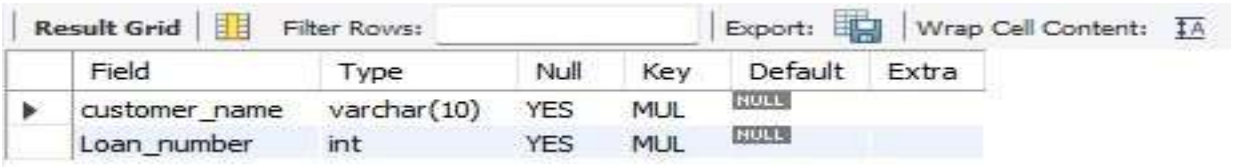
i. Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).
ii. Find all customers who have a loan at the bank but do not have an account.
iii. Find all customers who have both an account and a loan at the Bangalore branch
iv. Find the names of all branches that have greater assets than all branches located in Bangalore.
v. Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).
vi. Update the Balance of all accounts by 5%

Create table

```
create table borrower (
customer_name varchar(10),
Loan_number int,
foreign key (customer_name) references bankcustomer(customer_name),
foreign key (Loan_number) references loan(loan_number)
);
```

Structure of the table

desc borrower;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
customer_name	varchar(10)	YES	MUL	NULL	
Loan_number	int	YES	MUL	NULL	

Inserting Values to the table

```
insert into borrower values
("Avinash", 1),
("Dinesh", 2),
("Mohan", 3),
("Nikil", 4),
("Ravi", 5);
select * from borrower;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name	Loan_number
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5

QUERIES

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

```
select d.customer_name
from depositer d, branch b, bankaccount ba
where d.accno = ba.accno and ba.branch_name = b.branch_name and b.branchcity = "Delhi"
group by d.customer_name
Having count(distinct(B.branch_name)) =
(select count(branch_name) from branch where branchcity = 'Delhi');
```

Find all customers who have a loan at the bank but do not have an account.

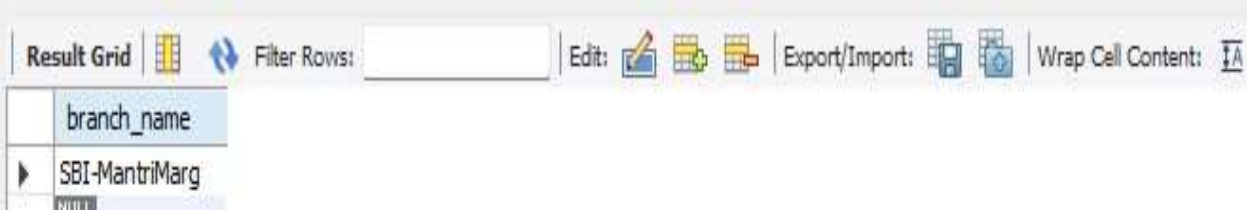
```
SELECT DISTINCT bo.customer_name
FROM borrower bo
WHERE bo.customer_name NOT IN (SELECT customer_name FROM depositer);
```

Find all customers who have both an account and a loan at the Bangalore branch

```
select bo.customer_name
from borrower bo, bankcustomer bc
where bo.customer_name in (select customer_name from depositer) and
bc.customer_city  in (select branchcity from branch where branchcity =
"Bengaluru") and
bc.customer_name = bo.customer_name
group by bo.customer_name;
```

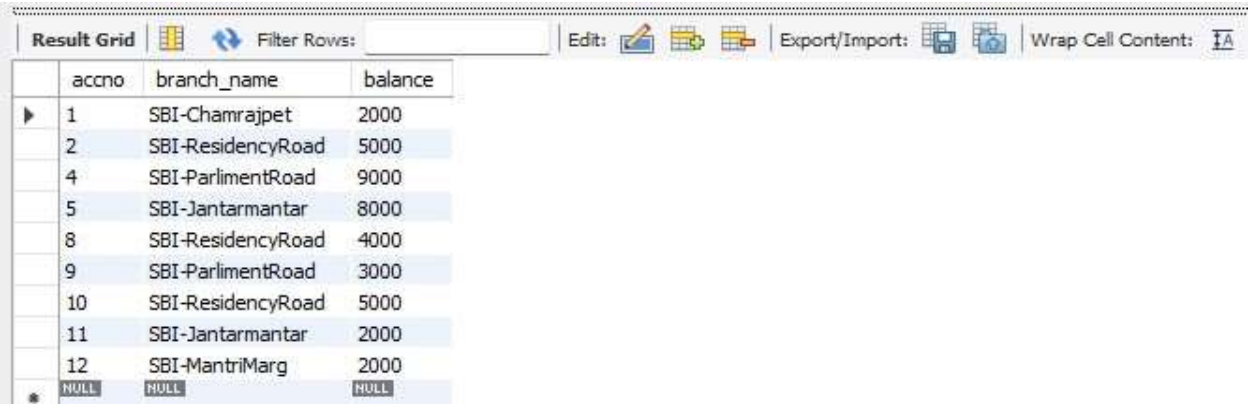
Find the names of all branches that have greater assets than all branches located in Bangalore.

```
select branch_name
from branch
where assets > all (select assets from branch where branchcity =
"Bengaluru");
```



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

```
DELETE FROM bankaccount
WHERE branch_name IN (
SELECT branch_name FROM branch WHERE branchcity = 'Bombay');
select * from bankaccount;
```

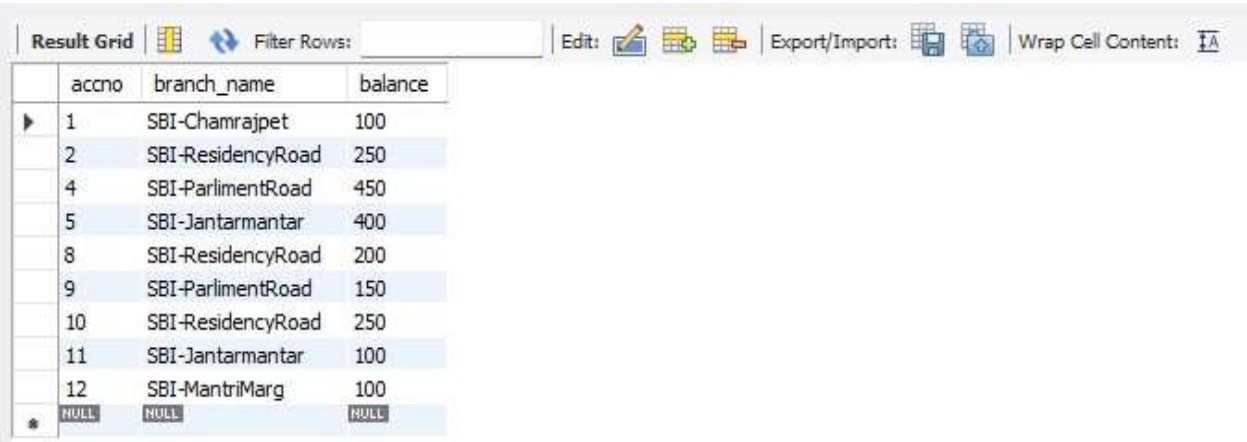


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

accno	branch_name	balance
1	SBI-Chamrajpet	2000
2	SBI-ResidencyRoad	5000
4	SBI-ParlimentRoad	9000
5	SBI-Jantarmantar	8000
8	SBI-ResidencyRoad	4000
9	SBI-ParlimentRoad	3000
10	SBI-ResidencyRoad	5000
11	SBI-Jantarmantar	2000
12	SBI-MantriMarg	2000
NULL	NULL	NULL

Update the Balance of all accounts by 5%

```
update bankaccount
set balance =  balance * 0.05
where balance;
select * from bankaccount;
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

accno	branch_name	balance
1	SBI-Chamrajpet	100
2	SBI-ResidencyRoad	250
4	SBI-ParlimentRoad	450
5	SBI-Jantarmantar	400
8	SBI-ResidencyRoad	200
9	SBI-ParlimentRoad	150
10	SBI-ResidencyRoad	250
11	SBI-Jantarmantar	100
12	SBI-MantriMarg	100
NULL	NULL	NULL

Employee Database

Question

(Week 5)

i. Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys.

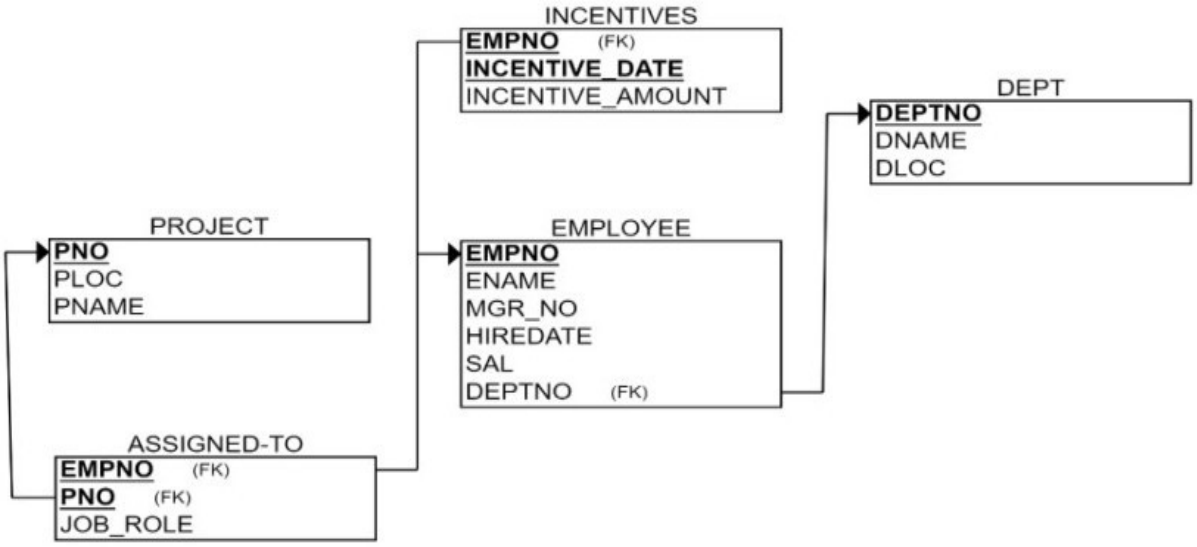
ii. Enter greater than five tuples for each table.

iii. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru

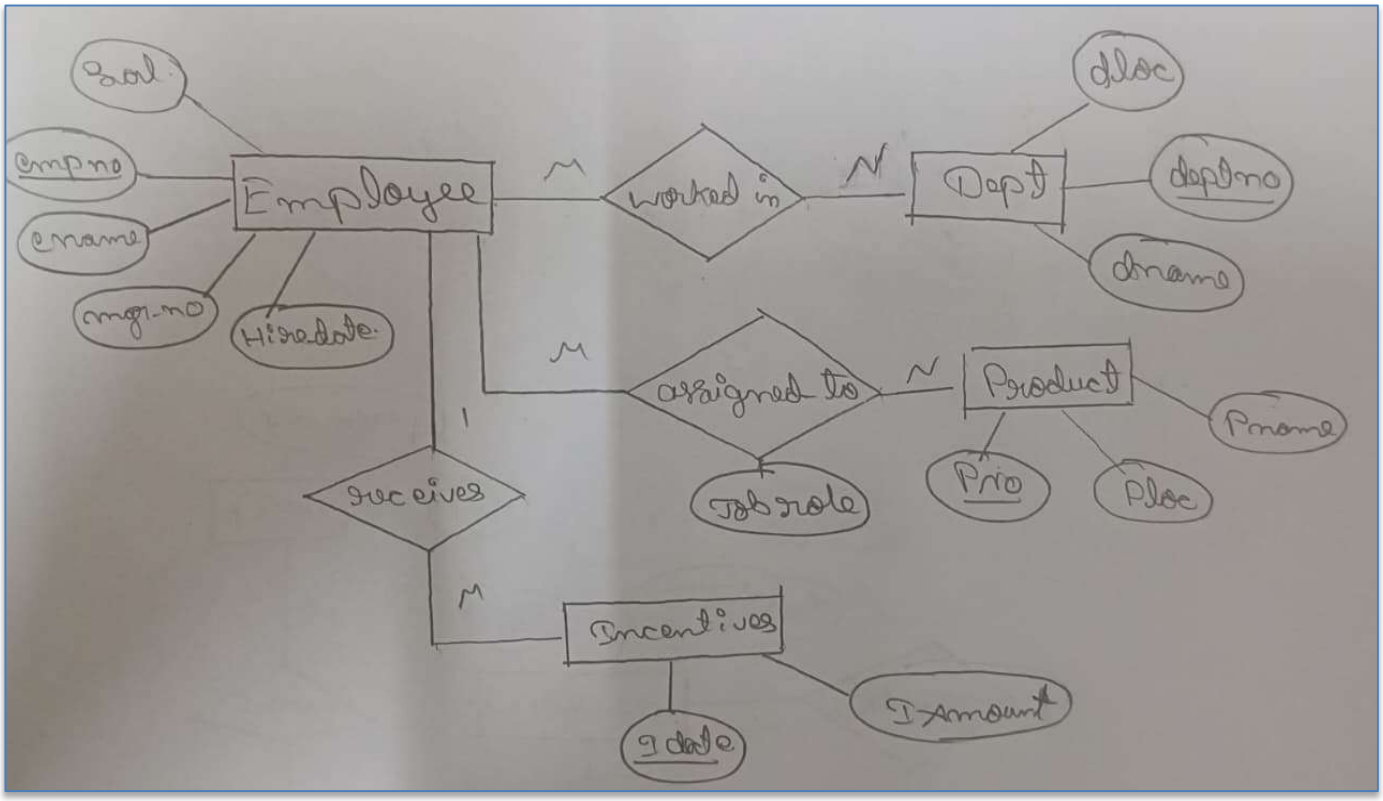
iv. Get Employee ID's of those employees who didn't receive incentives

v. Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.

Schema Diagram



Entity-Relationship Diagram:



Create database

create database company;

use company;

Create table

```
create table dept (
dept_no int primary key,
dname varchar(20),
dloc varchar(10)
);
```

```
create table employee(
empno int primary key,
ename varchar(20),
mgr_no int,
hireddate date,
sal float,
deptno int,
foreign key (deptno) references dept(dept_no)
);
create table project(
pno int primary key,
ploc varchar(10) ,
pname varchar(10));
create table assignedto(
empno int,
pno int,
foreign key(empno) references employee(empno) ,
foreign key(pno) references project(pno),
jobrole varchar(10));
create table incentives(
empno int,
incentivedate date,
incentiveamount float,
foreign key (empno) references employee(empno));
```

Structure of the table

desc dept;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
dept_no	int	NO	PRI	NULL	
dname	varchar(20)	YES		NULL	
dloc	varchar(10)	YES		NULL	

desc employee;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
empno	int	NO	PRI	NULL	
ename	varchar(20)	YES		NULL	
mgr_no	int	YES		NULL	
hireddate	date	YES		NULL	
sal	float	YES		NULL	
deptno	int	YES	MUL	NULL	

desc project;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

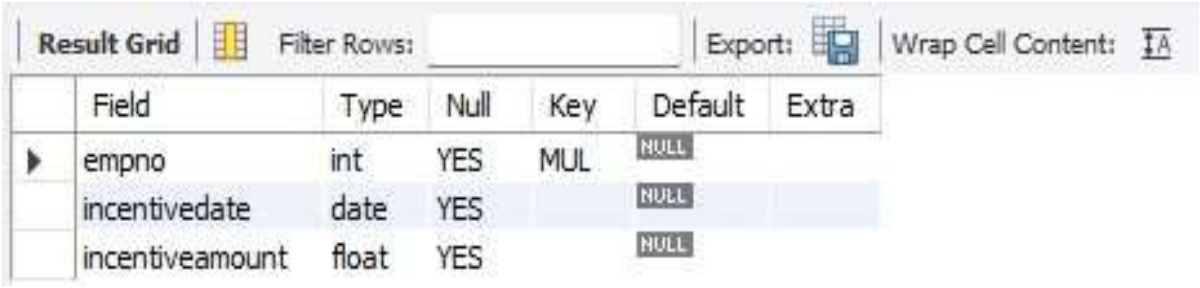
Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
ploc	varchar(10)	YES		NULL	
pname	varchar(10)	YES		NULL	

desc assignedto;

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
pno	int	YES	MUL	NULL	
jobrole	varchar(10)	YES		NULL	

desc incentives;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
incentivedate	date	YES		NULL	
incentiveamount	float	YES		NULL	

Inserting Values to the table

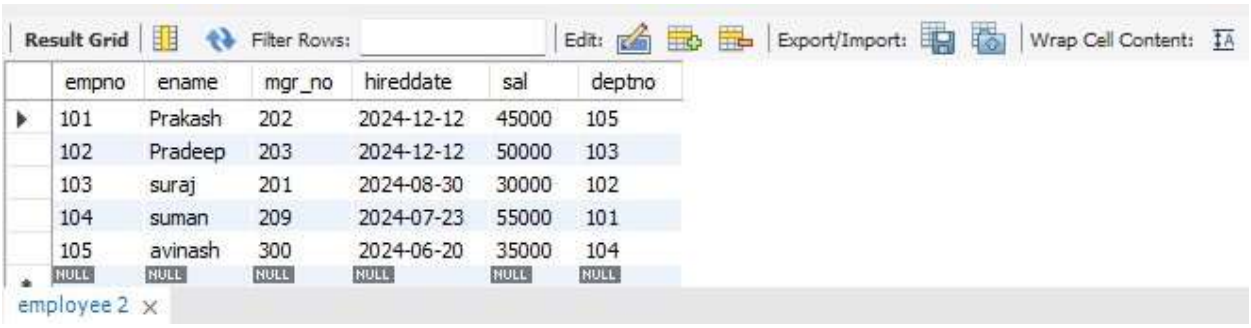
insert into dept values
(101, "marketing", "Bengaluru"),
(102, "designing", "Mysuru"),
(103, "sales", "Hyderabad"),
(104, "Finance;", "Mumbai"),
(105, "Customer service", "Chennai");
select * from dept;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

dept_no	dname	dloc
101	marketing	Bengaluru
102	designing	Mysuru
103	sales	Hyderabad
104	Finance;	Mumbai
105	Customer service	Chennai

dept 11 ×

insert into employee values
(101, "Prakash", 202, "2024-12-12", 45000, 105),
(102, "Pradeep", 203, "2024-12-12", 50000, 103),
(103, "suraj", 201, "2024-08-30", 30000, 102),
(104, "suman", 209, "2024-07-23", 55000, 101),
(105, "avinash", 300, "2024-06-20", 35000, 104);
select * from employee;



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	ename	mgr_no	hireddate	sal	deptno
101	Prakash	202	2024-12-12	45000	105
102	Pradeep	203	2024-12-12	50000	103
103	suraj	201	2024-08-30	30000	102
104	suman	209	2024-07-23	55000	101
105	avinash	300	2024-06-20	35000	104
NULL	NULL	NULL	NULL	NULL	NULL

employee 2 ×

```
insert into project
values
(505,"Bengaluru","aaa"),
(504,"Hyderabad","bbb"),
(503,"Mysuru","ccc"),
(502,"Mumbai","ddd"),
(501,"Chennai","eee");
select * from project;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

pno	ploc	pname
501	Chennai	eee
502	Mumbai	ddd
503	Mysuru	ccc
504	Hyderabad	bbb
505	Bengaluru	aaa
NULL	NULL	NULL

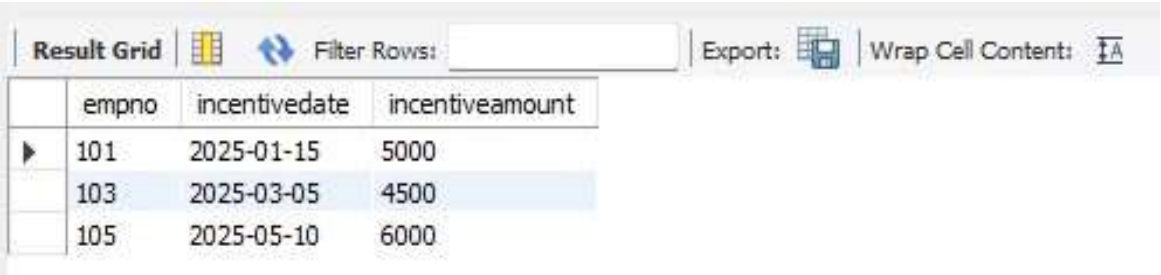
project 3 ×

```
insert into assignedto
values
(101,504,"zzz"),
(102,503,"yyy"),
(103,501,"xxx"),
(104,505,"uuu"),
(105,502,"vvv");
select * from assignedto;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empno	pno	jobrole
101	504	zzz
102	503	yyy
103	501	xxx
104	505	uuu
105	502	vvv

INSERT INTO INCENTIVES
VALUES
(101, '2025-01-15', 5000.00),
(103, '2025-03-05', 4500.00),
(105, '2025-05-10', 6000.00);
select * from incentives;



Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empno	incentivedate	incentiveamount
101	2025-01-15	5000
103	2025-03-05	4500
105	2025-05-10	6000

QUERIES

Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru

```
SELECT DISTINCT empno
FROM assignedto
JOIN project ON assignedto.pno = project.pno
WHERE project.ploc IN ('Bengaluru', 'Mysuru', 'Hyderabad');
```

empno
102
101
104

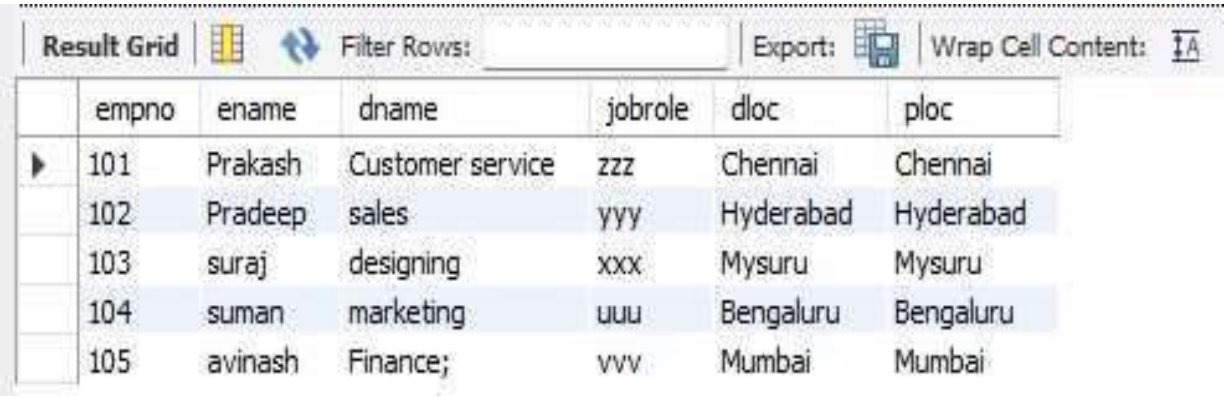
Get Employee ID's of those employees who didn't receive incentives

```
SELECT EMPLOYEE.EMPNO

FROM EMPLOYEE

LEFT JOIN INCENTIVES

ON EMPLOYEE.EMPNO = INCENTIVES.EMPNO

WHERE incentives.empno IS NULL ;
```

EMPNO
104
102

Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.

```
select e.empno, e.ename, d.dname, a.jobrole, d.dloc, p.ploc from
employee e, dept d, assignedto a, project p where p.ploc = d.dloc and
d.dept_no = e.deptno and a.empno = e.empno order by e.empno;
```



empno	ename	dname	jobrole	dloc	ploc
101	Prakash	Customer service	zzz	Chennai	Chennai
102	Pradeep	sales	yyy	Hyderabad	Hyderabad
103	suraj	designing	xxx	Mysuru	Mysuru
104	suman	marketing	uuu	Bengaluru	Bengaluru
105	avinash	Finance;	vvv	Mumbai	Mumbai

More Queries on Employee Database

Question
(Week 6)

i. Using Scheme diagram (under Program-5), Create tables by properly specifying the primary keys and the foreign keys.

ii. Enter greater than five tuples for each table.

iii. List the name of the managers with the maximum employees iv. Display those managers name whose salary is more than average salary of his employee.

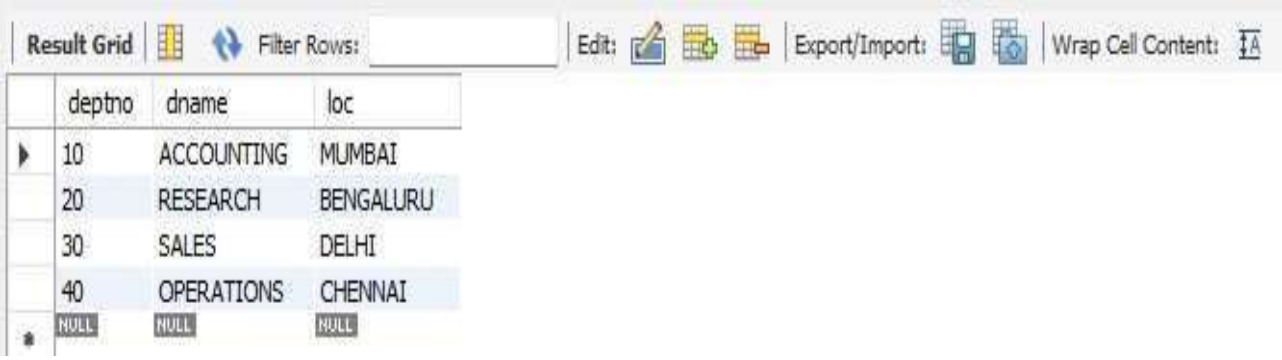
v. Find the name of the second top level managers of each department.

vi. Find the employee details who got second maximum incentive in January 2019.

vii. Display those employees who are working in the same department where his manager is working

Inserting Values to the table

```
INSERT INTO dept VALUES
(10,'ACCOUNTING','MUMBAI'),
(20,'RESEARCH','BENGALURU'),
(30,'SALES','DELHI'),
(40,'OPERATIONS','CHENNAI');
Select * from dept;
```



deptno	dname	loc
10	ACCOUNTING	MUMBAI
20	RESEARCH	BENGALURU
30	SALES	DELHI
40	OPERATIONS	CHENNAI
NULL	NULL	NULL

```
INSERT INTO emp VALUES (7369,'Adarsh',7902,'2012-
1217','80000.00','20'),
(7499,'Shruthi',7698,'2013-02-20','16000.00','30'),
(7521,'Anvitha',7698,'2015-02-22','12500.00','30'),
(7566,'Tanvir',7839,'2008-04-02','29750.00','20'),
(7654,'Ramesh',7698,'2014-09-28','12500.00','30'),
(7698,'Kumar',7839,'2015-05-01','28500.00','30'),
(7782,'CLARK',7839,'2017-06-09','24500.00','10'),
(7788,'SCOTT',7566,'2010-12-09','30000.00','20'),
('7839','KING',null,'2009-11-17','50000.00','10'),
('7844','TURNER',7698,'2010-09-08','15000.00','30'),
('7876','ADAMS',7788,'2013-01-12','11000.00','20'),
('7900','JAMES',7698,'2017-12-03','9500.00','30'),
('7902','FORD','7566','2010-12-03','30000.00','20');
select * from emp;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

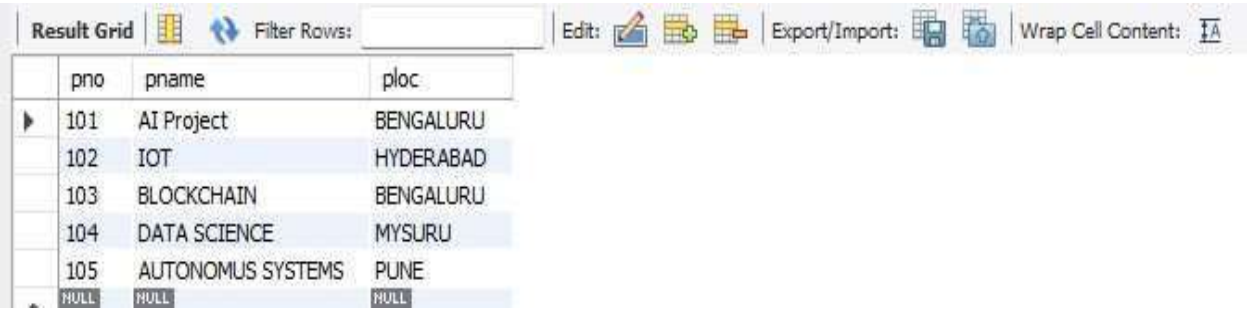
empno	ename	mgr_no	hiredate	sal	deptno
7369	Adarsh	7902	2012-12-17	80000.00	20
7499	Shruthi	7698	2013-02-20	16000.00	30
7521	Anvitha	7698	2015-02-22	12500.00	30
7566	Tanvir	7839	2008-04-02	29750.00	20
7654	Ramesh	7698	2014-09-28	12500.00	30
7698	Kumar	7839	2015-05-01	28500.00	30
7782	CLARK	7839	2017-06-09	24500.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
7839	KING	NULL	2009-11-17	50000.00	10
7844	TURNER	7698	2010-09-08	15000.00	30
7876	ADAMS	7788	2013-01-12	11000.00	20
7900	JAMES	7698	2017-12-03	9500.00	30
7902	FORD	7566	2010-12-03	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

INSERT INTO incentives VALUES(7499,'2019-02-01',5000.00),
(7521,'2019-03-01',2500.00),
(7566,'2022-02-01',5070.00),
(7654,'2020-02-01',2000.00), (7654,'2022-04-01',879.00),
(7521,'2019-02-01',8000.00), (7698,'2019-03-01',500.00),
(7698,'2020-03-01',9000.00),
(7698,'2022-04-01',4500.00);
select * from incentives;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	incentive_date	incentive_amount
7499	2019-02-01	5000.00
7521	2019-02-01	8000.00
7521	2019-03-01	2500.00
7566	2022-02-01	5070.00
7654	2020-02-01	2000.00
7654	2022-04-01	879.00
7698	2019-03-01	500.00
7698	2020-03-01	9000.00
7698	2022-04-01	4500.00
NULL	NULL	NULL

```
INSERT INTO project VALUES(101,'AI Project','BENGALURU'),
(102,'IOT','HYDERABAD'),
(103,'BLOCKCHAIN','BENGALURU'),
(104,'DATA     SCIENCE','MYSURU'),
(105,'AUTONOMUS
SYSTEMS','PUNE');
select * from project;
```



pno	pname	ploc
101	AI Project	BENGALURU
102	IOT	HYDERABAD
103	BLOCKCHAIN	BENGALURU
104	DATA SCIENCE	MYSURU
105	AUTONOMUS SYSTEMS	PUNE
NULL	NULL	NULL

```
INSERT INTO assigned_to VALUES
(7499,101,'Software Engineer'),
(7521,101,'Software Architect'),
(7566,101,'Project Manager'),
(7654,102,'Sales'),
(7521,102,'Software Engineer'),
(7499,102,'Software
Engineer'), (7654,103,'Cyber
Security'); select * from
assigned_to;
```

empno	pno	job_role
7499	101	Software Engineer
7499	102	Software Engineer
7521	101	Software Architect
7521	102	Software Engineer
7566	101	Project Manager
7654	102	Sales
7654	103	Cyber Security

QUERIES

List the name of the managers with the maximum employees

```
SELECT m.ename, count(*)
FROM emp e,emp m
WHERE e.mgr_no = m.empno
GROUP BY m.ename
HAVING count(*) =(SELECT MAX(mycount) from (SELECT COUNT(*)
mycount FROM emp GROUP BY mgr_no) a);
```

	ename	count(*)
▶	Kumar	5

Display those managers name whose salary is more than average salary of his employee.

```
SELECT *
FROM emp m
WHERE m.empno IN (SELECT mgr_no FROM emp) AND
m.sal > (SELECT avg(e.sal) FROM emp e WHERE e.mgr_no = m.empno );
```

	empno	ename	mgr_no	hiredate	sal	deptno
▶	7698	Kumar	7839	2015-05-01	28500.00	30
	7839	KING	NULL	2009-11-17	50000.00	10
	7788	SCOTT	7566	2010-12-09	30000.00	20
	NULL	NULL	NULL	NULL	NULL	NULL

Find the name of the second top level managers of each department.

```
select distinct m.mgr_no from emp e, emp m where
e.mgr_no = m.mgr_no and e.deptno = m.deptno and
e.empno in (select distinct m.mgr_no from emp e, emp m where e.mgr_no =
m.mgr_no and e.deptno = m.deptno);
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

mgr_no
7839
7566

Find the employee details who got second maximum incentive in January 2019

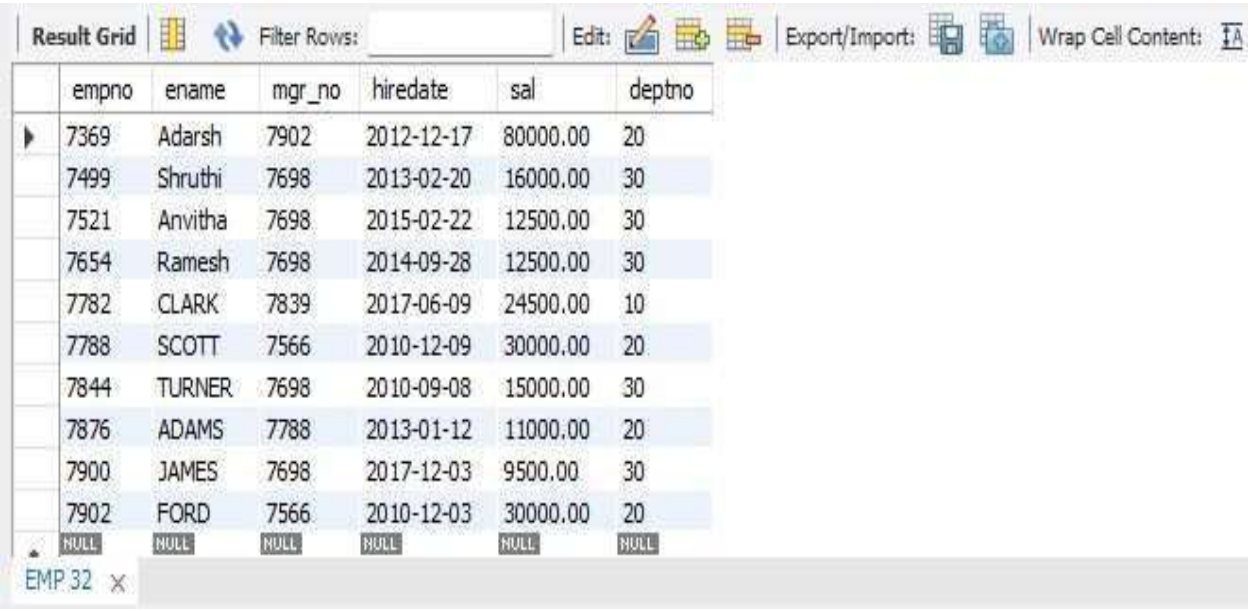
```
select *  from emp
e,incentives i where
e.empno=i.empno and
2 = ( select count(*) from incentives j where i.incentive_amount <=
j.incentive_amount );
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empno	ename	mgr_no	hiredate	sal	deptno	empno	incentive_date	incentive_amount
7521	Anvitha	7698	2015-02-22	12500.00	30	7521	2019-02-01	8000.00

Display those employees who are working in the same department where his manager is working

```
SELECT * FROM EMP E
WHERE E.DEPTNO = (SELECT E1.DEPTNO FROM EMP E1 WHERE
E1.EMPNO=E.MGR_NO);
```



Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	ename	mgr_no	hiredate	sal	deptno
7369	Adarsh	7902	2012-12-17	80000.00	20
7499	Shruthi	7698	2013-02-20	16000.00	30
7521	Anvitha	7698	2015-02-22	12500.00	30
7654	Ramesh	7698	2014-09-28	12500.00	30
7782	CLARK	7839	2017-06-09	24500.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
7844	TURNER	7698	2010-09-08	15000.00	30
7876	ADAMS	7788	2013-01-12	11000.00	20
7900	JAMES	7698	2017-12-03	9500.00	30
7902	FORD	7566	2010-12-03	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

EMP 32

Supplier Database

Question
(Week 7)

i. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.

ii. Insert appropriate records in each table.

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

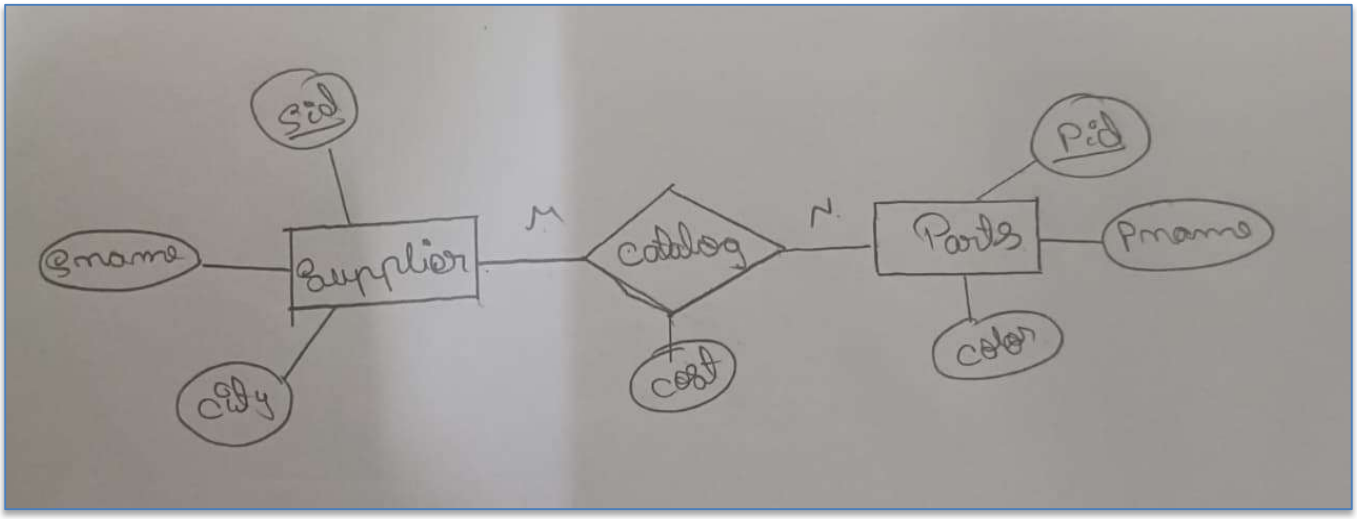
iv. Find the snames of suppliers who supply every part.

v. Find the snames of suppliers who supply every red part.

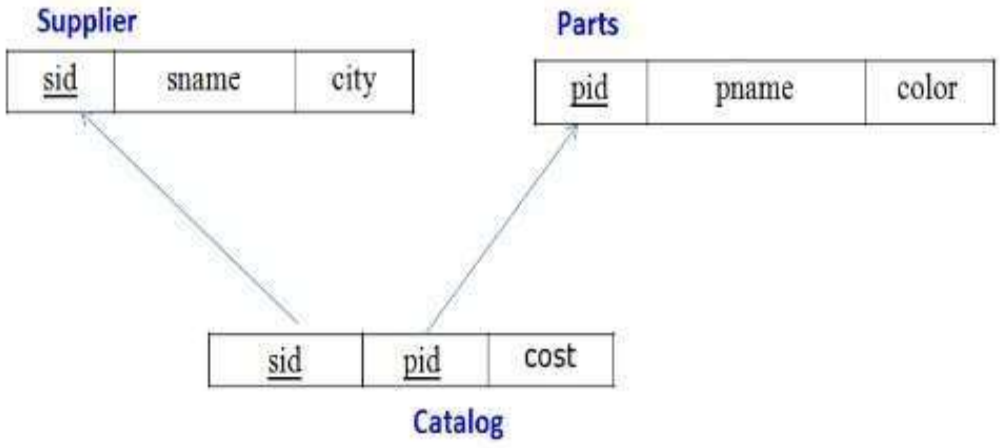
vi. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

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

Entity-Relationship Diagram:



Schema Diagram



Create database

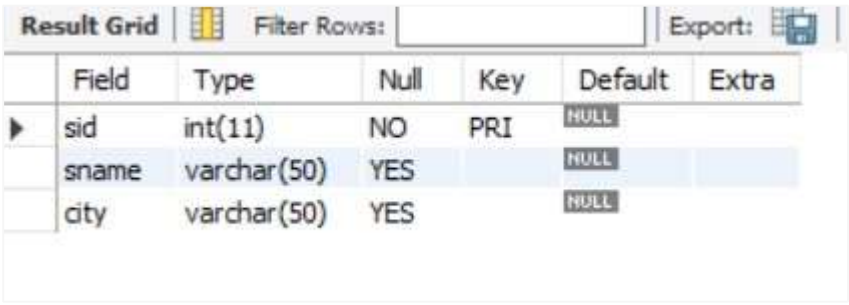
```
Create database Supply;
use Supply;
```

Create table

```
create table suppliers (
sid int primary key,
 sname varchar(50),
 city varchar(50)
);
create table parts (
pid int primary key,
 pname varchar(50),
 color varchar(20)
);
create table catalog (
sid int,
pid int,
cost int,
 foreign key (sid) references suppliers(sid),
 foreign key (pid) references parts(pid)
);
```

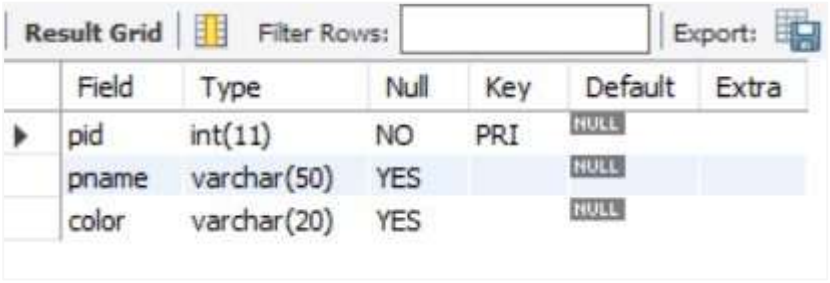
Structure of the table

desc suppliers;



Field	Type	Null	Key	Default	Extra
sid	int(11)	NO	PRI	NULL	
sname	varchar(50)	YES		NULL	
city	varchar(50)	YES		NULL	

desc parts;



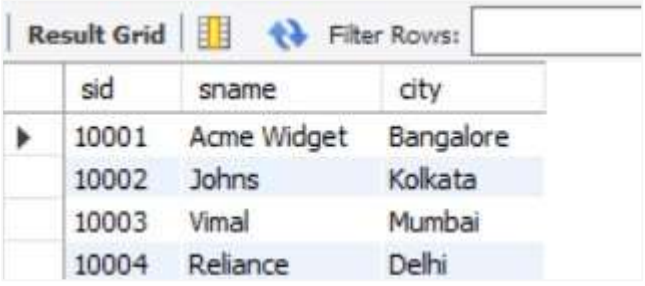
Field	Type	Null	Key	Default	Extra
pid	int(11)	NO	PRI	NULL	
pname	varchar(50)	YES		NULL	
color	varchar(20)	YES		NULL	

desc catalog;

Field	Type	Null	Key	Default	Extra
sid	int(11)	YES	MUL	NULL	
pid	int(11)	YES	MUL	NULL	
cost	int(11)	YES		NULL	

Inserting Values to the table

insert into suppliers (sid, sname, city) values
(10001, 'Acme Widget', 'Bangalore'),
(10002, 'Johns', 'Kolkata'),
(10003, 'Vimal', 'Mumbai'), (10004, 'Reliance', 'Delhi'); select * from suppliers;



sid	sname	city
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi

```
insert into parts (pid, pname, color) values
(20001, 'Book', 'Red'),
(20002, 'Pen', 'Red'),
(20003, 'Pencil', 'Green'),
(20004, 'Mobile', 'Green'), (20005, 'Charger', 'Black'); select * from parts;
```

Result Grid | Filter Rows:

pid	pname	color
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black

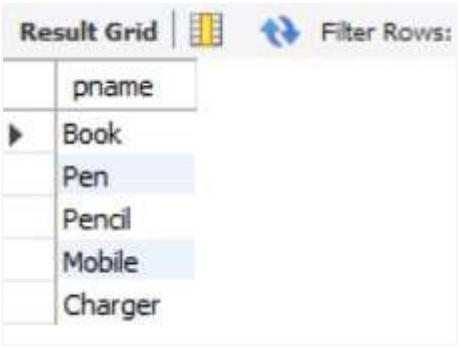
```
insert into catalog (sid, pid, cost) values
(10001, 20001, 10),
(10001, 20002, 10),
(10001, 20003, 30),
(10001, 20004, 30),
(10001, 20005, 10),
(10002, 20001, 20),
(10002, 20002, 20),
(10003, 20002, 30),
(10003, 20003, 20),
(10004, 20003, 40);
```

Result Grid | Filter Rows:

sid	pid	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	30
10001	20005	10
10002	20001	20
10002	20002	20
10003	20002	30
10003	20003	20
10004	20003	40

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

select distinct p.pname from parts p join catalog c on p.pid = c.pid;



Find the snames of suppliers who supply every part.

select s.sname from suppliers s join catalog c on s.sid = c.sid group by s.sid, s.sname having count(distinct c.pid) = (select count(*) from parts);

Find the snames of suppliers who supply every red part.

select s.sname from suppliers s join catalog c on s.sid = c.sid where c.pid in (select pid from parts where color = 'Red') group by s.sid, s.sname having count(distinct c.pid) = (select count(*) from parts where color = 'Red');

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

select p.pname from parts p where p.pid in (select pid from catalog where sid = 10001and pid not in (select pid from catalog where sid <> 10001));

Result Grid

	pname
▶	Mobile
	Charger

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 join (select pid, avg(cost) as avg_cost from catalog group by pid) x on c.pid = x.pid where c.cost > x.avg_cost;

Result Grid

	sid
▶	10002
	10003
	10004

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

select p.pname, s.sname from parts p join catalog c on p.pid = c.pid join suppliers s on c.sid = s.sid where (c.pid, c.cost) in (select pid, max(cost)
from catalog group by pid);

Result Grid | Filter Rows:

	pname	sname
▶	Mobile	Acme Widget
	Charger	Acme Widget
	Book	Johns
	Pen	Vimal
	Pencil	Reliance

NoSQL Student Database

Question
(Week 8)

Perform the following DB operations using MongoDB.

i. Create a database “Student” with the following attributesRollno, Age, ContactNo, Email-Id.

ii. Insert appropriate values

iii. Write query to update Email-Id of a student with rollno 10.

iv. Replace the student name from “ABC” to “FEM” of rollno 11.

v. Export the created table into local file system

vi. Drop the table.

vii. Import a given csv dataset from local file system into mongodb collection.

Create Database

```
test> db.createCollection("student");
{ ok: 1 }
```

Inserting values

```
test> db.student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936655093569dc27263b112') }
}
test> db.student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936656893569dc27263b113') }
}
test> db.student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936658293569dc27263b114') }
}
test> db.student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6936659a93569dc27263b115') }
}
test> db.student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693665af93569dc27263b116') }
}
```

View values

```
test> db.student.find()
[
  {
    _id: ObjectId('6936655093569dc27263b112'),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId('6936656893569dc27263b113'),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId('6936658293569dc27263b114'),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId('6936659a93569dc27263b115'),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId('693665af93569dc27263b116'),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

Queries:

Write query to update Email-Id of a student with rollno 10

```
test> db.student.update({ RollNo: 10 }, { $set: { email: "Abhinav@gmail.com" } })
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Replace the student name from “ABC” to “FEM” of rollno 11.

```
test> db.student.insert({RollNo:11,Age:22,Name:"ABC",Cont:2276,email:"rea.de9@gmail.com"});
{ncaught:
  acknowledged: true,ed token (4:0)
  insertedIds: { '0': ObjectId('6936690093569dc27263b118') }
} 2 |
```

```
test> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
  {
    _id: ObjectId('6936690093569dc27263b118'),
    RollNo: 11,
    Age: 22,
    Name: 'FEM',
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  }
]
```

Drop the table

```
test> db.student.drop();
true
test> db.student.find();
```

NoSQL Customer Database

Question
(Week 9)

Perform the following DB operations using MongoDB.

i. Create a collection by name Customers with the following attributes.

   Cust_id, Acc_Bal, Acc_Type

ii. Insert at least 5 values into the table.

iii. Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.

iv. Determine Minimum and Maximum account balance for each customer_id.

v. Export the created collection into local file system.

vi. Drop the table.

vii. Import a given csv dataset from local file system into mongodb collection

Create Database

```
test> db.createCollection("customer");
{ ok: 1 }
```

Create table

```
test> db.customer.insert({"Cust_id":1,"Acc_Bal":5000,"Acc_Type":"Savings"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faa69c75a2a4b111e2626') }
}
test> db.customer.insert({"Cust_id":2,"Acc_Bal":7500,"Acc_Type":"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faa75c75a2a4b111e2627') }
}
test> db.customer.insert({"Cust_id":3,"Acc_Bal":200,"Acc_Type":"Transactions"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faaa4c75a2a4b111e2628') }
}
test> db.customer.insert({"Cust_id":4,"Acc_Bal":10000,"Acc_Type":"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faabac75a2a4b111e2629') }
}
test> db.customer.insert({"Cust_id":5,"Acc_Bal":1000,"Acc_Type":"Transactions"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('693faacbc75a2a4b111e262a') }
}
```

Queries

Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.

```
Atlas atlas-3acvm6-shard-0 [primary] DBME_DEMO> db.Customers.find({ Acc_Bal: { $gt: 1200 }, Acc_Type:
"Z" })
[
  {
    _id: ObjectId('693f8376701108f7231e262c'),
    Cust_id: 1,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('693f8376701108f7231e262e'),
    Cust_id: 3,
    Acc_Bal: 2000,
    Acc_Type: 'Z'
  },
  {
    _id: ObjectId('693f8376701108f7231e2630'),
    Cust_id: 5,
    Acc_Bal: 1800,
    Acc_Type: 'Z'
  }
]
```

Determine Minimum and Maximum account balance for each customer_id.

```
Atlas atlas-3acvm6-shard-0 [primary] DBME_DEMO> db.Customers.aggregate([
...   {
...     $group: {
...       _id: "$Cust_id",
...       Min_Bal: { $min: "$Acc_Bal" },
...       Max_Bal: { $max: "$Acc_Bal" }
...     }
...   }
... ])
...
[
  { _id: 5, Min_Bal: 1800, Max_Bal: 1800 },
  { _id: 2, Min_Bal: 800, Max_Bal: 800 },
  { _id: 4, Min_Bal: 500, Max_Bal: 500 },
  { _id: 3, Min_Bal: 2000, Max_Bal: 2000 },
  { _id: 1, Min_Bal: 1500, Max_Bal: 1500 }
]
```

Export the created collection into local file system

```
C:\Users\BMSCECSE-L3-27>mongoexport --uri="mongodb+srv://annapurna_cs24_user:anusm060706@cluster0.eavk
tnp.mongodb.net/DBME_DEMO" --collection=New_Customer --type=csv --fields=Cust_id,Acc_Bal,Acc_Type --ou
t="C:\Users\BMSCECSE-L3-27\Downloads\New_Customer_Export.csv"
2025-12-15T09:35:30.774+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.eavktnp.mongodb.ne
t/DBME_DEMO
2025-12-15T09:35:30.856+0530    exported 5 records
```

Drop the table

```
Atlas atlas-3acvm6-shard-0 [primary] DBME_DEMO> db.Customers.drop()
true
```

Import a given csv dataset from local file system into mongodb collection

```
C:\Users\BMSCECSE-L3-27>mongoimport --uri="mongodb+srv://annapurna_cs24_user:anusm060706@cluster0.eavk
tnp.mongodb.net/DBME_DEMO" --collection=New_Customer --type=csv --headerline --file="C:\Users\BMSCECSE
-L3-27\Downloads\New_Customer_Export.csv"
2025-12-15T09:35:58.160+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.eavktnp.mongodb.ne
t/DBME_DEMO
2025-12-15T09:35:58.234+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

NoSQL Restaurant Database

Question
(Week 10)

Perform the following DB operations using MongoDB.

i. Write NoSQL Queries on “Restaurant” collection.

ii. Write a MongoDB query to display all the documents in the collection restaurants.

iii. Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

iv. Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.

v. Write a MongoDB query to find the average score for each restaurant.

vi. Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

Create Database

```
test> db.createCollection("customer");
{ ok: 1 }
```

Inserting Values

```
test> db.restaurants.insertMany([
... { name: "Meghna Foods", town: "Jayanagar", cuisine: "Indian", score: 8, address: { zipcode: "10001", street: "Jayanagar"
... } },
... { name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode: "10100", street: "MG Road" } },
... { name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar" } },
... { name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" } },
... { name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" }
... } ])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693fa0ac01c1a3091e1e2621'),
    '1': ObjectId('693fa0ac01c1a3091e1e2622'),
    '2': ObjectId('693fa0ac01c1a3091e1e2623'),
    '3': ObjectId('693fa0ac01c1a3091e1e2624'),
    '4': ObjectId('693fa0ac01c1a3091e1e2625')
  }
}
```

Queries

Write a MongoDB query to display all the documents in the collection restaurants.

```
test> db.restaurants.find({})
[
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.

```
test> db.restaurants.find({}).sort({ name: -1 })
[
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2623'),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.

```
test> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })
[
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2621'),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2622'),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2624'),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId('693fa0ac01c1a3091e1e2625'),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
]
test>
```

Write a MongoDB query to find the average score for each restaurant

```
test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'Empire', average_score: 7 },
  { _id: 'WOW Momos', average_score: 5 }
]
test>
```

Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

```
test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
[
  { name: 'Meghna Foods', address: { street: 'Jayanagar' } },
  { name: 'Empire', address: { street: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
test>
```