VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT on

Database Management Systems (23CS3PCDBM)

Submitted by

MANISHA CS (1BM24CS163)

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 **Manisha CS (1BM24CS163),** 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.

Lab faculty Incharge Name Assistant Professor Department of CSE, BMSCE	Dr. Joythi S Nayak Professor & HOD Department of CSE, BMSCE

Index

4	17-10-25	Bank Database week 3	
5	20-10-25	More Queries on Bank Database week 4	
6	31-10-25	Employee Database week 5	
7	05-11-25	More Queries on Employee Database week 6	
8	14-11-25	Supplier Database week 7	
9	14-11-25	More Queries on Supplier Database	
10	11-12-25	NO SQL	

Week 0

Illustrate an example to create a table student, student table will contain the following attributes: stdid, stdname, dob, doj, fee, gender etc.

```
create database student_detail;

use student_detail;
create table student (
stdid int,
stdname char(20),
dob date,
doj date,
fee int,
gender char(1)
);
```

Insert a few records into the student table.

```
insert into student (stdid,stdname,dob,doj,fee,gender)
values(1,'harshitha','2006-11-12','2006-01-05',10000,'F');
insert into student(stdid,stdname,dob,doj,fee,gender)
values(2,'sowkhya','2006-07-18','2006-02-06',15000,'F');
```

Add a column to the student table (that is phone_no).

```
alter table student add phone_no int;
```

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

stdid	stdname	dob	doj	fee	gender	phone_no
1	harshitha	2006-11-12	2006-01-05	10000	F	NULL
2	sowkhya	2006-07-18	2006-02-06	15000	F	NULL

Modify the column name of phone_no to student_no.

```
alter table student rename column phone_no to student_no;
```

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

stdid	stdname	dob	doj	fee	gender	student_no
1	harshitha	2006-11-12	2006-01-05	10000	F	NULL
2	sowkhya	2006-07-18	2006-02-06	15000	F	NULL

To drop a column from the table

```
30 •    alter table student_info drop column gender;
31
```

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

stdid	stdname	dob	doj	fee	student_no
1	harshitha	2006-11-12	2006-01-05	10000	NULL

Rename the table name to student_info

```
28 •    alter table student rename to student_info;
29
```

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

stdid	stdname	dob	doj	fee	student_no
1	harshitha	2006-11-12	2006-01-05	10000	NULL
2	sowkhya	2006-07-18	2006-02-06	15000	NULL

Delete any records from the table.

```
32 •    delete from student_info where stdid=2;
33
```

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

stdid	stdname	dob	doj	fee	student_no
1	harshitha	2006-11-12	2006-01-05	10000	NULL

INSURANCE DATABASE Week-1:

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.

-Enter at least five tuples for each relation

-Display Accident date and location

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

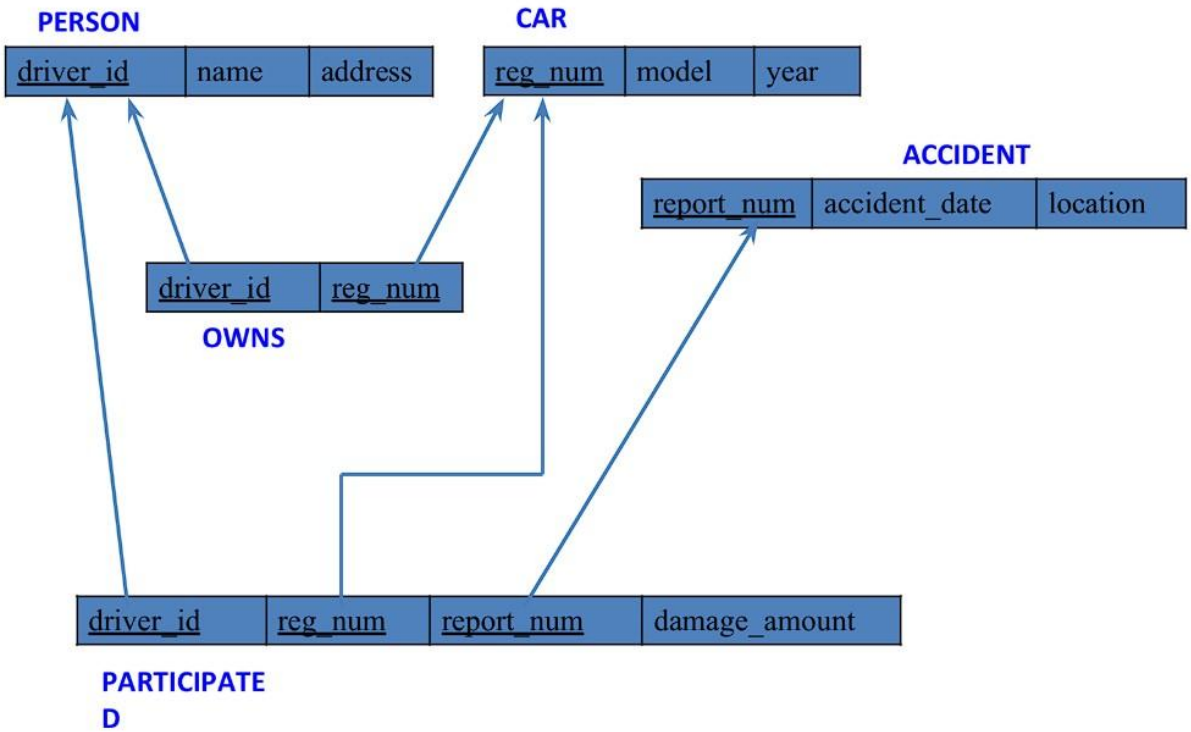
-Add a new accident to the database.

To Do:

-Display Accident date and location

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

Schema Diagram:



Create Database:

Create database insurance;

Use insurance; **Create**

table:

```
Create table person (
Driver_id varchar(20) primary key,
Person_name char(20),
Address char(20)
);
Create table car (
Reg_num varchar(20) primary key,
Model char(20),
Year_of_manufacture int
);
Create table owns (
Driver_id varchar(20),
Reg_num varchar(20),
Foreign key (driver_id) references person (driver_id),
Foreign key (reg_num) references car (reg_num)
);
Create table participated (
Driver_id varchar(20),
Reg_num varchar(20),
Report_num int,
Damage_amount int,
Foreign key (driver_id) references person(driver_id),
```

Foreign key (reg_num) references car (reg_num)

);

Create table accident (

Report_num int,

Accident_date date,

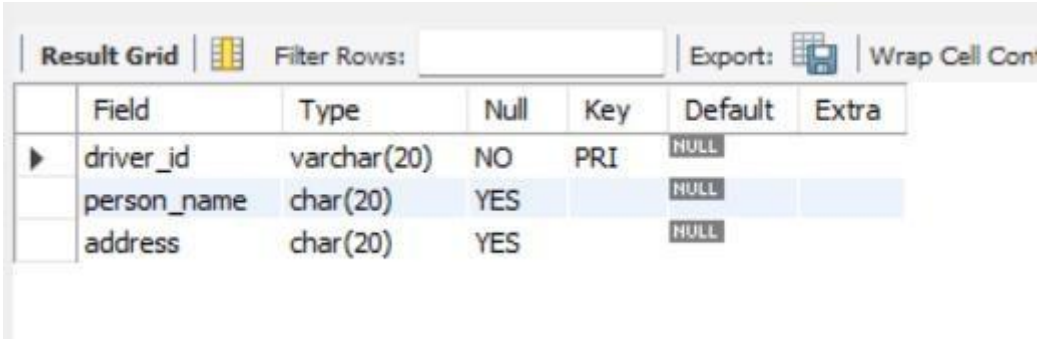
Location char(30),

Foreign key (report_num) references participated (report_num)

);

Structure of table

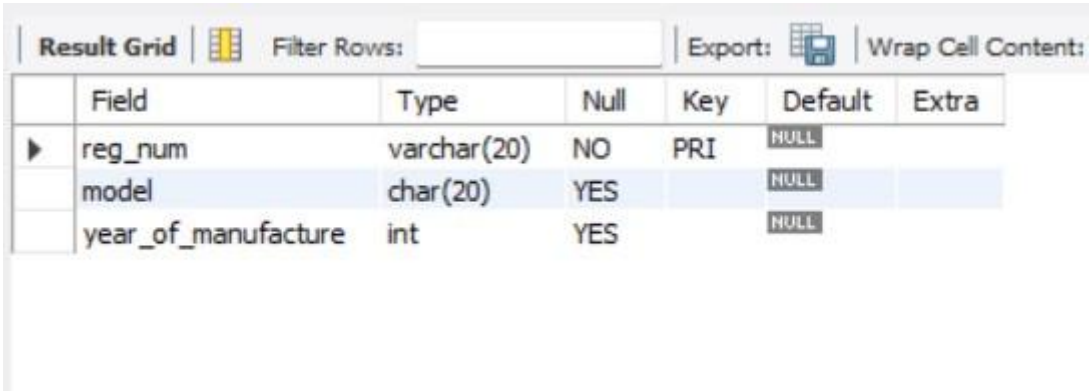
Desc person;



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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
person_name	char(20)	YES		NULL	
address	char(20)	YES		NULL	

Desc car;



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

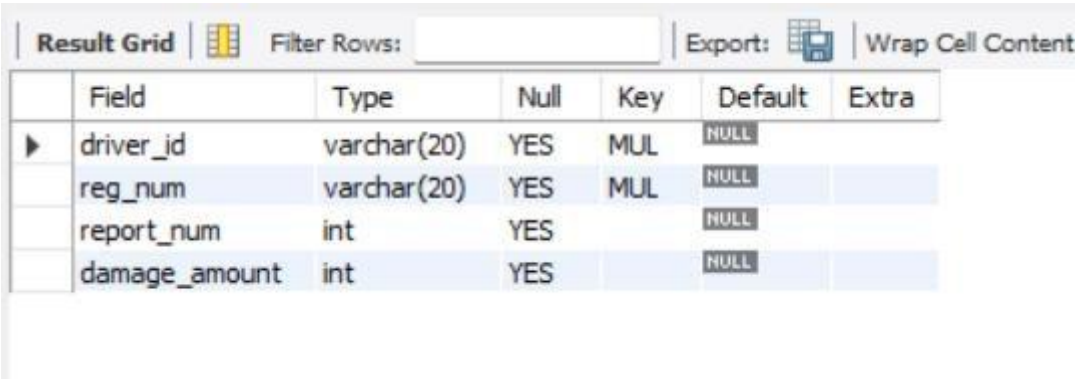
Field	Type	Null	Key	Default	Extra
reg_num	varchar(20)	NO	PRI	NULL	
model	char(20)	YES		NULL	
year_of_manufacture	int	YES		NULL	

Desc owns;

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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	YES	MUL	NULL	
reg_num	varchar(20)	YES	MUL	NULL	

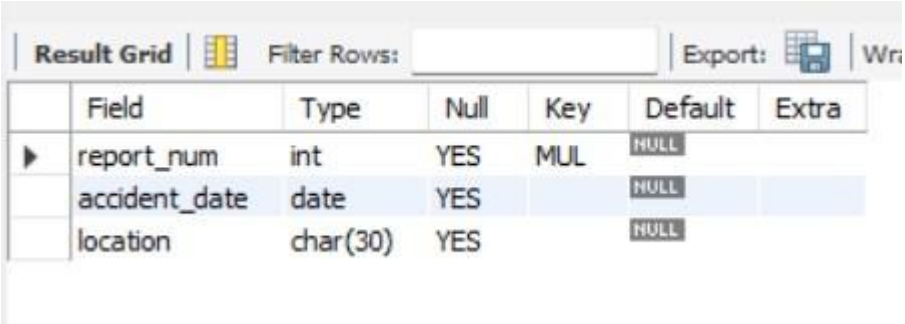
Desc participated;



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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	YES	MUL	NULL	
reg_num	varchar(20)	YES	MUL	NULL	
report_num	int	YES		NULL	
damage_amount	int	YES		NULL	

Desc accident;



Result Grid | Filter Rows: | Export: | Wr

Field	Type	Null	Key	Default	Extra
report_num	int	YES	MUL	NULL	
accident_date	date	YES		NULL	
location	char(30)	YES		NULL	

Inserting Values:

Insert into person(driver_id,person_name,address)

Values('A01','richard','srinivasa nagar');

Insert into person(driver_id,person_name,address)

Values('A02','pradeep','rajaji nagar');

Insert into person(driver_id,person_name,address)

Values('A03','smith','ashok nagar');

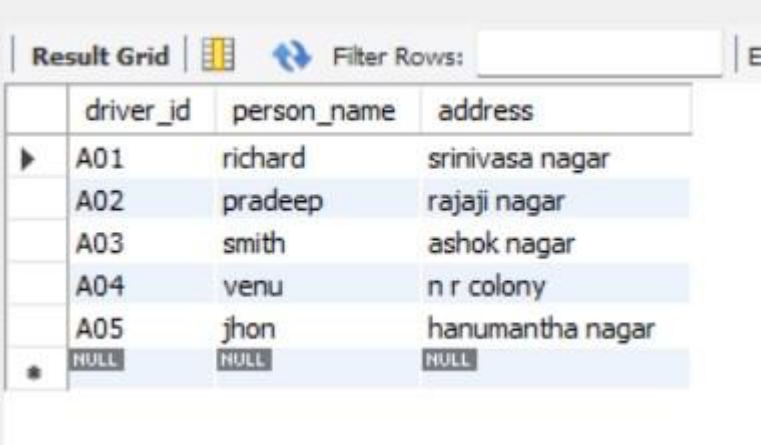
Insert into person(driver_id,person_name,address)

Values('A04','venu','n r colony');

Insert into person(driver_id,person_name,address)

Values('A05','jhon','hanumantha nagar');

Select * from person;



Result Grid | Filter Rows:

driver_id	person_name	address
A01	richard	srinivasa nagar
A02	pradeep	rajaji nagar
A03	smith	ashok nagar
A04	venu	n r colony
A05	jhon	hanumantha nagar
NULL	NULL	NULL

Insert into car (reg_num,model,year_of_manufacture)
Values('KA052250','indica',1990);

Insert into car(reg_num,model,year_of_manufacture)

Values('KA031182','mahendra',1958);

Insert into car(reg_num,model,year_of_manufacture)

Values('KA095477','toyota',1998);

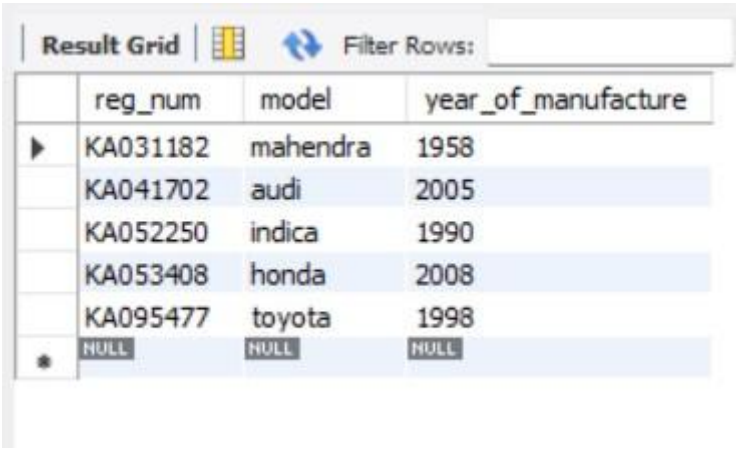
Insert into car(reg_num,model,year_of_manufacture)

Values('KA053408','honda',2008);

Insert into car(reg_num,model,year_of_manufacture)

Values('KA041702','audi',2005);

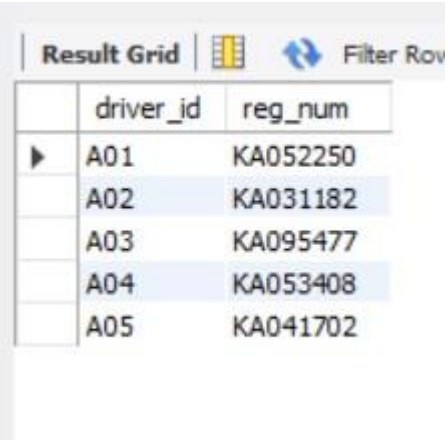
Select * from car;



reg_num	model	year_of_manufacture
KA031182	mahendra	1958
KA041702	audi	2005
KA052250	indica	1990
KA053408	honda	2008
KA095477	toyota	1998
NULL	NULL	NULL

```
Insert into owns(driver_id,reg_num)
Values('A01','KA052250');
Insert into owns(driver_id,reg_num)
Values('A02','KA031182');
Insert into owns(driver_id,reg_num)
Values('A03','KA095477');
Insert into owns(driver_id,reg_num)
Values('A04','KA053408');
Insert into owns(driver_id,reg_num)
Values('A05','KA041702');
```

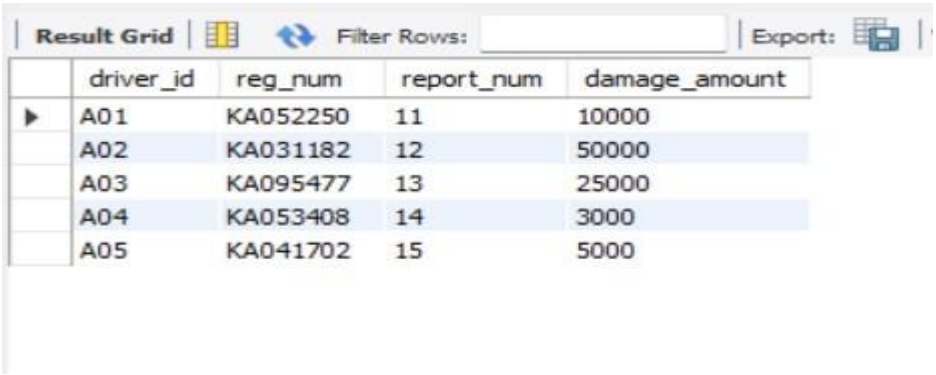
```
Select * from owns;
```



driver_id	reg_num
A01	KA052250
A02	KA031182
A03	KA095477
A04	KA053408
A05	KA041702

```
Insert into participated(driver_id,reg_num,report_num,damage_amount)
Values('A01','KA052250',11,10000);
```

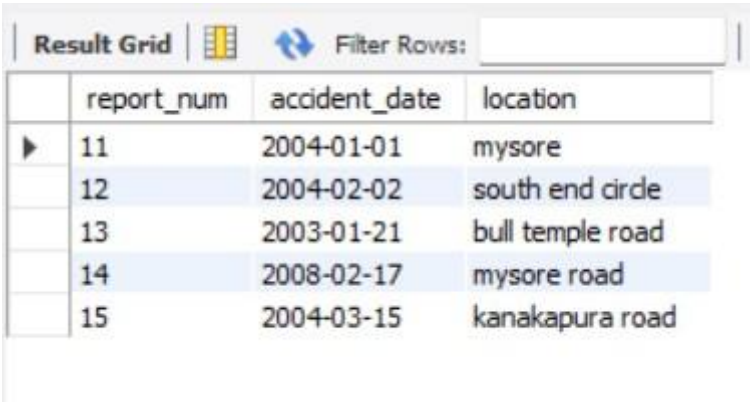
```
Insert into participated(driver_id,reg_num,report_num,damage_amount)
Values('A02','KA031182',12,50000);
Insert into participated(driver_id,reg_num,report_num,damage_amount)
Values('A03','KA095477',13,25000);
Insert into participated(driver_id,reg_num,report_num,damage_amount)
Values('A04','KA053408',14,3000);
Insert into participated(driver_id,reg_num,report_num,damage_amount)
Values('A05','KA041702',15,5000);

Select * from participated;
```


Result Grid | Filter Rows: | Export:

driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA031182	12	50000
A03	KA095477	13	25000
A04	KA053408	14	3000
A05	KA041702	15	5000

```
Insert into accident (report_num,accident_date,location)
Values(11,'2004-01-01','mysore');
Insert into accident (report_num,accident_date,location)
Values(12,'2004-02-02','south end circle');
Insert into accident (report_num,accident_date,location)
Values(13,'2003-01-21','bull temple road');
Insert into accident (report_num,accident_date,location)
Values(14,'2008-02-17','mysore road');
Insert into accident (report_num,accident_date,location)
Values (15,'2004-03-15','kanakapura road');
```

Select * from accident;



Result Grid | Filter Rows:

report_num	accident_date	location
11	2004-01-01	mysore
12	2004-02-02	south end circle
13	2003-01-21	bull temple road
14	2008-02-17	mysore road
15	2004-03-15	kanakapura road

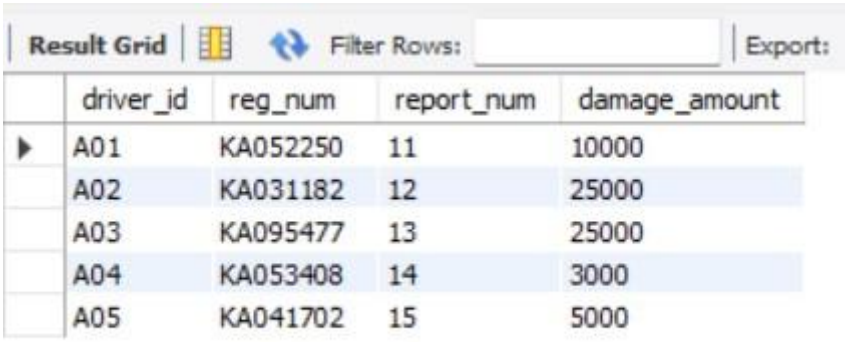
Queries

- **Update the damage amount to 25000 for the car with a specific reg-num (example KA031182) for which the accident report number was 12.**

Update participated set damage_amount=25000

Where reg_num='KA031182' and report_num=12;

Select * from participate



Result Grid | Filter Rows: | Export:

driver_id	reg_num	report_num	damage_amount
A01	KA052250	11	10000
A02	KA031182	12	25000
A03	KA095477	13	25000
A04	KA053408	14	3000
A05	KA041702	15	5000

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

Select count(distinct driver_id) CNT

From participated a, accident b

Where a.report_num=b.report_num and b.accident_date like '2008%';

Result Grid | Filter Rows: | Export:

	CNT
▶	1

-Add a new accident to the database.

Insert into accident values(15,'2008-03-08','Domlur');

Result Grid | Filter Rows:

	report_num	accident_date	location
▶	11	2004-01-01	mysore
	12	2004-02-02	south end circle
	13	2003-01-21	bull temple road
	14	2008-02-17	mysore road
	15	2004-03-15	kanakapura road
	15	2008-03-08	Domlur

-Display Accident date and location

Select accident_date,location from accident;

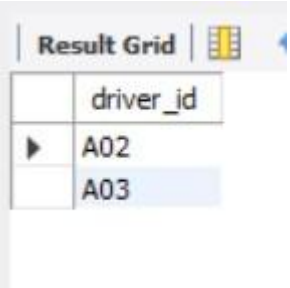
Result Grid | Filter Rows:

	accident_date	location
▶	2004-01-01	mysore
	2004-02-02	south end circle
	2003-01-21	bull temple road
	2008-02-17	mysore road
	2004-03-15	kanakapura road
	2008-03-08	Domlur

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

Select distinct(driver_id) from participated

Where damage_amount >= 25000;



Week-2:

More queries on insurance database

List of operations

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

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

-Find the number of accidents in which cars belonging to a specific model (ex: 'toyota')were involved accidents in 2008

Additional queries:

- list the entire participated relation in the descending order of damage amount.

-find the average damage amount

-delete the tuple from participated relation whose damage amount is below the average damage amount

-list the name of drivers whose damage is greater than the average damage amount.

-find maximum damage amount.

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

Select * from car order by year_of_manufacture asc;

Result Grid | Filter Rows:

reg_num	model	year_of_manufacture
KA031182	mahendra	1958
KA052250	indica	1990
KA095477	toyota	1998
KA041702	audi	2005
KA053408	honda	2008
NULL	NULL	NULL

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

Select count(report_num) CNT from car c,participated p where c.reg_num=p.reg_num and Model="toyota";

Result Grid

CNT
1

3. Find the number of accidents in which cars belonging to a specific model (ex: 'toyota')were involved accidents in 2008

Select count(distinct driver_id) CNT from participated a, accident b

Where a.report_num=b.report_num and b.accident_date like '2008%';

Result Grid

CNT
2

4. 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

driver_id	reg_num	report_num	damage_amount
A02	KA031182	12	25000
A03	KA095477	13	25000
A01	KA052250	11	10000
A05	KA041702	15	5000
A04	KA053408	14	3000

5. find the average damage amount

Select avg(damage_amount) from participated;

AVG(DAMAGE_AMOUNT)
13600.0000

6. list the name of drivers whose damage is greater than the average damage amount.

Select person_name from person a, participated b where a.driver_id = b.driver_id and damage_amount>(select avg(damage_amount) from participated);

Result Grid

PERSON_NAME
pradeep
smith

7. find maximum damage amount.

Select max(damage_amount) from participated;

Result Grid | Filter Row

MAX(DAMAGE_AMOUNT)
25000

8. delete the tuple from participated relation whose damage amount is below the average damage amount

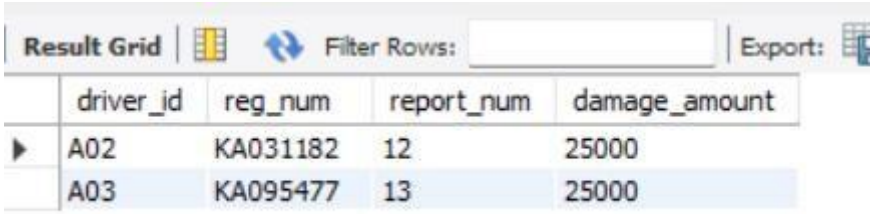
Delete from participated

Where damage_amount <

(select avg(damage_amount)

From (select damage_amount from participated) as t);

Select *from participated;



Result Grid | Filter Rows: | Export:

driver_id	reg_num	report_num	damage_amount
A02	KA031182	12	25000
A03	KA095477	13	25000

BANK DATABASE

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)

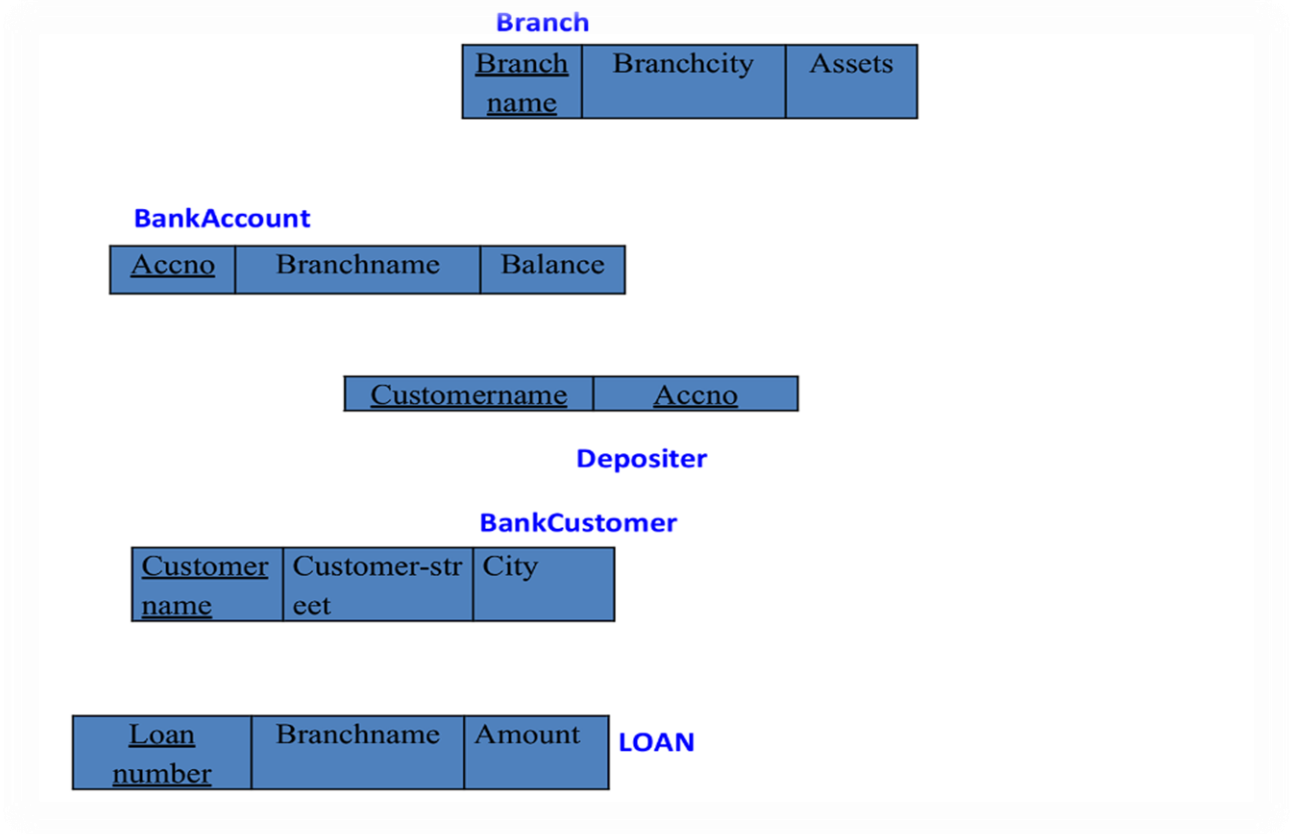
To do:

- Create the above tables by properly specifying the primary keys and the foreign keys.
- Enter at least five tuples for each relation.
- Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
- Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).
- Create a view which gives each branch the sum of the amount of all the loans at the branch.

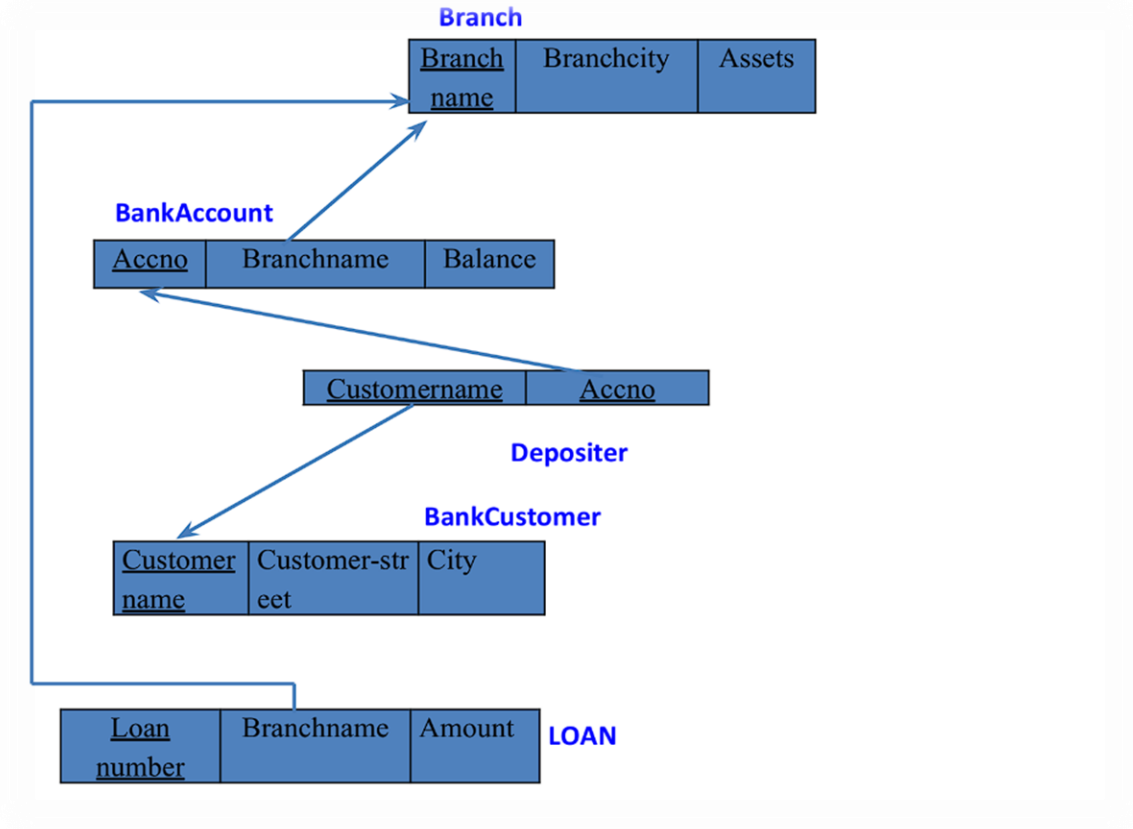
Spot Query:

- Update or add rupees 1000 to account balance for the customers who are residing in Bangalore.

Relational Schema Diagram:



Schema Diagram:



Create database

create database Bank; use

bank;

Create table

create table Branch(

branch_name varchar(30),

branch_city varchar(25),

assets int,

primary key(branch_name));

create table BankAccount(acc_no int, branch_name

varchar(30), balance int, primary key(acc_no), foreign

key(branch_name)references Branch(branch_name));

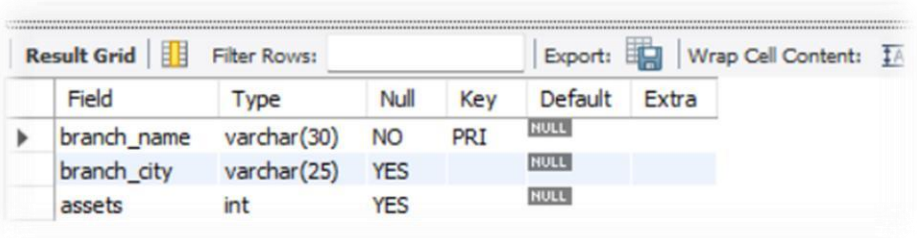
create table BankCustomer(

customer_name varchar(20),

customer_street varchar(30),

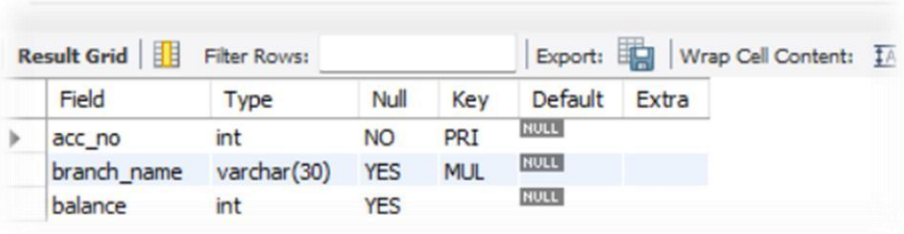
```
customer_city varchar(35),
primary key(customer_name));

create table Depositer(
customer_name varchar(20),
acc_no int, primary key(customer_name, acc_no),
foreign key(acc_no)references BankAccount
(acc_no),
foreign key(customer_name)references BankCustomer(customer_name));

create table Loan( loan_number int, branch_name
varchar(30), amount int, primary key(loan_number), foreign
key(branch_name)references Branch(branch_name));
```

Structure of the table desc Branch;



Field	Type	Null	Key	Default	Extra
branch_name	varchar(30)	NO	PRI	NULL	
branch_city	varchar(25)	YES		NULL	
assets	int	YES		NULL	

desc BankAccount;



Field	Type	Null	Key	Default	Extra
acc_no	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
balance	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(35)	YES		NULL	

desc Depositer;

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

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
acc_no	int	NO	PRI	NULL	

desc Loan;

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

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

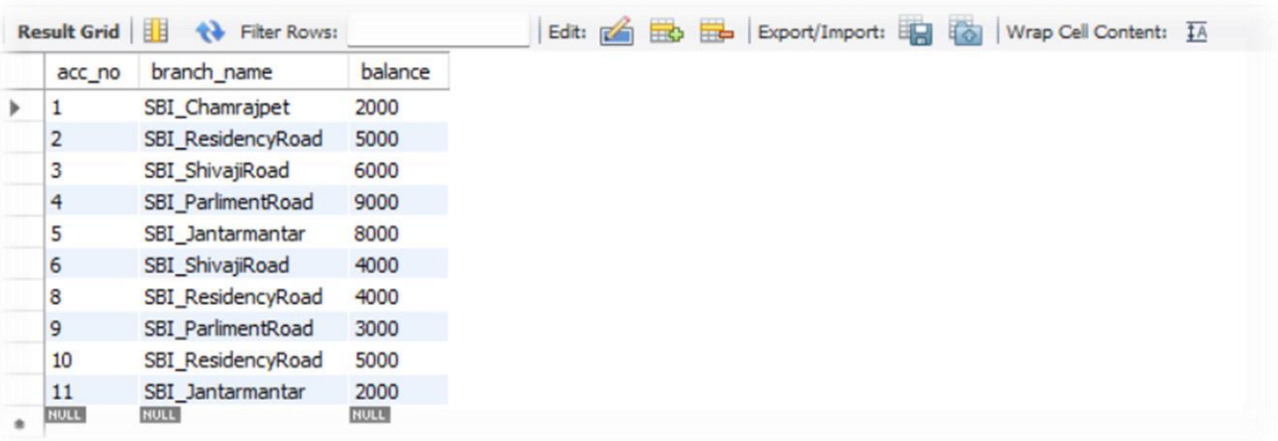
Inserting Values to the table insert into Branch values("SBI_chamrajpet", "Bangalore", 50000); insert into Branch values("SBI_ResidencyRoad", "Bangalore", 10000); insert into Branch values("SBI_ShivajiRoad", "Bombay", 20000); insert into Branch values("SBI_ParlimentRoad", "Delhi", 10000); insert into Branch values("SBI_Jantarmantar", "Delhi", 20000); select * from Branch;

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

branch_name	branch_city	assets
SBI_chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

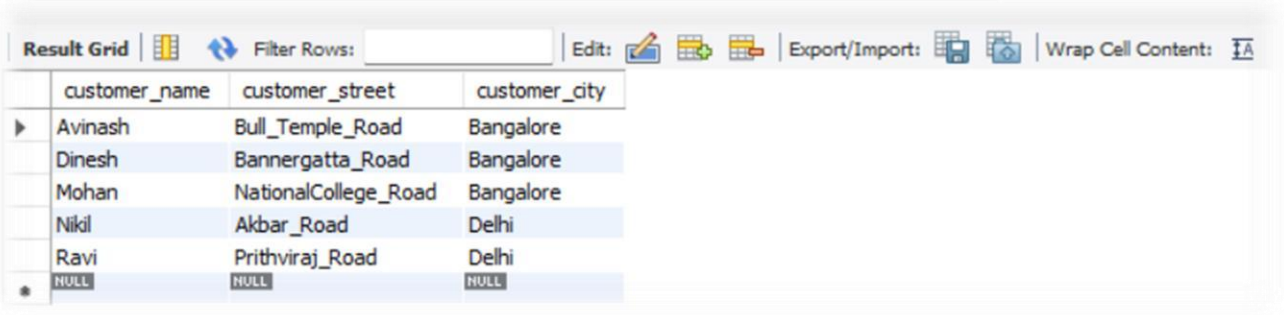
insert into BankAccount values(1, "SBI_Chamrajpet", 2000); insert into BankAccount values(2, "SBI_ResidencyRoad", 5000); insert into BankAccount values(3, "SBI_ShivajiRoad", 6000); insert into BankAccount values(4, "SBI_ParlimentRoad", 9000); insert into

BankAccount values(5, "SBI_Jantarmantar", 8000); insert into
BankAccount values(6, "SBI_ShivajiRoad", 4000); insert into
BankAccount values(8, "SBI_ResidencyRoad", 4000); insert into
BankAccount values(9, "SBI_ParlimentRoad", 3000); insert into
BankAccount values(10, "SBI_ResidencyRoad", 5000); insert into
BankAccount values(11, "SBI_Jantarmantar", 2000); select * from
BankAccount;



acc_no	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
NULL	NULL	NULL

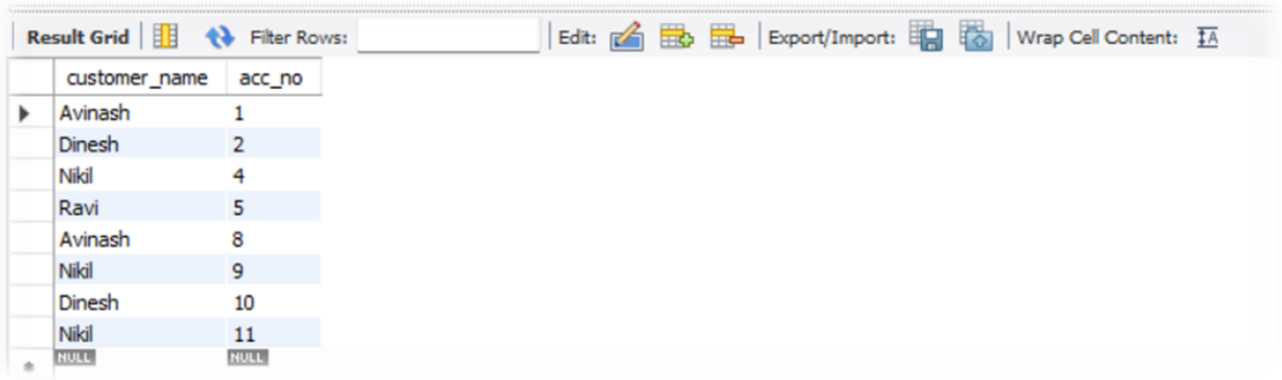
insert into BankCustomer values("Avinash", "Bull_Temple_Road", "Bangalore");
insert into BankCustomer values("Dinesh", "Bannergatta_Road", "Bangalore");
insert into BankCustomer values("Mohan", "NationalCollege_Road",
"Bangalore"); insert into BankCustomer values("Nikil", "Akbar_Road", "Delhi");
insert into BankCustomer values("Ravi", "Prithviraj_Road", "Delhi"); select * from
BankCustomer;



customer_name	customer_street	customer_city
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

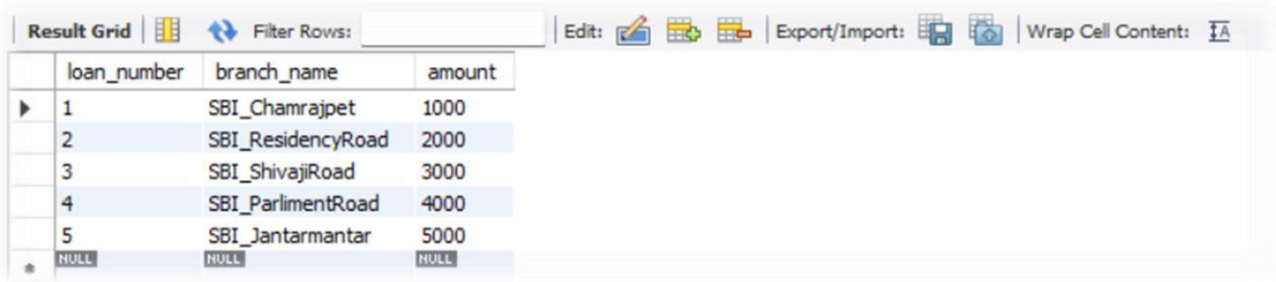
insert into Depositer values("Avinash", 1);
insert into Depositer values("Dinesh", 2);
insert into Depositer values("Nikil", 4);

```
insert into Depositer values("Ravi", 5);
insert into Depositer values("Avinash", 8);
insert into Depositer values("Nikil", 9);
insert into Depositer values("Dinesh", 10);
insert into Depositer values("Nikil", 11);
select * from Depositer;
```



customer_name	acc_no
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
NULL	NULL

```
insert into Loan values(1, "SBI_Chamrajpet", 1000); insert
into Loan values(2, "SBI_ResidencyRoad", 2000); insert
into Loan values(3, "SBI_ShivajiRoad", 3000); insert into
Loan values(4, "SBI_ParlimentRoad", 4000); insert into
Loan values(5, "SBI_Jantarmantar", 5000); select * from
Loan;
```

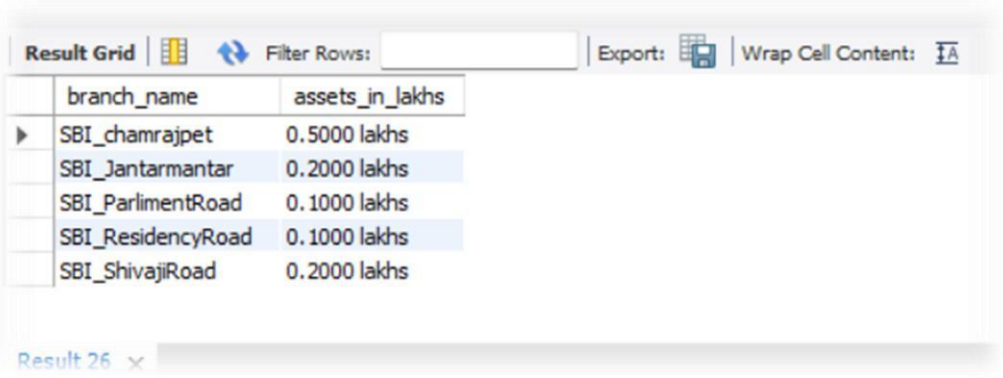


loan_number	branch_name	amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParlimentRoad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

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;
```



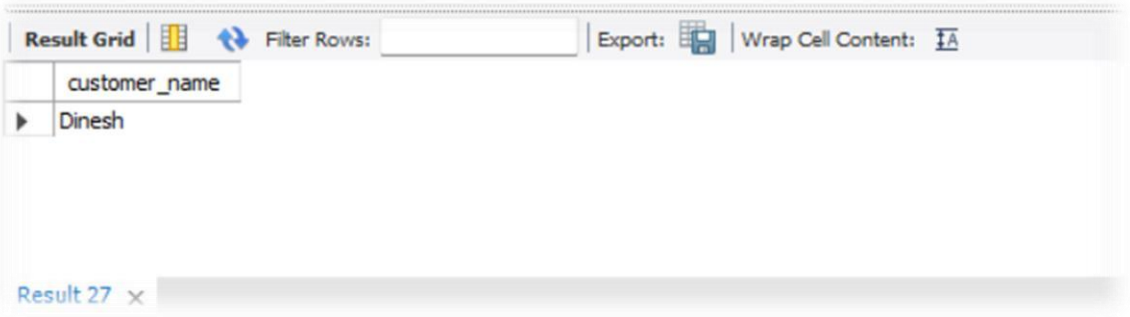
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

branch_name	assets_in_lakhs
SBI_chamrajpet	0.5000 lakhs
SBI_Jantarmantar	0.2000 lakhs
SBI_ParlimentRoad	0.1000 lakhs
SBI_ResidencyRoad	0.1000 lakhs
SBI_ShivajiRoad	0.2000 lakhs

Result 26

- **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.acc_no = b.acc_no group by d.customer_name having count(d.acc_no)>= 2;

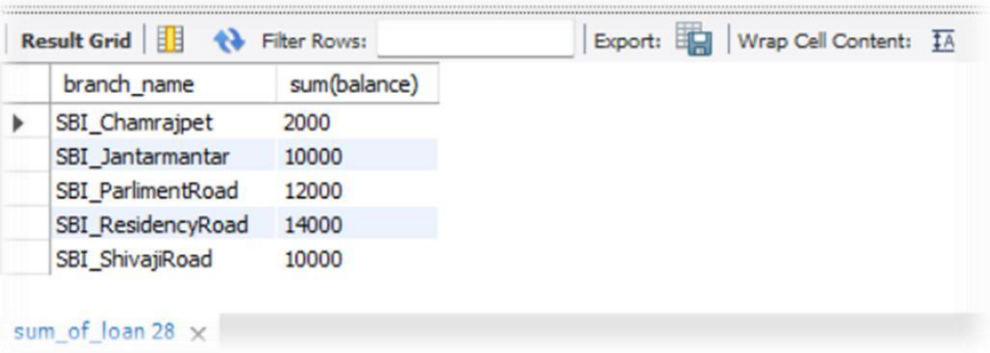


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

customer_name
Dinesh

Result 27

- **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;



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

branch_name	sum(balance)
SBI_Chamrajpet	2000
SBI_Jantarmantar	10000
SBI_ParlimentRoad	12000
SBI_ResidencyRoad	14000
SBI_ShivajiRoad	10000

sum_of_loan 28

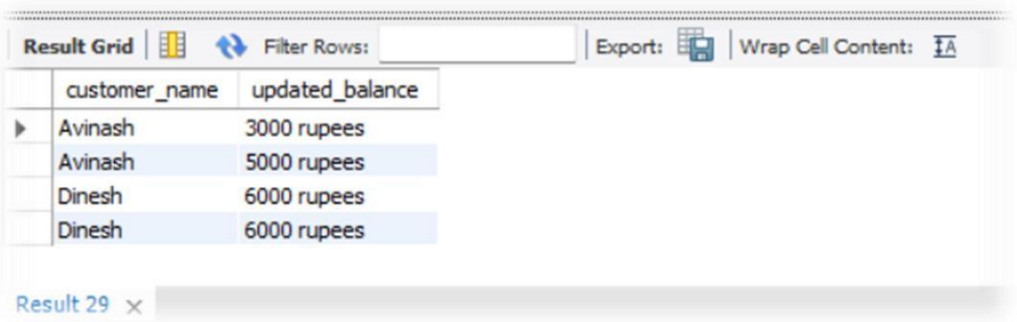
Spot Query

- **Update or add rupees 1000 to account balance for the customers who are residing in Bangalore.**

  select bc.customer_name, concat(balance+1000, ' rupees')

  updated_balance from BankAccount b, BankCustomer bc, Depositer d

where bc.customer_name = d.customer_name and b.acc_no = d.acc_no and bc.customer_city = 'Bangalore';



Week-4:

More Queries on Bank Database

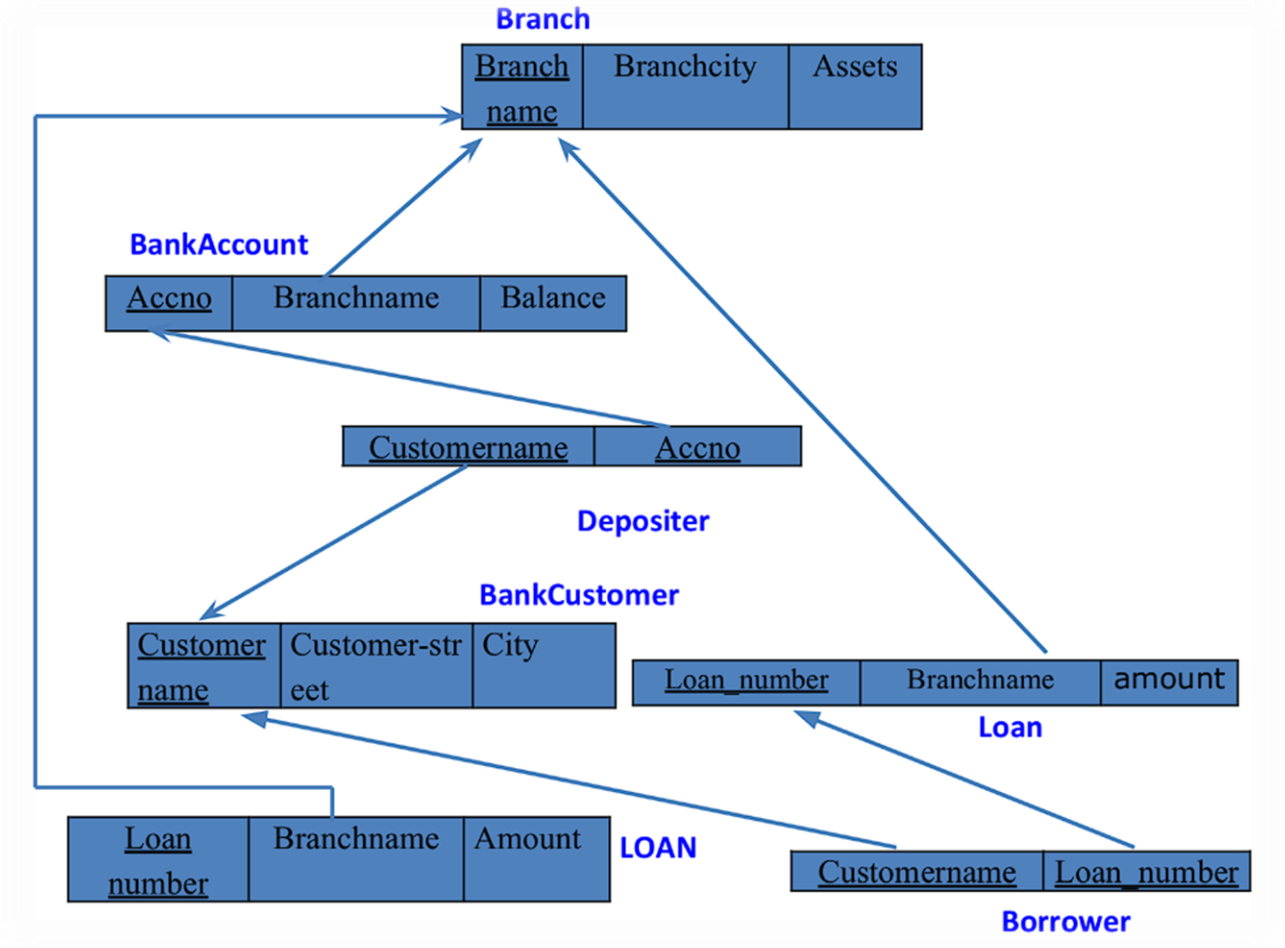
- Branch (branch-name: String, branch-city: String, assets: real)
- BankAccount(accno: int, branch-name: String, balance: real)
- BankCustomer (customer-name: String, customer-street: String,
- customer-city: String)
- Depositer(customer-name: String, accno: int)
- LOAN (loan-number: int, branch-name: String, amount: real)
- Borrower (customer-name: String, loan-number: int)

To do:

- Create the above tables by properly specifying the primary keys and the foreign keys.
- Enter at least five tuples for each relation.
- Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).
- Find all customers who have a loan at the bank but do not have an account.

- Find all customers who have both an account and a loan at the Bangalore branch
- Find the names of all branches that have greater assets than all branches located in Bangalore.
- Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).
- Update the Balance of all accounts by 5% **Additional queries:**
- List the entire loan relation in the descending order of amount.
- Find all customers having a laon, an account or both at the Bank
- Create a view which gives each branch the sum of the amount of All the loans at the branch.
- The annual interest payments are made and all branches are to Be increased by 5%.

Schema Diagram:



Create database

create database Bank; use

bank;

Create table

create table Branch(branch_name

varchar(30), branch_city

varchar(25),

assets int,

primary key(branch_name));

create table

BankAccount(acc_no int,

branch_name varchar(30),

```
balance int, primary key(acc_no), foreign
key(branch_name)references Branch(branch_name));

create table BankCustomer(
customer_name varchar(20),
customer_street varchar(30),
customer_city varchar(35),
primary key(customer_name));

create table Depositer(
customer_name varchar(20),
acc_no int, primary key(customer_name, acc_no), foreign
key(acc_no)references BankAccount (acc_no), foreign
key(customer_name)references BankCustomer(customer_name));

create table Loan( loan_number int, branch_name
varchar(30), amount int, primary key(loan_number), foreign
key(branch_name)references Branch(branch_name));


create table Borrower(
customer_name varchar(20),
loan_number int, primary
key(customer_name, loan_number),
foreign
key(customer_name)references
BankCustomer(customer_name),
foreign key(loan_number)references
Loan(loan_number));
```

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	
branch_city	varchar(25)	YES		NULL	
assets	int	YES		NULL	

desc BankAccount;

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

Field	Type	Null	Key	Default	Extra
acc_no	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
balance	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(35)	YES		NULL	

desc Depositer;

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

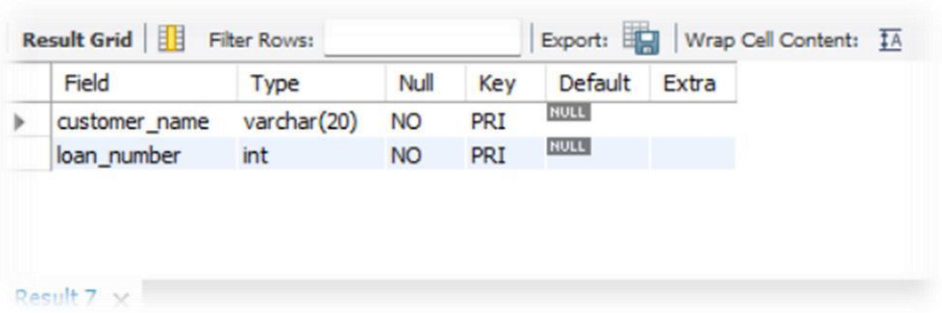
Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
acc_no	int	NO	PRI	NULL	

desc Loan;

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

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

desc Borrower;



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

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
loan_number	int	NO	PRI	NULL	

Result 7

Inserting Values to the table insert into Branch

values("SBI_chamrajpet", "Bangalore", 50000); insert into Branch

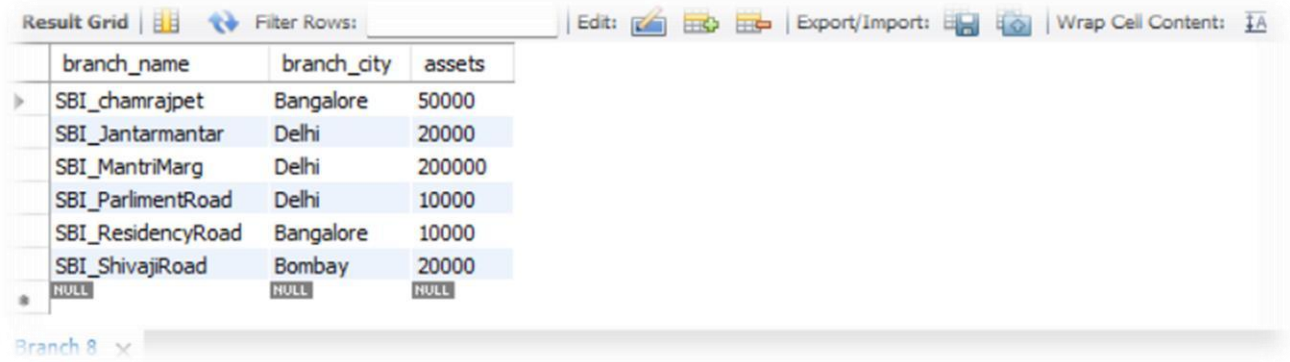
values("SBI_ResidencyRoad", "Bangalore", 10000); insert into Branch

values("SBI_ShivajiRoad", "Bombay", 20000); insert into Branch

values("SBI_ParlimentRoad", "Delhi", 10000); insert into Branch

values("SBI_Jantarmantar", "Delhi", 20000); insert into Branch

values("SBI_MantriMarg", "Delhi", 200000); select * from Branch;



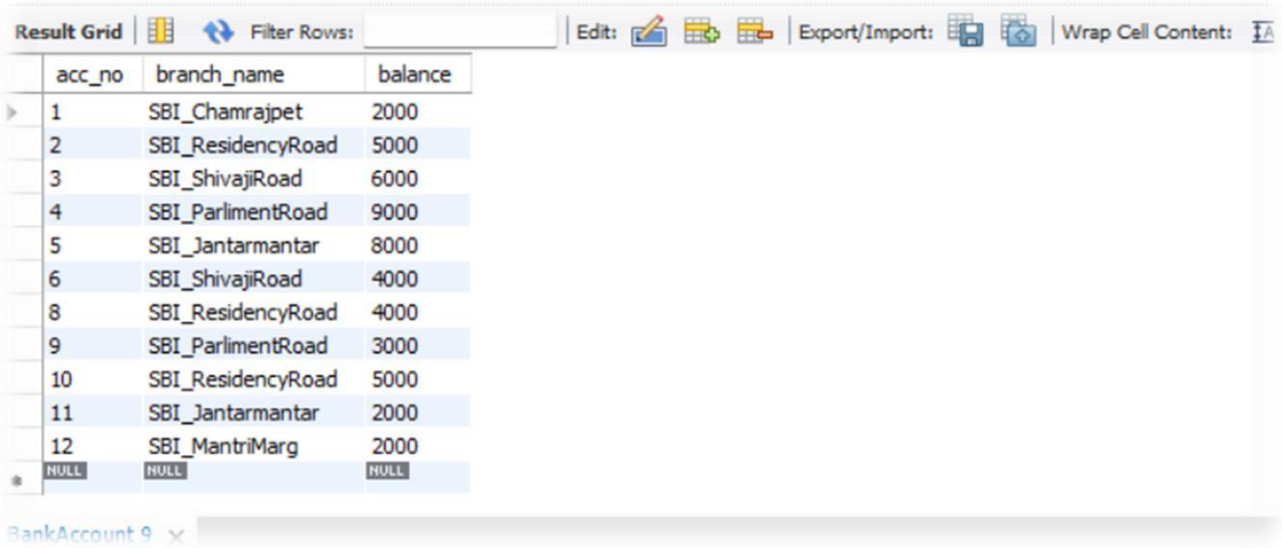
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

branch_name	branch_city	assets
SBI_chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_MantriMarg	Delhi	200000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

Branch 8

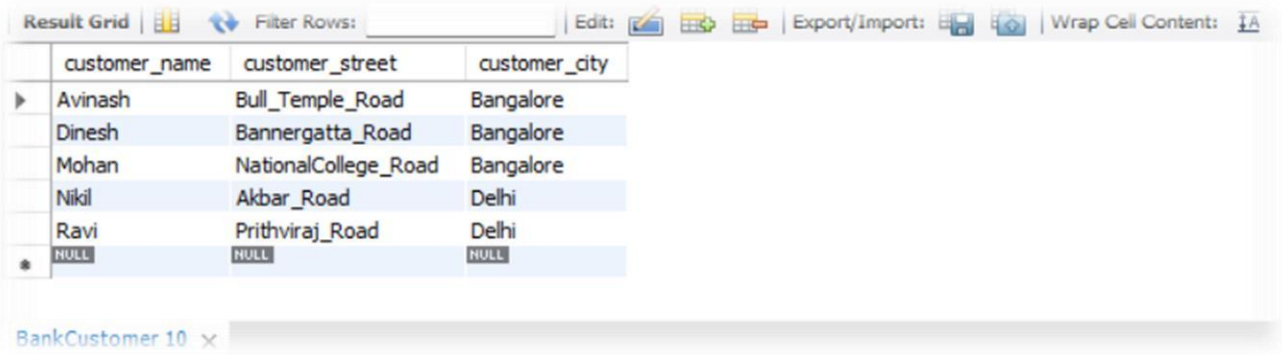
insert into BankAccount values(1, "SBI_Chamrajpet", 2000); insert into BankAccount values(2, "SBI_ResidencyRoad", 5000); insert into BankAccount values(3, "SBI_ShivajiRoad", 6000); insert into BankAccount values(4, "SBI_ParlimentRoad", 9000); insert into BankAccount values(5, "SBI_Jantarmantar", 8000); insert into BankAccount values(6, "SBI_ShivajiRoad", 4000); insert into BankAccount values(8, "SBI_ResidencyRoad", 4000); insert into BankAccount values(9, "SBI_ParlimentRoad", 3000); insert into BankAccount values(10, "SBI_ResidencyRoad", 5000); insert into BankAccount values(11, "SBI_Jantarmantar", 2000); insert into

BankAccount values(12, "SBI_MantriMarg", 2000); select * from BankAccount;



acc_no	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

insert into BankCustomer values("Avinash", "Bull_Temple_Road", "Bangalore");
insert into BankCustomer values("Dinesh", "Bannergatta_Road", "Bangalore");
insert into BankCustomer values("Mohan", "NationalCollege_Road", "Bangalore"); insert into BankCustomer values("Nikil", "Akbar_Road", "Delhi");
insert into BankCustomer values("Ravi", "Prithviraj_Road", "Delhi"); select * from BankCustomer;



customer_name	customer_street	customer_city
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

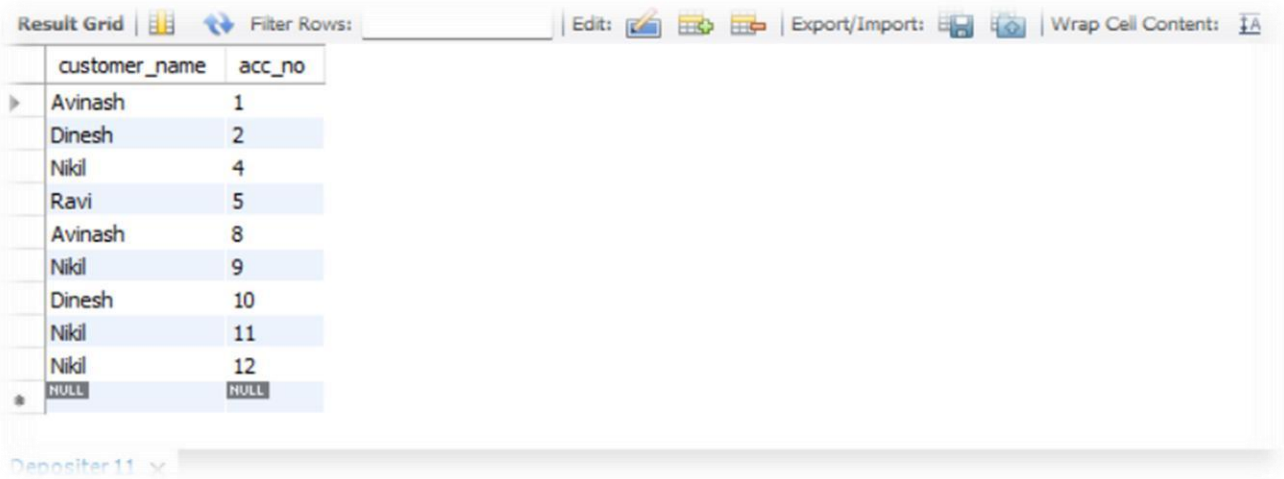
insert into Depositer values("Avinash", 1);
insert into Depositer values("Dinesh", 2);
insert into Depositer values("Nikil", 4);
insert into Depositer values("Ravi", 5);
insert into Depositer values("Avinash", 8);
insert into Depositer values("Nikil", 9);

insert into Depositer values("Dinesh", 10);

insert into Depositer values("Nikil", 11);

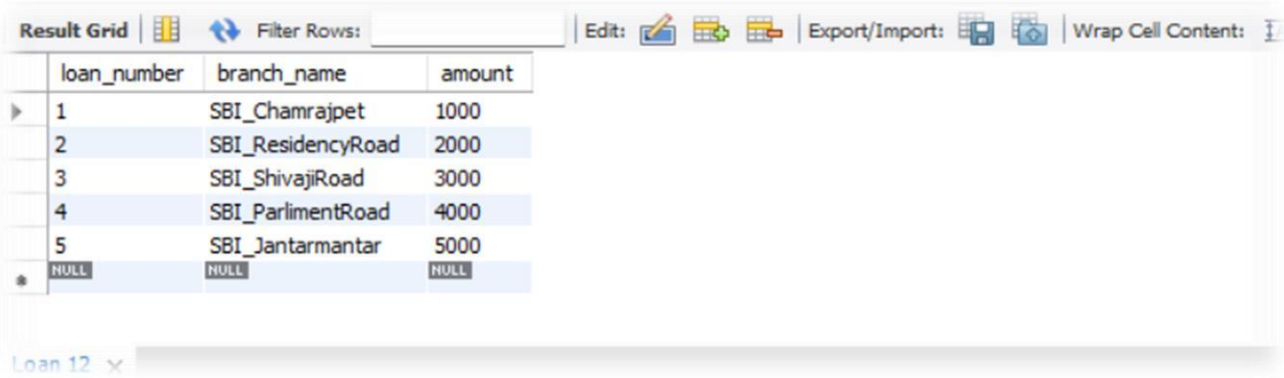
insert into Depositer values("Nikil", 12);

select * from Depositer;



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

insert into Loan values(1, "SBI_Chamrajpet", 1000); insert into Loan values(2, "SBI_ResidencyRoad", 2000); insert into Loan values(3, "SBI_ShivajiRoad", 3000); insert into Loan values(4, "SBI_ParlimentRoad", 4000); insert into Loan values(5, "SBI_Jantarmantar", 5000); select * from Loan;



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

insert into Borrower values("Avinash", 1);

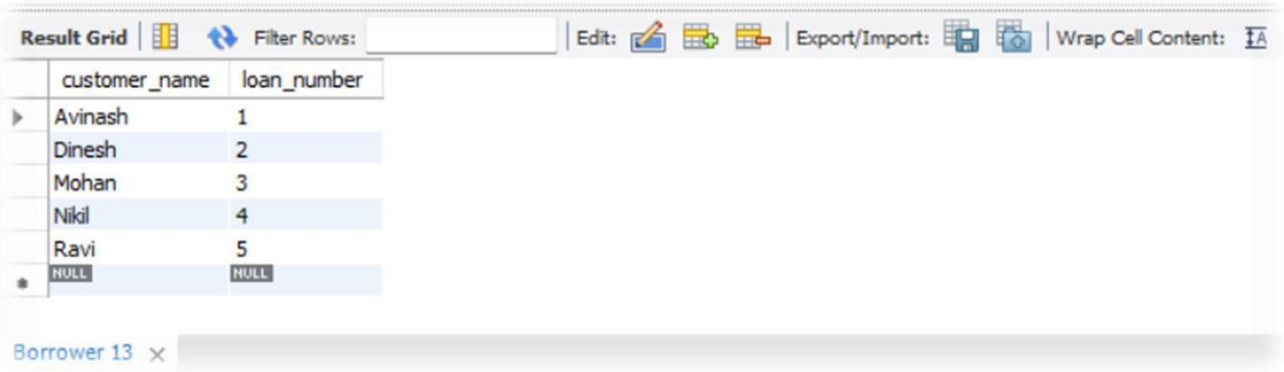
insert into Borrower values("Dinesh", 2);

insert into Borrower values("Mohan", 3);

insert into Borrower values("Nikil", 4);

insert into Borrower values("Ravi", 5);

select * from Borrower;



customer_name	loan_number
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5
NULL	NULL

Borrower 13

Queries

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

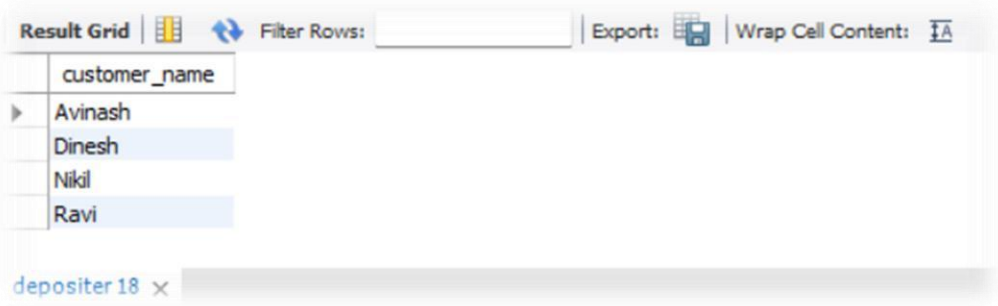
  select distinct S.customer_name from depositer as S

  where not exists(select branch_name from branch where branch_city = 'Dehli' and

  branch_name not in(select R.branch_name from depositer as T, BankAccount as R where

  T.acc_no = R.acc_no

  and S.customer_name = T.customer_name));



customer_name
Avinash
Dinesh
Nikil
Ravi

depositer 18

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

  select distinct customer_name from

  borrower where customer_name not in

  (select customer_name from depositer);

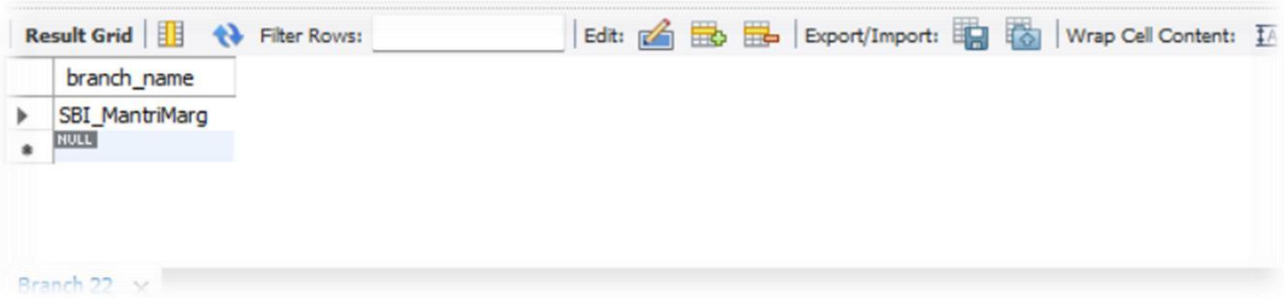
customer_name
Mohan

borrower 20

- **Find all customers who have both an account and a loan at the Bangalore branch** select customer_name from Borrower, Loan where Borrower.loan_number = Loan.loan_number and Loan.branch_name in(select branch_name from Depositer, BankAccount where Depositer.acc_no = BankAccount.branch_name in (select branch_name from Branch where Branch.branch_city = "Bangalore"));

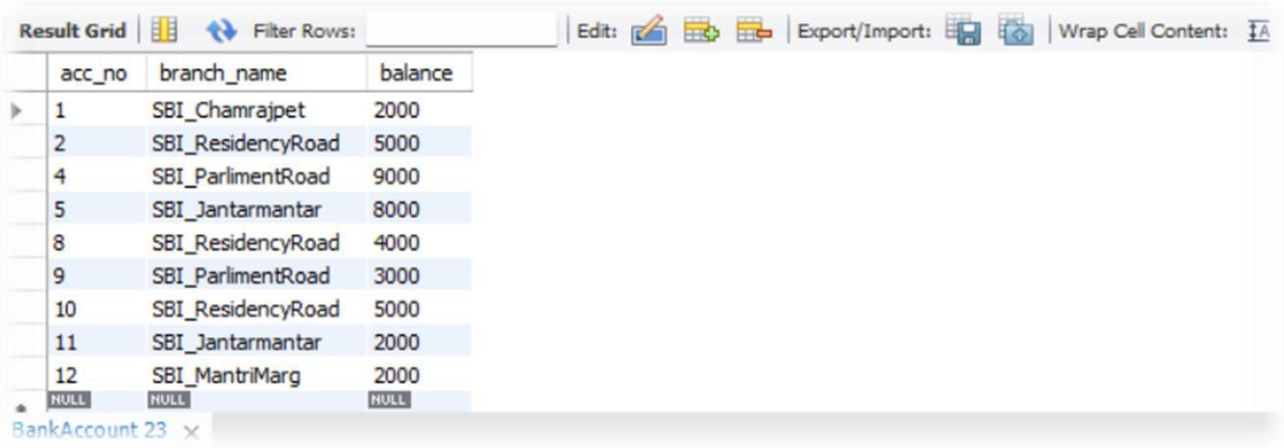
- **Find the names of all branches that have greater assets than all branches located in Bangalore.**

  select branch_name from Branch where assets > all (select assets from Branch where branch_city = 'Bangalore');



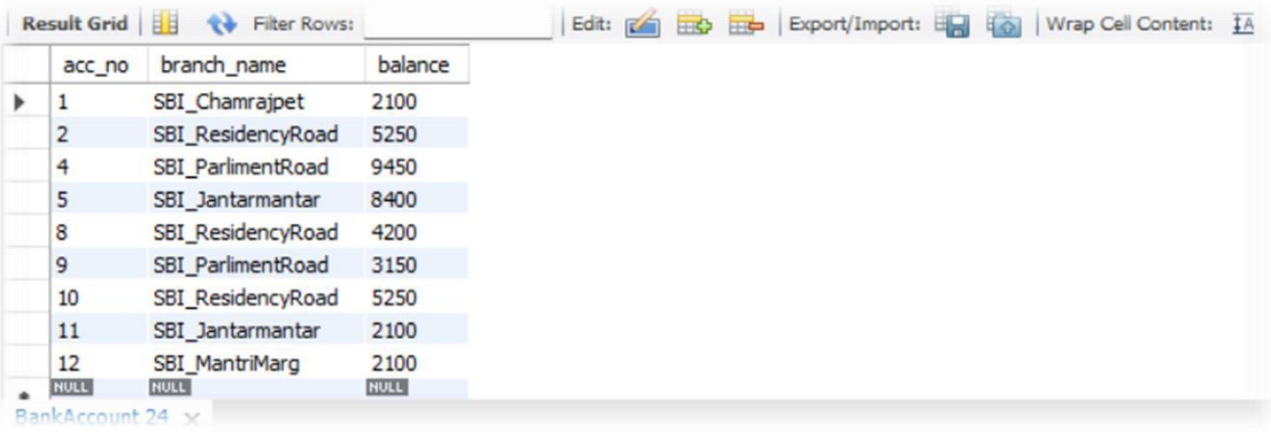
- **Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).**

  delete from BankAccount
  where branch_name in(select branch_name from Branch where branch_city = 'Bombay');
  select * from BankAccount;



acc_no	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

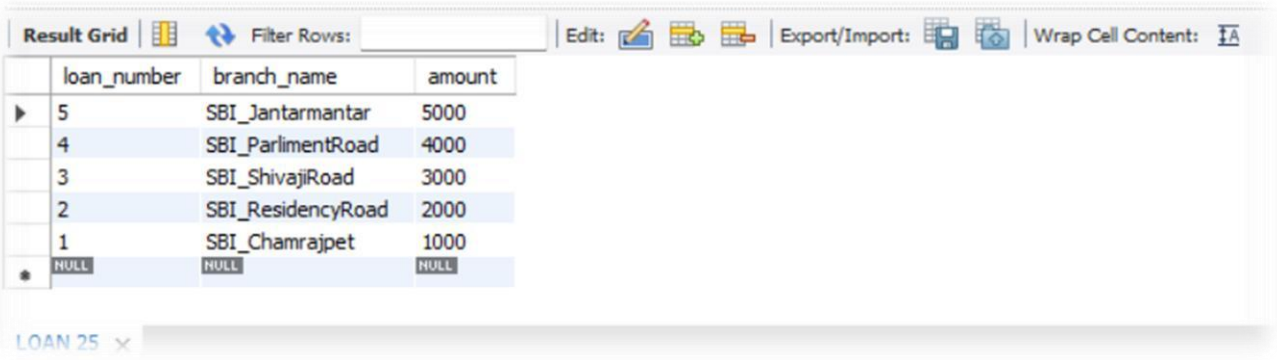
- **Update the Balance of all accounts by 5%** set sql_safe_updates = 0;
  update BankAccount set balance = balance + balance *0.05; select
  * from BankAccount;



acc_no	branch_name	balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
8	SBI_ResidencyRoad	4200
9	SBI_ParlimentRoad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
12	SBI_MantriMarg	2100
NULL	NULL	NULL

Additional queries

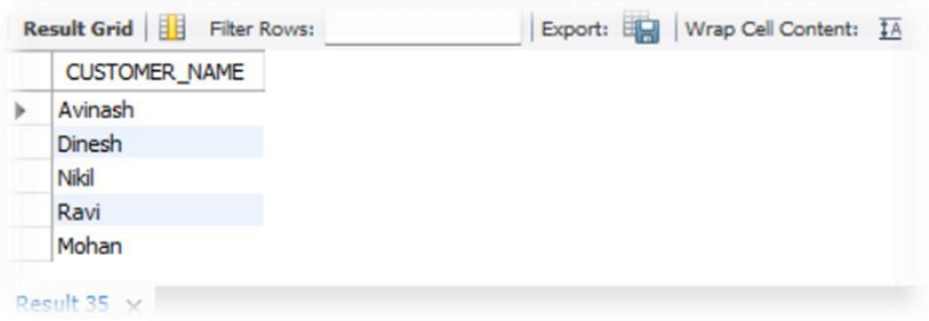
- **List the entire loan relation in the descending order of amount.**
  select * from loan order by amount desc;



loan_number	branch_name	amount
5	SBI_Jantarmantar	5000
4	SBI_ParlimentRoad	4000
3	SBI_ShivajiRoad	3000
2	SBI_ResidencyRoad	2000
1	SBI_Chamrajpet	1000
NULL	NULL	NULL

- **Find all customers having a laon, an account or both at the Bank**

  (select customer_name from depositer) union (select customer_name from borrower);



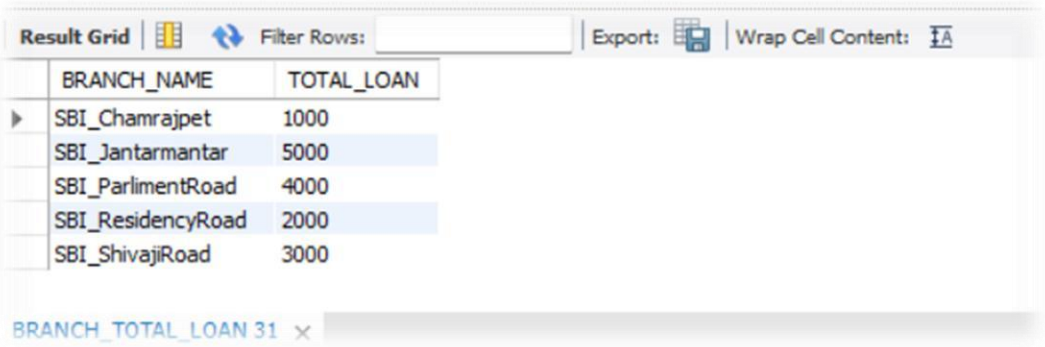
CUSTOMER_NAME
Avinash
Dinesh
Nikil
Ravi
Mohan

- **Create a view which gives each branch the sum of the amount of All the loans at the branch.**

  create view branch_total_loan (branch_name,

  total_loan) as select branch_name, sum(amount) from loan

  group by branch_name; select * from branch_total_loan;

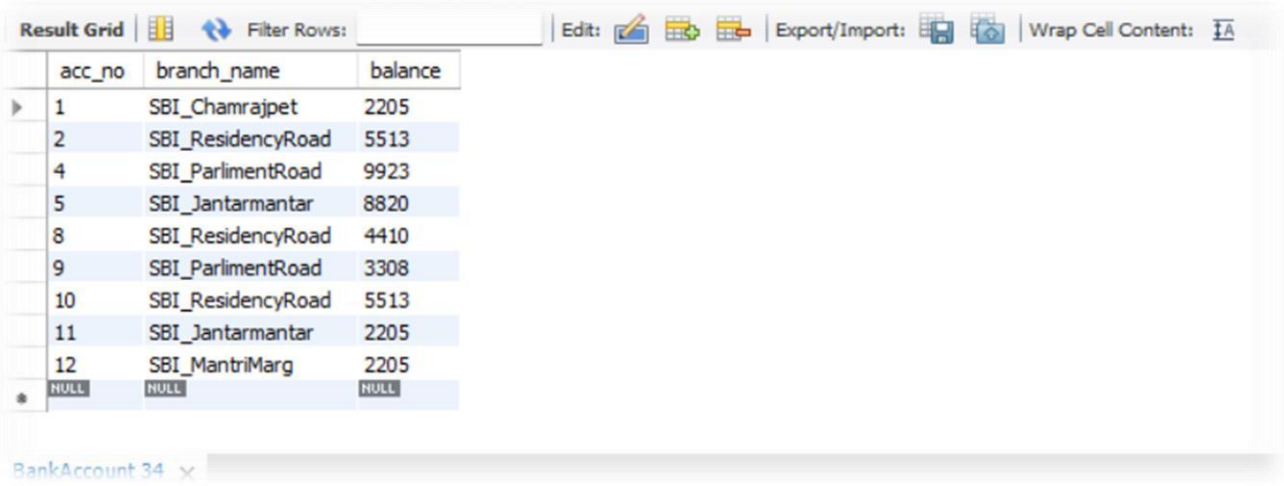


BRANCH_NAME	TOTAL_LOAN
SBI_Chamrajpet	1000
SBI_Jantarmantar	5000
SBI_ParlimentRoad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

- **The annual interest payments are made and all branches are to Be increased by 5%.**

  update bankaccount set balance = balance * 1.05;

  select * from bankaccount;



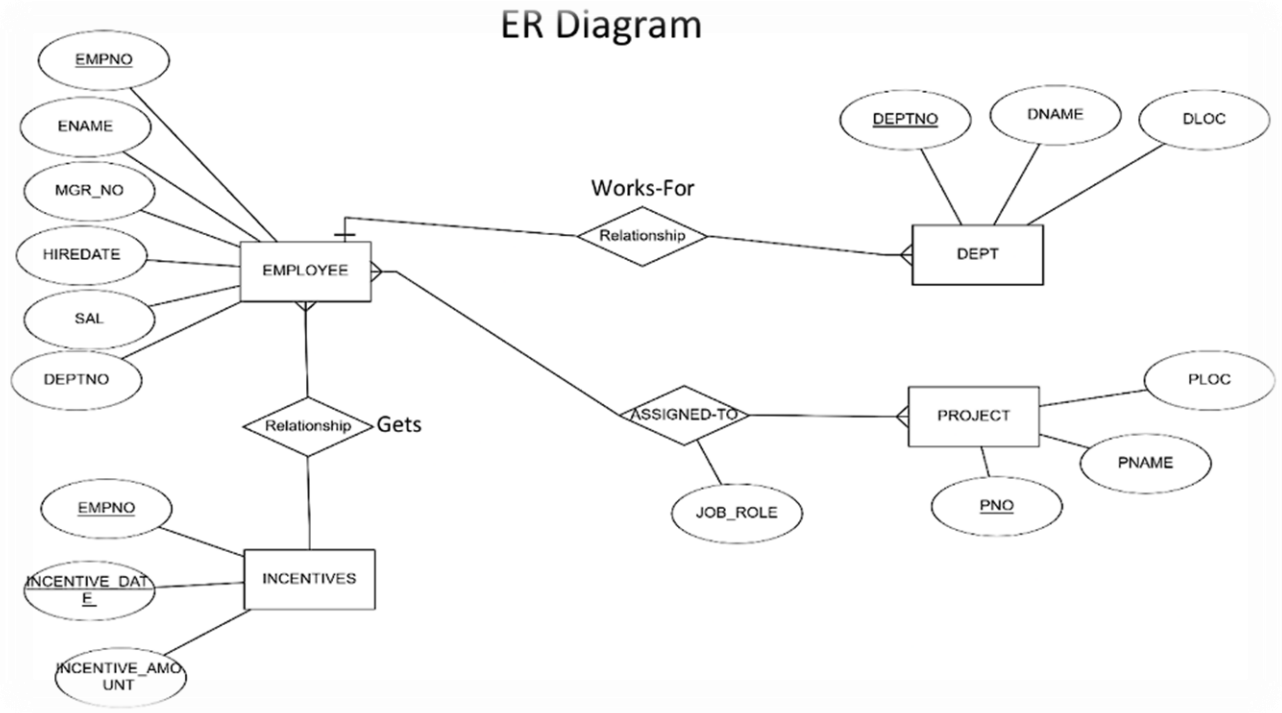
acc_no	branch_name	balance
1	SBI_Chamrajpet	2205
2	SBI_ResidencyRoad	5513
4	SBI_ParlimentRoad	9923
5	SBI_Jantarmantar	8820
8	SBI_ResidencyRoad	4410
9	SBI_ParlimentRoad	3308
10	SBI_ResidencyRoad	5513
11	SBI_Jantarmantar	2205
12	SBI_MantriMarg	2205
NULL	NULL	NULL

EMPLOYEE DATABASE

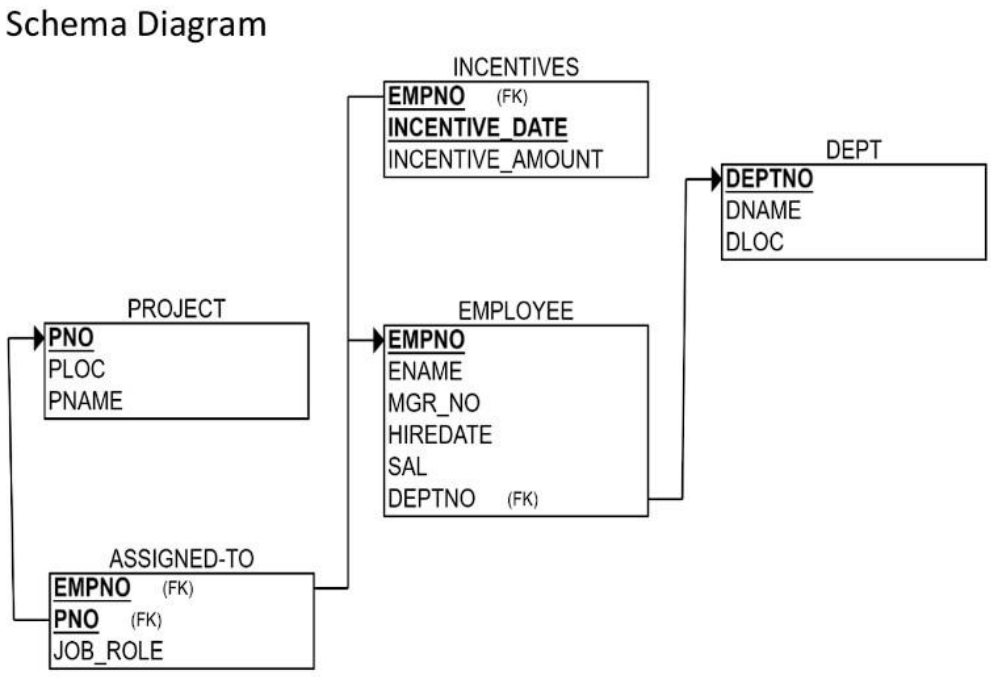
Week-5:

- Create the above tables by properly specifying the primary keys and the foreign keys.
- Enter at least five tuples for each relation.
- Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru
- Get Employee ID's of those employees who didn't receive incentives
- 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.

ER Diagram:



Schema Diagram:



Create database

create database Employee; use

Employee;

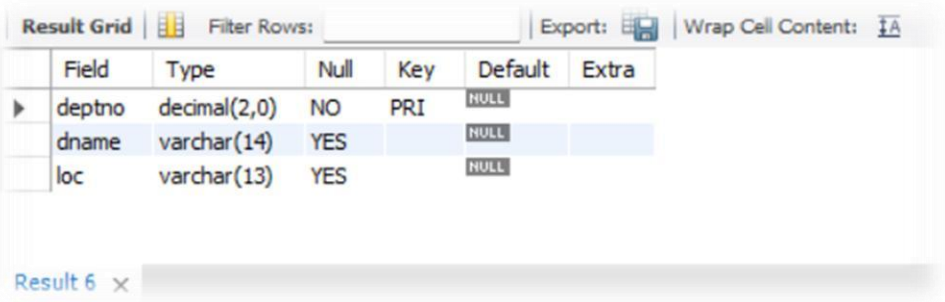
Create table

```
create table Dept ( deptno
decimal(2,0) primary key, dname
varchar(14) default NULL, loc
varchar(13) default NULL
);

create table Emp ( empno
decimal(4,0) primary key, ename
varchar(10) default NULL, mgr_no
decimal(4,0) default NULL,
hiredate date default NULL, sal
decimal(7,2) default NULL, deptno
decimal(2,0) references
dept(deptno) on delete cascade on
update cascade
);

create table Incentives ( empno decimal(4,0) references emp(empno) on delete
cascade on update cascade, incentive_date date, incentive_amount decimal(10,2),
primary key(empno,incentive_date)
);

create table Project ( pno
int primary key, pname
varchar(30) not null, ploc
varchar(30)
);

create table Assigned_to ( empno decimal(4,0) references emp(empno) on delete
cascade on update cascade, pno int references project(pno) on delete cascade on
update cascade, job_role varchar(30), primary key(empno,pno)
```

);

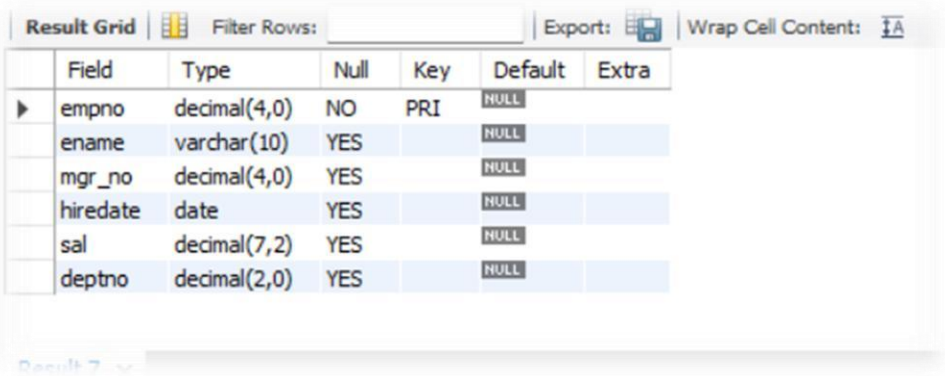
Structure of the table desc

Dept;



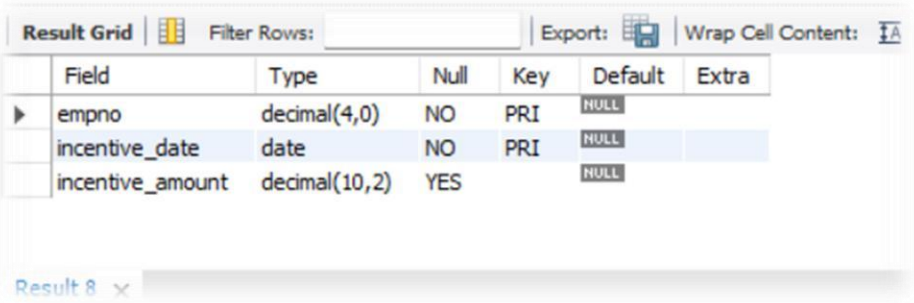
Field	Type	Null	Key	Default	Extra
deptno	decimal(2,0)	NO	PRI	NULL	
dname	varchar(14)	YES		NULL	
loc	varchar(13)	YES		NULL	

desc Emp;



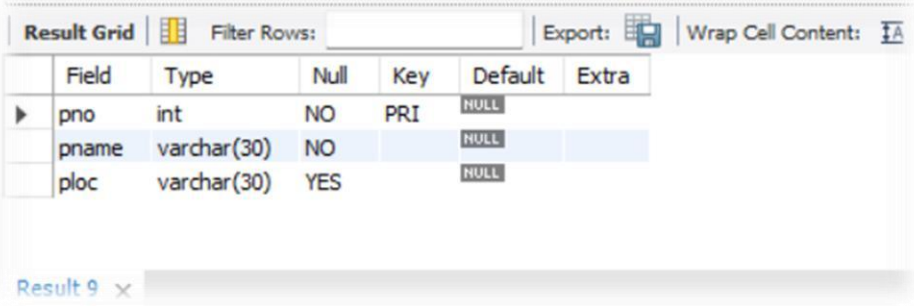
Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
ename	varchar(10)	YES		NULL	
mgr_no	decimal(4,0)	YES		NULL	
hiredate	date	YES		NULL	
sal	decimal(7,2)	YES		NULL	
deptno	decimal(2,0)	YES		NULL	

desc Incentives;



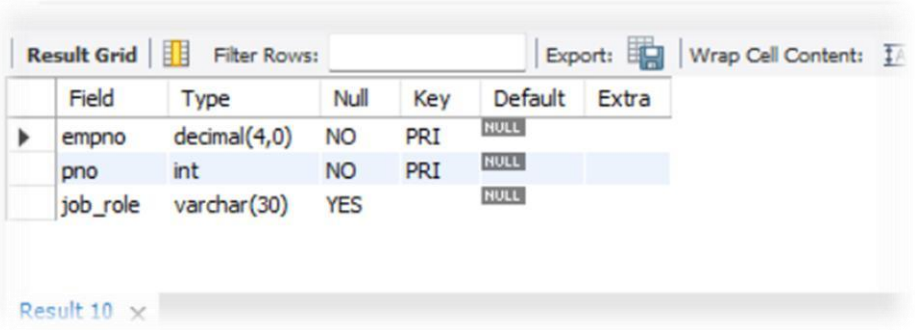
Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
incentive_date	date	NO	PRI	NULL	
incentive_amount	decimal(10,2)	YES		NULL	

desc Project;



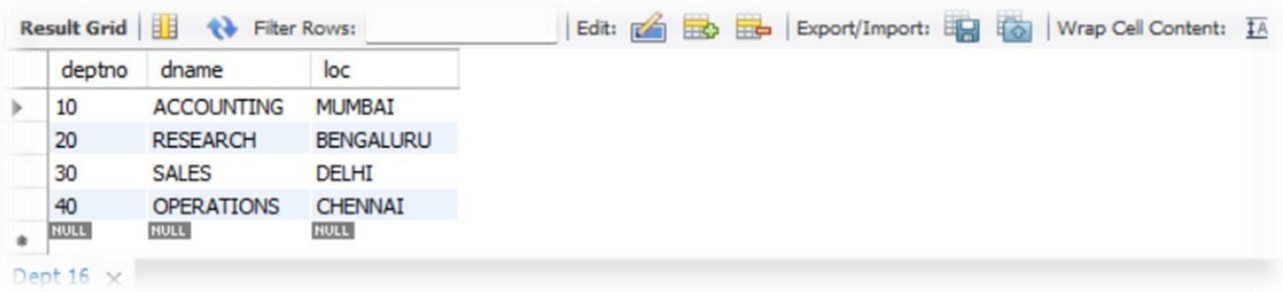
Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
pname	varchar(30)	NO		NULL	
ploc	varchar(30)	YES		NULL	

desc Assigned_to;



Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
pno	int	NO	PRI	NULL	
job_role	varchar(30)	YES		NULL	

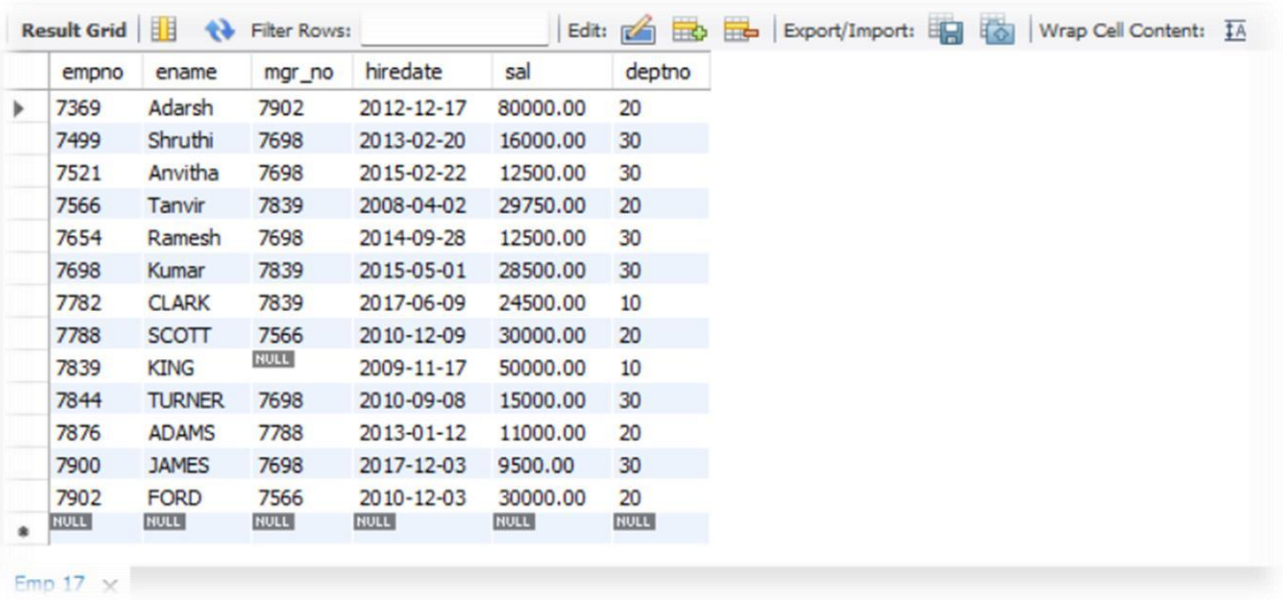
Inserting Values to the table insert into Dept values (10, 'ACCOUNTING', 'MUMBAI'); insert into Dept values (20, 'RESEARCH', 'BENGALURU'); insert into Dept values (30, 'SALES', 'DELHI'); insert into Dept values (40, 'OPERATIONS', 'CHENNAI'); select * from Dept;



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, '2012-12-17', '80000.00', '20');
insert into Emp values (7499, 'Shruthi', 7698, '2013-02-20', '16000.00', '30');
insert into Emp values (7521, 'Anvitha', 7698, '2015-02-22', '12500.00', '30');
insert into Emp values (7566, 'Tanvir', 7839, '2008-04-02', '29750.00', '20');
insert into Emp values (7654, 'Ramesh', 7698, '2014-09-28', '12500.00', '30');
insert into Emp values (7698, 'Kumar', 7839, '2015-05-01', '28500.00', '30');
insert into Emp values (7782, 'CLARK', 7839, '2017-06-09', '24500.00', '10');
insert into Emp values (7788, 'SCOTT', 7566, '2010-12-09', '30000.00', '20');
insert into Emp values (7839, 'KING', NULL, '2009-11-17', '50000.00', '10');
insert into Emp values (7844, 'TURNER', 7698, '2010-09-08', '15000.00', '30');
insert into Emp values (7876, 'ADAMS', 7788, '2013-01-12', '11000.00', '20');

```
insert into Emp values  (7900, 'JAMES', 7698, '2017-12-03', '9500.00', '30');
insert into Emp values  (7902, 'FORD', 7566, '2010-12-03', '30000.00', '20');
select * from Emp;
```

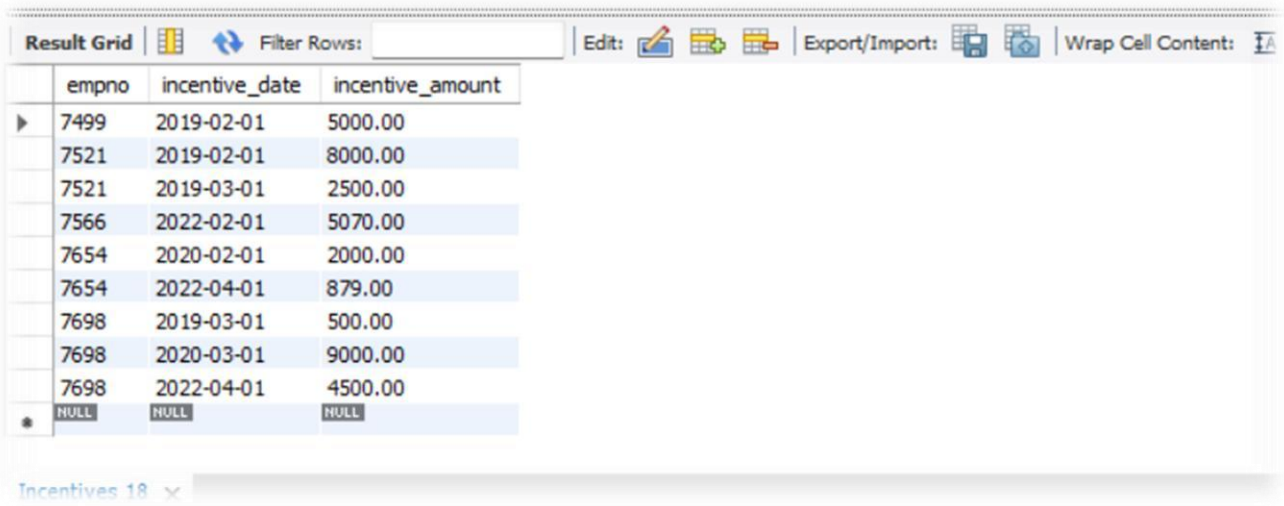


Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

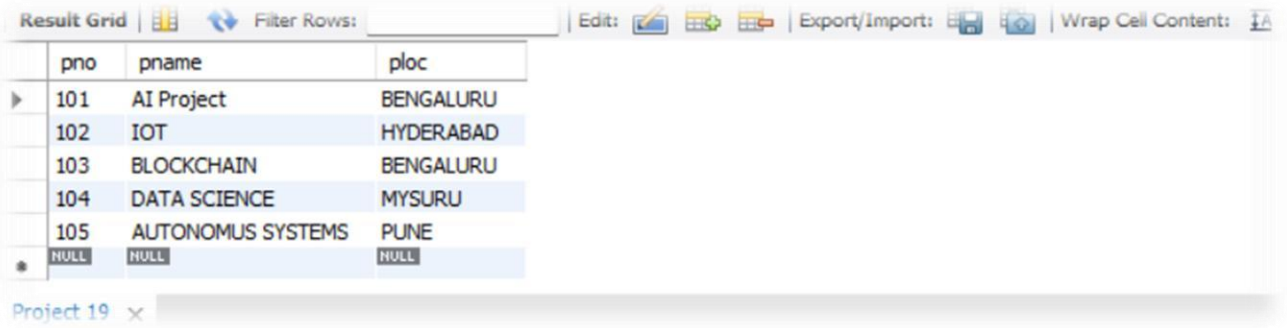
Emp 17

```
insert into Incentives values (7499, '2019-02-01', 5000.00);
insert into Incentives values (7521, '2019-03-01', 2500.00);
insert into Incentives values (7566, '2022-02-01', 5070.00);
insert into Incentives values (7654, '2020-02-01', 2000.00);
insert into Incentives values (7654, '2022-04-01', 879.00);
insert into Incentives values (7521, '2019-02-01', 8000.00);
insert into Incentives values (7698, '2019-03-01', 500.00);
insert into Incentives values (7698, '2020-03-01', 9000.00);
insert into Incentives values (7698, '2022-04-01', 4500.00);
select * from Incentives;
```



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'); insert
into Project values (102, 'IOT', 'HYDERABAD'); insert into Project
values (103, 'BLOCKCHAIN', 'BENGALURU'); insert into Project
values (104, 'DATA SCIENCE', 'MYSURU'); insert into Project
values (105, 'AUTONOMUS SYSTEMS', 'PUNE');
select * from Project;
```

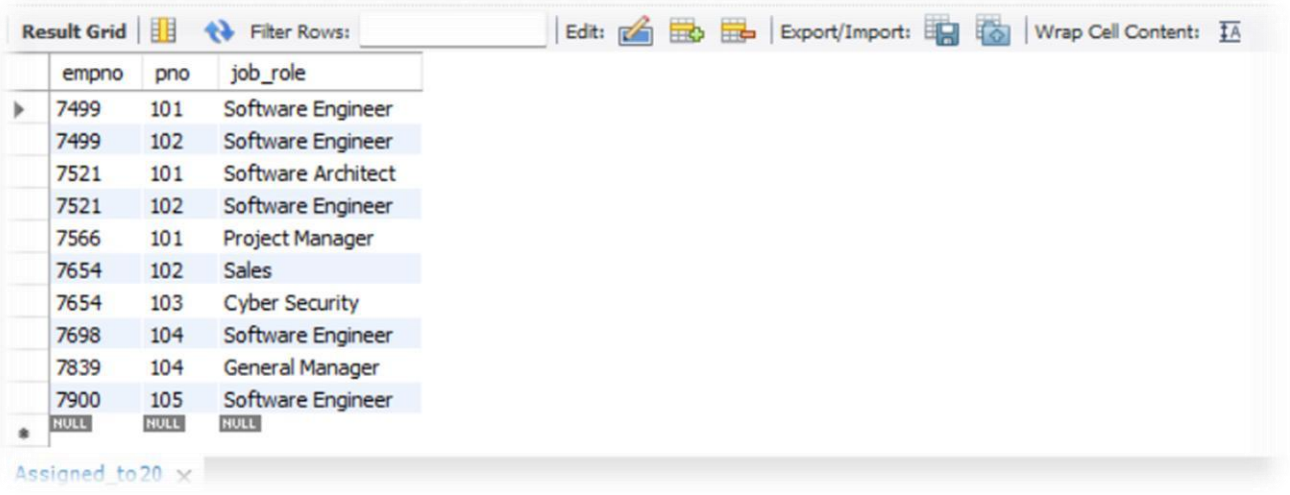


pno	pname	ploc
101	AI Project	BENGALURU
102	IOT	HYDERABAD
103	BLOCKCHAIN	BENGALURU
104	DATA SCIENCE	MYSURU
105	AUTONOMUS SYSTEMS	PUNE
NULL	NULL	NULL

```
insert into Assigned_to values (7499, 101, 'Software Engineer');
insert into Assigned_to values (7521, 101, 'Software
Architect'); insert into Assigned_to values (7566, 101, 'Project
Manager'); insert into Assigned_to values (7654, 102, 'Sales');
insert into Assigned_to values (7521, 102, 'Software Engineer');
insert into Assigned_to values (7499, 102, 'Software Engineer');
insert into Assigned_to values (7654, 103, 'Cyber Security');
insert into Assigned_to values (7698, 104, 'Software Engineer');
insert into Assigned_to values (7900, 105, 'Software Engineer');
```

insert into Assigned_to values (7839, 104, 'General Manager');

select * from Assigned_to;



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

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
7698	104	Software Engineer
7839	104	General Manager
7900	105	Software Engineer
NULL	NULL	NULL

Assigned_to20

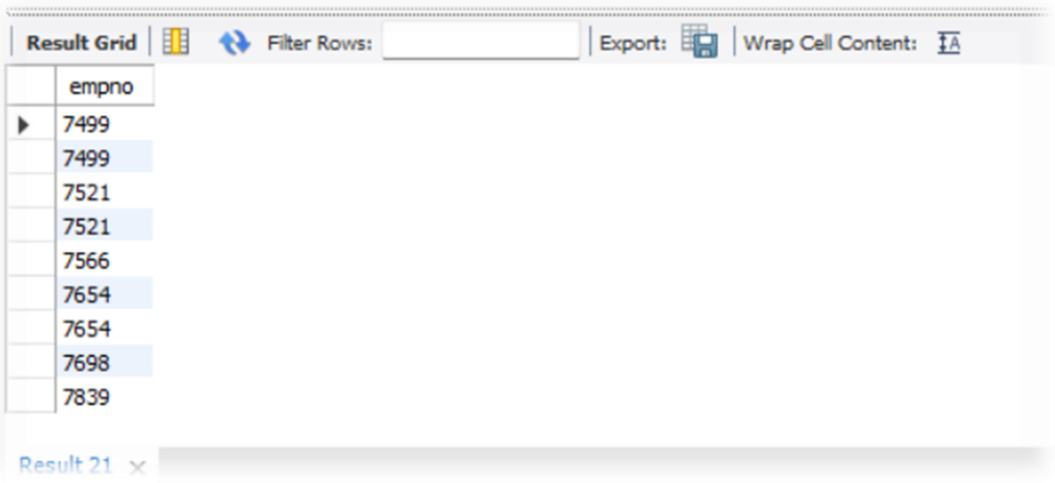
Queries

- **Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru** select e.empno

  from emp e, assigned_to a, project p where

  e.empno=a.empno and a.pno=p.pno and

  p.ploc in ('Bengaluru', 'Hyderabad', 'Mysuru');



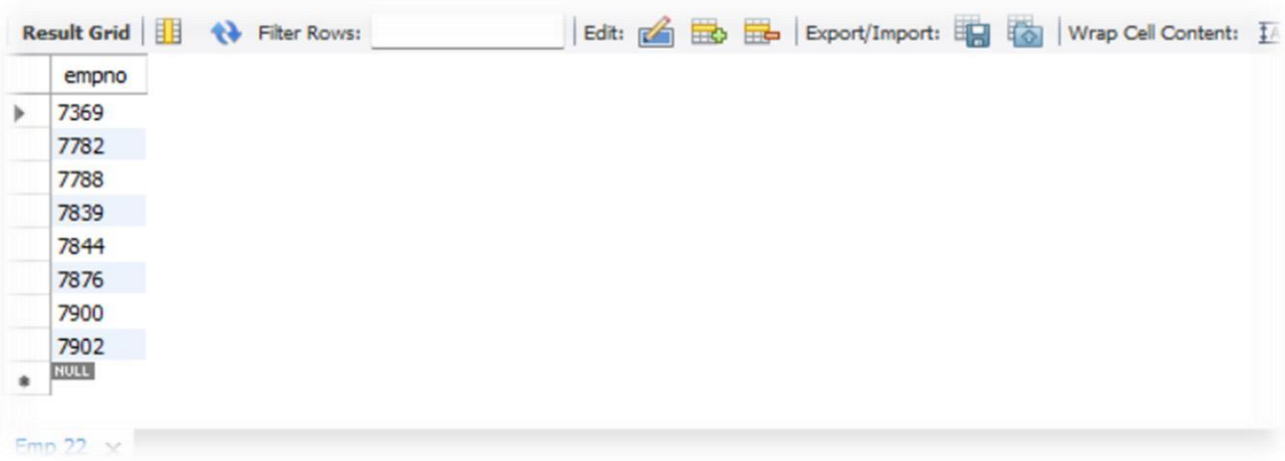
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

empno
7499
7499
7521
7521
7566
7654
7654
7698
7839

Result 21

- **Get Employee ID's of those employees who didn't receive incentives** select empno

  from Emp where empno not in (select empno from incentives);

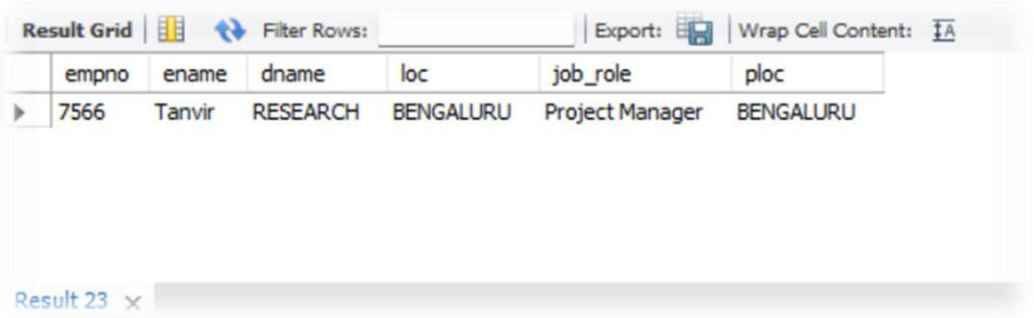


- **Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.**

  select e.empno, e.ename, d.dname, d.loc, a.job_role, p.ploc from

  Emp e, Dept d, Assigned_to a, Project p

  where e.deptno=d.deptno and e.empno=a.empno and a.pno-p.pno and d.loc=p.ploc;



Week-6:

More Queries on Employee Database

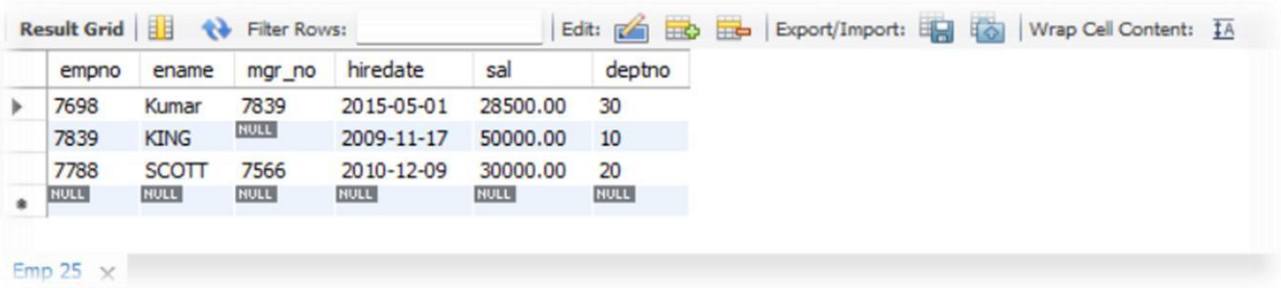
- List the name of the managers with the most employees
- Display those managers name whose salary is more than average salary of his employee?
- SQL Query to find the name of the top level manager of each department.

- SQL Query to find the employee details who got second maximum incentive in Febrauary 2019.
- Display those employees who are working in the same dept where his manager is work. ?
- Write a SQL query to find those employees whose net pay are higher than or equal to the salary of any other employee in the company

Queries

- **List the name of the managers with the most 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

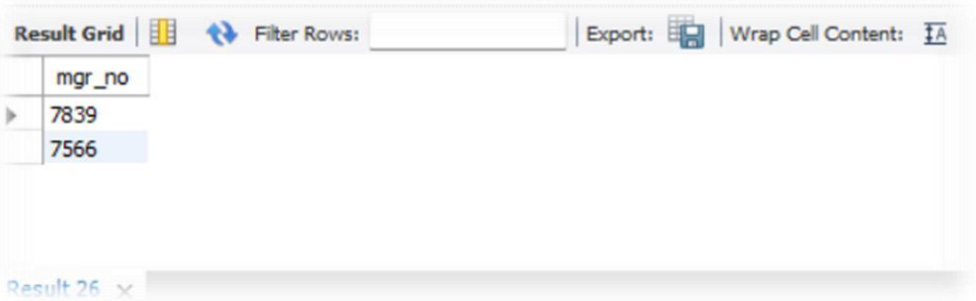
- **Display those managers name whose salary is more than average salary of his employee?** select * from emp m where m.empno in (select mgr_no from emp) and m.sal > (select avg(e.sal) from emp e where e.mgr_no = m.empno);



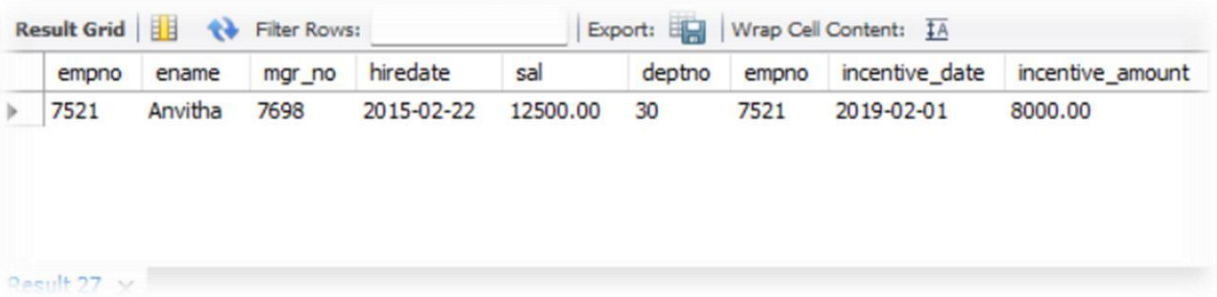
group by mgr_no) a);

- **SQL Query to find the name of the top level manager of each department.**

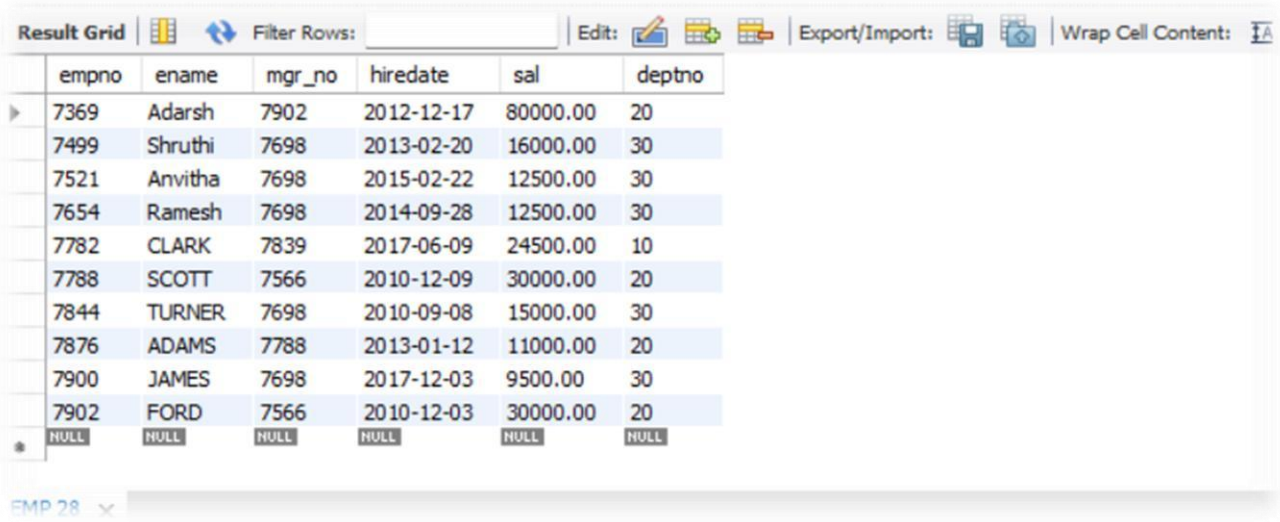
  select distinct m.mgr_no from emp e, emp m
  where e.mgr_no = m.mgr_no and e.deptno = m.deptno and e.empno in
  (select distinct m.mgr_no from emp e, emp m where e.mgr_no =
  m.mgr_no and e.deptno = m.deptno);



- **SQL Query to find the employee details who got second maximum incentive in Febrauary 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);



- **Display those employees who are working in the same dept where his manager is work. ?** select * from emp e where e.deptno = (select e1.deptno from emp e1 where e1.empno=e.mgr_no);



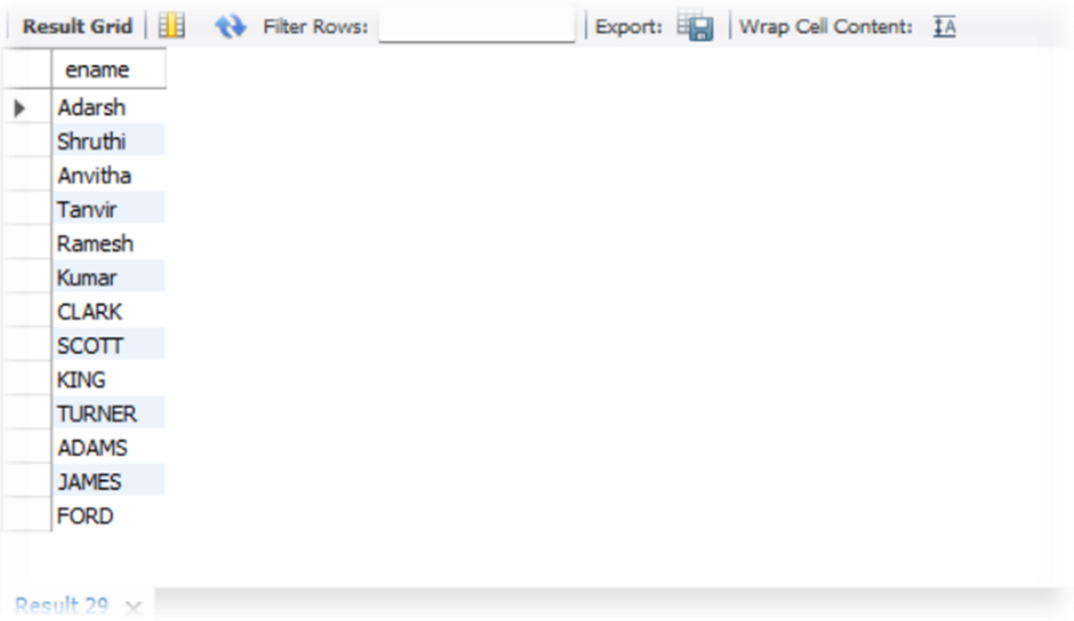
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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 28

- **Write a SQL query to find those employees whose net pay are higher than or equal to the salary of any other employee in the company**

```
select distinct e.ename from emp e,incentives i
where (select max(sal+incentive_amount)
from emp,incentives) >= any
(select sal
from emp e1
where e.deptno=e1.deptno);
```



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

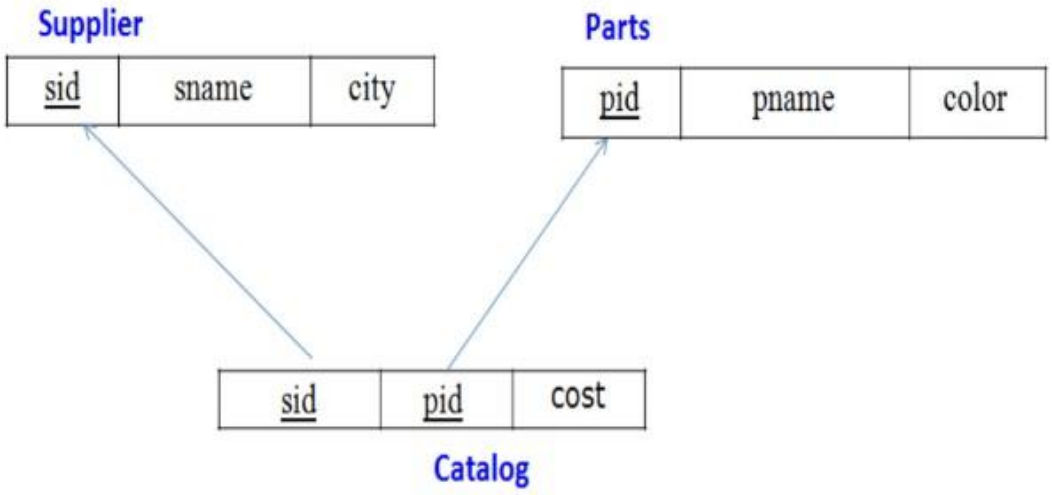
ename
Adarsh
Shruthi
Anvitha
Tanvir
Ramesh
Kumar
CLARK
SCOTT
KING
TURNER
ADAMS
JAMES
FORD

Result 29

Week-7:

1. Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2. Insert appropriate records in each table.
3. Find the pnames of parts for which there is some supplier.
4. Find the snames of suppliers who supply every part.
5. Find the snames of suppliers who supply every red part.
6. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
7.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).
8. For each part, find the sname of the supplier who charges the most for that
part.

Schema Diagram:-



Create Database:-

create database Supplier_database; use
supplier_database;

Create Table:-

CREATE TABLE Supplier (
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
DECIMAL(1
0,2),
   PRIMARY KEY (sid, pid),
   FOREIGN KEY (sid) REFERENCES Supplier(sid),
   FOREIGN KEY (pid) REFERENCES Parts(pid)
);
```

Structure of the table

desc supplier;

Result Grid | Filter Rows: | Export:

	Field	Type	Null	Key	Default	Extra
▶	sid	int	NO	PRI	NULL	
	sname	varchar(40)	YES		NULL	
	city	varchar(40)	YES		NULL	

desc parts;

	Field	Type	Null	Key	Default	Extra
▶	pid	int	NO	PRI	NULL	
	pname	varchar(40)	YES		NULL	
	colour	varchar(20)	YES		NULL	

desc catalog;

	Field	Type	Null	Key	Default	Extra
▶	sid	int	YES	MUL	NULL	
	pid	int	YES	MUL	NULL	
	cost	int	NO	PRI	NULL	

Insert values into tables

INSERT INTO Supplier VALUES

(1, 'Acme Widget Suppliers', 'London'),

(2, 'Best Parts Co', 'Paris'),

(3, 'Global Supplies', 'New York');

	sid	sname	city
▶	1	Acme Widget Suppliers	New York
	2	Global Parts Co	London
	3	Universal Traders	Paris
*	NULL	NULL	NULL

INSERT INTO Parts VALUES

(101, 'Bolt', 'Red'),

(102, 'Nut', 'Green'),

(103, 'Screw', 'Red'), (104,

'Washer', 'Blue'); select *

from Parts;

	pid	pname	colour
▶	101	Bolt	Red
	102	Nut	Green
	103	Screw	Red
	104	Washer	Blue
*	NULL	NULL	NULL

INSERT INTO Catalog VALUES

(1, 101, 12.50),

(1, 102, 14.00),

(2, 101, 10.00),

(2, 103, 15.50),

(3, 101, 11.00),

(3, 104, 16.00); (1,

103, 15.00),

(1, 104, 17.00);

sid	pid	cost
1	101	10
3	101	11
2	101	12
3	102	13
2	103	14
1	102	15
3	104	18
NULL	NULL	NULL

Queries:

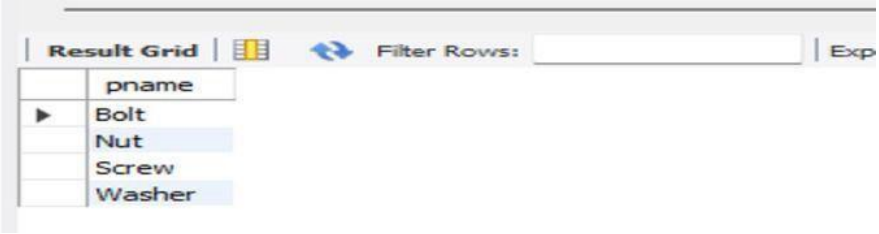
Find the pnames of parts for which there is some supplier?

SELECT DISTINCT p.pname

FROM Parts p

JOIN Catalog c ON p.pid = c.pid;

SELECT s.name



Find the snames of suppliers who supply every part?

SELECT s.sname

FROM Supplier s

WHERE NOT EXISTS (

SELECT p.pid

FROM parts p

WHERE p.pid NOT IN (

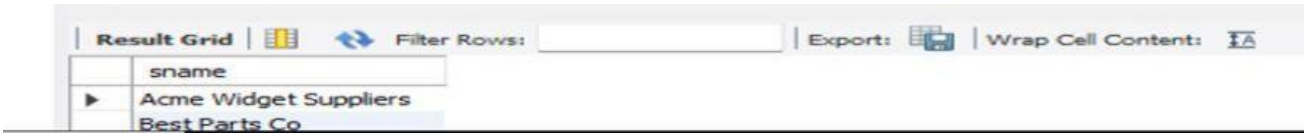
SELECT c.pid FROM Catalog c WHERE c.sid = s.sid

)

);

Find the snames of suppliers who supply every red part?

```
SELECT s.sname
FROM Supplier s
WHERE NOT EXISTS (
SELECT p.pid
FROM Parts p
WHERE p.color = 'Red'
AND p.pid NOT IN (
SELECT c.pid FROM Catalog c WHERE c.sid = s.sid
)
);
```



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

```
SELECT p.pname
FROM Parts p
JOIN Catalog c ON p.pid – c.pid
JOIN Supplier s ON s.sid – c.sid
WHERE s.sname = 'Acme Widget Suppliers'
AND p.pid NOT IN  (
SELECT c2.pid
FROM Catalog c2
JOIN Supplier s2 ON s2.sid – c2.sid
WHERE s2.sname <> 'Acme Widget Suppliers'
);
```

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

```
SELECT p.pname, s.sname, c.cost
FROM Catalog c
JOIN Supplier s ON  s.sid = c.sid
JOIN Parts p ON  p.pid = c.pid
WHERE c.cost = (
SELECT MAX(c2.cost)
FROM Catalog c2
WHERE c2.pid = c.pid
);
```

pname	sname	cost
Bolt	Acme Widget Suppliers	12.50
Nut	Acme Widget Suppliers	14.00
Washer	Acme Widget Suppliers	17.00
Screw	Best Parts Co	15.50

Find the snames of suppliers who charge more for some part than the average cost of that part

```
SELECT DISTINCT s.sname
FROM Supplier s
JOIN Catalog c ON s.sid = c.sid
WHERE c.cost > (
SELECT AVG(c2.cost)
FROM CAtalog c2
WHERE c2.pid = c.pid
);
```

Week-9:-

No SQL Student

Create a database “Student” with the following attributesRollno, Age, ContactNo, EmailId?

use Student

db.createCollection("student")

```
switched to db Student
Student>  db.createCollection("student")
{ ok: 1 }
Student>
```

Insert appropriate values?

db.student.insertMany([{ Rollno: 10, Name: "ABC", Age: 20, ContactNo: "9876543210", EmailId: "abc@gmail.com" },{ Rollno: 11, Name: "ABC", Age: 21, ContactNo: "9876543211", EmailId: "abc11@gmail.com" }, { Rollno: 12, Name: "XYZ", Age: 22, ContactNo: "9876543212", EmailId: "xyz@gmail.com" }])

```
Student> db.student.insertMany([
...   { Rollno: 10, Name: "ABC", Age: 20, ContactNo: "9876543210", EmailId: "abc@gmail.com" },
...   { Rollno: 11, Name: "ABC", Age: 21, ContactNo: "9876543211", EmailId: "abc11@gmail.com" },
...   { Rollno: 12, Name: "XYZ", Age: 22, ContactNo: "9876543212", EmailId: "xyz@gmail.com" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693e33cbb4911698371e2626'),
    '1': ObjectId('693e33cbb4911698371e2627'),
    '2': ObjectId('693e33cbb4911698371e2628')
  }
}
```

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

db.student.updateOne({ Rollno: 10 }, { $set: { EmailId: "newemail10@gmail.com" } })

```
Student> db.student.updateOne( { Rollno: 10 }, { $set: { EmailId: "newemail10@gmail.com" } } )
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

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

db.student.updateOne({ Rollno: 11, Name: "ABC" }, { $set: { Name: "FEM" } }

```
Student> db.student.updateOne(
...    { Rollno: 11, Name: "ABC" },
...    { $set: { Name: "FEM" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

Drop the table.

db.student.drop()

```
Student> db.student.drop()
true
```

No SQL Customer

Create a collection by name Customers with the following attributes.

Cust_id, Acc_Bal, Acc_Type?

use bankDB

db.createCollection("Customers")

Insert at least 5 values into the table?

```
db.Customers.insertMany([
  { Cust_id: 101, Acc_Bal: 1500,  Acc_Type: "Z" },
  { Cust_id: 101, Acc_Bal: 1600,  Acc_Type: "A" },
  { Cust_id: 102, Acc_Bal: 1700, Acc_Type: "B" },
  { Cust_id: 103, Acc_Bal: 1800,  Acc_Type: "C" },
  { Cust_id: 103, Acc_Bal: 800,  Acc_Type: "Z" }])
```

```
test> use CollegeDB
switched to db CollegeDB
CollegeDB> db.createCollection("Customers")
{ ok: 1 }
CollegeDB> db.Customers.insertMany([{Cust_id:101,Acc_bal:1500,Acc_type:"Z"},{Cust_id:102,Acc_
bal:1600,Acc_type:"A"},{Cust_id:103,Acc_bal:1700,Acc_type:"B"},{Cust_id:104,Acc_bal:1800,Acc_
type:"C"},{Cust_id:105,Acc_bal:800,Acc_type:"Z"}])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('693a362307f57dbf9f63b112'),
    '1': ObjectId('693a362307f57dbf9f63b113'),
    '2': ObjectId('693a362307f57dbf9f63b114'),
    '3': ObjectId('693a362307f57dbf9f63b115'),
    '4': ObjectId('693a362307f57dbf9f63b116')
  }
```

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

```
db.Customers.aggregate([
  { $match: { Acc_type: "Z" } },
```

{ $group: { _id: "$Cust_id", TotalBalance: { $sum: "$Acc_bal" }}},

{ $match: { TotalBalance: { $gt: 1200 } } }])

```
CollegeDB>  db.Customers.find({Acc_bal:{$gt:1200},Acc_type:"Z"})
[
  {
    _id: ObjectId('693a362307f57dbf9f63b112'),
    Cust_id: 101,
    Acc_bal: 1500,
    Acc_type: 'Z'
  }
]
```

Determine Minimum and Maximum account balance for each customer_id.?

db.Customers.aggregate([{$group: { _id: "$Cust_id", MinBalance: { $min: "$Acc_Bal" },MaxBalance: { $max: "$Acc_Bal" }}}])

```
CollegeDB> db.Customers.aggregate([
...   {
...     $group: {
...       _id: "$Cust_id",
...       Min_Balance: { $min: "$Acc_Bal" },
...       Max_Balance: { $max: "$Acc_Bal" }
...     }
...   }
... ])
...
[
  { _id: 104, Min_Balance: null, Max_Balance: null },
  { _id: 102, Min_Balance: null, Max_Balance: null },
  { _id: 101, Min_Balance: null, Max_Balance: null },
  { _id: 103, Min_Balance: null, Max_Balance: null },
  { _id: 105, Min_Balance: null, Max_Balance: null }
]
```

Drop the table. db.Customers.drop()

```
CollegeDB> db.customers.drop();
true
CollegeDB>
```

WEEK 10:

No SQL Restaurant

Write NoSQL Queries on “Restaurant” collection. db.createCollection("restaurants")

```
test> use CollegeDB
switched to db CollegeDB
CollegeDB> db.createCollection("restaurants")
{ ok: 1 }
```

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

db.restaurants.insertMany([{restaurant_id: 1,name: "Spice Hub",town: "Hyderabad",cuisine: "Indian",score: 8},{restaurant_id: 2,name: "Food Palace",town: "Bangalore"cuisine:

"Chinese",score: 12 },{restaurant_id: 3,name: "Tandoori Treats",town: "Delhi",cuisine: "Indian",score: 9 },{restaurant_id: 4,name: "Pizza Corner",town: "Chennai",cuisine: "Italian",score: 15},{ restaurant_id: 5,name: "Urban Bites",town: "Mumbai",cuisine: "Continental", score: 10}])

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

db.restaurants.find().sort({ name: -1 })

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.

db.restaurants.find({ score: { $lte: 10 } }, { restaurant_id: 1, name: 1, town: 1, cuisine: 1, _id: 0 })

```
CollegeDB> db.restaurants.find(
... { score: { $lte: 10 } }, { restaurant_id: 1, name: 1, town: 1, cuisine: 1, _id: 0 } )
[
  {
    restaurant_id: 1,
    name: 'Spice Hub',
    town: 'Hyderabad',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 3,
    name: 'Tandoori Treats',
    town: 'Delhi',
    cuisine: 'Indian'
  },
  {
    restaurant_id: 5,
    name: 'Urban Bites',
    town: 'Mumbai',
    cuisine: 'Continental'
  }
]
```

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

db.restaurants.aggregate([{ $group: { _id: "$restaurant_id", avgScore: { $avg: "$score" }}}])

```
]
CollegeDB> db.restaurants.aggregate([
... { $group: {  _id: "$restaurant_id", avgScore: { $avg: "$score" }}}])
[
  { _id: 5, avgScore: 10 },
  { _id: 2, avgScore: 12 },
  { _id: 1, avgScore: 8 },
  { _id: 3, avgScore: 9 },
  { _id: 4, avgScore: 15 }
]
CollegeDB>
```

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

db.restaurants.find({ "address.zipcode": { $regex: "^10" } }, { name: 1, address: 1, _id: 0 })

```
]
CollegeDB> db.restaurants.find(
... { "address.zipcode": { $regex: "^10" } }, { name: 1, address: 1, _id: 0 } )

CollegeDB>
```