VISVESVARAYA TECHNOLOGICAL UNIVERSITY

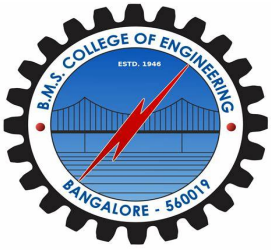
"Jnana Sangama", Belgaum - 590014, Karnataka

LAB REPORT
on
Database Management Systems (22CS3PCDBM)

Submitted by

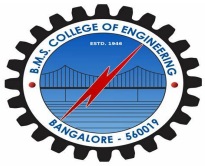
GAMANA YELURI R (1BM21CS065)

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
OCTOBER-2022 to FEB-2023

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 (22CS3PCDBM)” carried out by **GAMANA YELURI R (1BM21CS065),** who is a 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 during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of Database Management Systems (22CS3PCDBM) work prescribed for the said degree.

Dr.Nandhini Vineeth
Assistant Professor
Department of CSE
BMSCE, Bengaluru

Dr. Jyothi S Nayak
Professor & Head
Department of CSE
BMSCE, Bengaluru

INDEX

INSURANCE DATABASE

PERSON (driver_id: String, name: String, address: String)

CAR (reg_num: String, model: String, year: int)

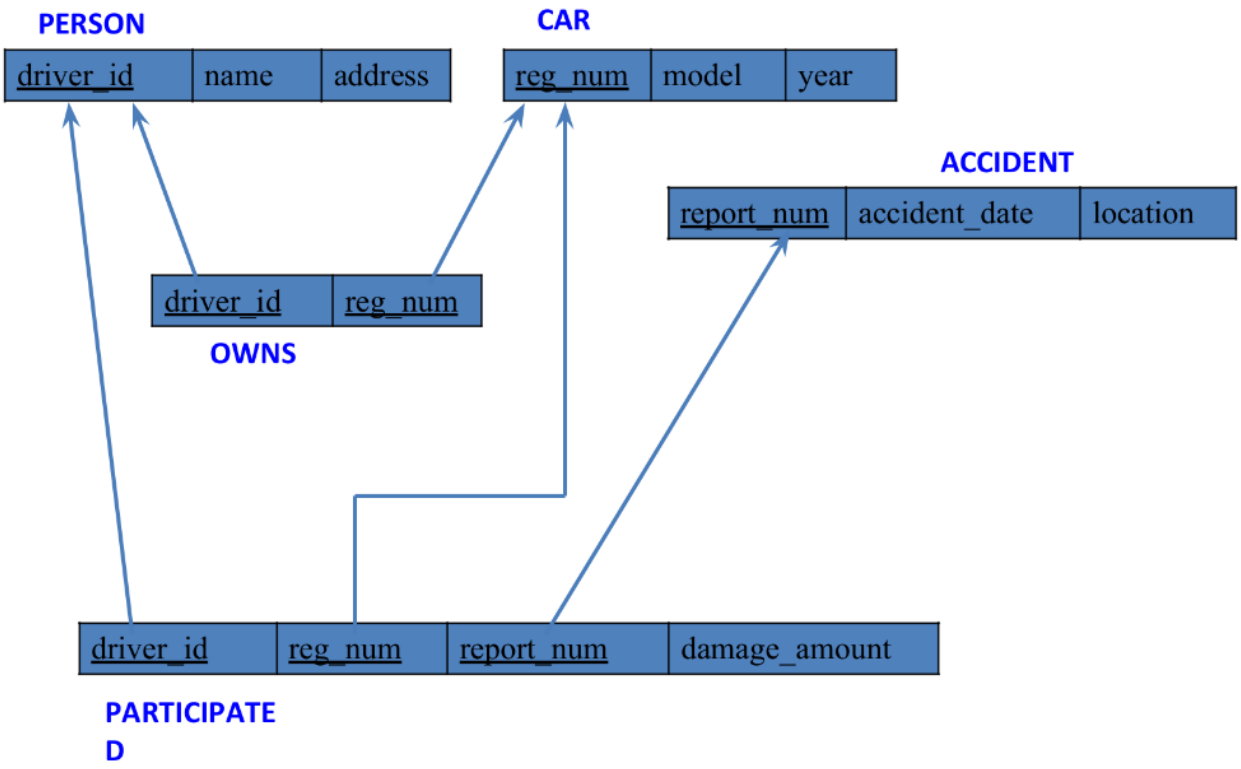
ACCIDENT (report_num: int, accident_date: date, location: String)

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

SCHEMA DIAGRAM



CREATE DATABASE

```
create database 1bm21cs050_insurance;
 use 1bm21cs050_insurance;
create table person(
driver_id varchar(20),
 name varchar(20),
 address varchar(30),
 primary key(driver_id)
 );
desc person;
create table car(
reg_num varchar(10),
model varchar(10),
year int,
primary key(reg_num)
 );
 desc car;
 create table accident(
 report_num int,
accident_date date,
 location varchar(20),
 primary key(report_num) );
 desc accident;
 create table owns(
driver_id varchar(20),
reg_num varchar(10),
 primary key(driver_id,reg_num),
 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(10),
```

```
report_num int,
damage_amount int,
foreign key(driver_id) references person(driver_id),
foreign key(reg_num) references car(reg_num),
 foreign key(driver_id) references person(driver_id), foreign
key(report_num) references accident(report_num)
);
desc participated;
```

STRUCTURE OF TABLE

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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

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

Field	Type	Null	Key	Default	Extra
report_num	int	NO	PRI	NULL	
accident_date	date	YES		NULL	
location	varchar(50)	YES		NULL	

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

Field	Type	Null	Key	Default	Extra
reg_num	varchar(15)	NO	PRI	NULL	
model	varchar(10)	YES		NULL	
year	int	YES		NULL	

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

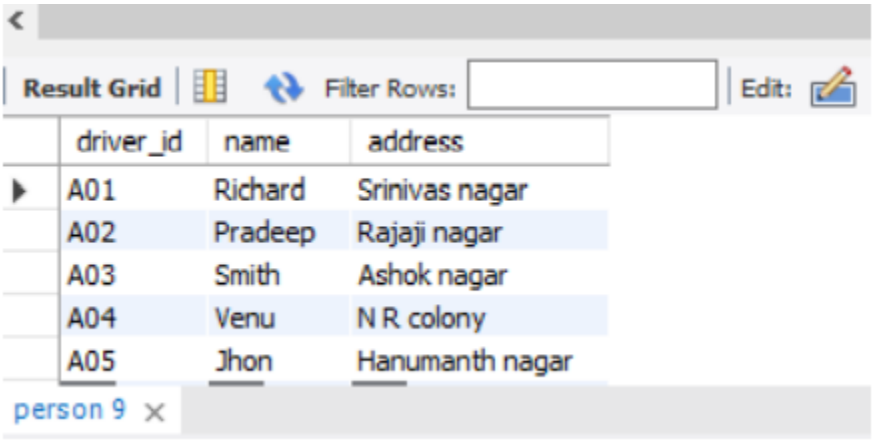
Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
address	varchar(50)	YES		NULL	

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

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	

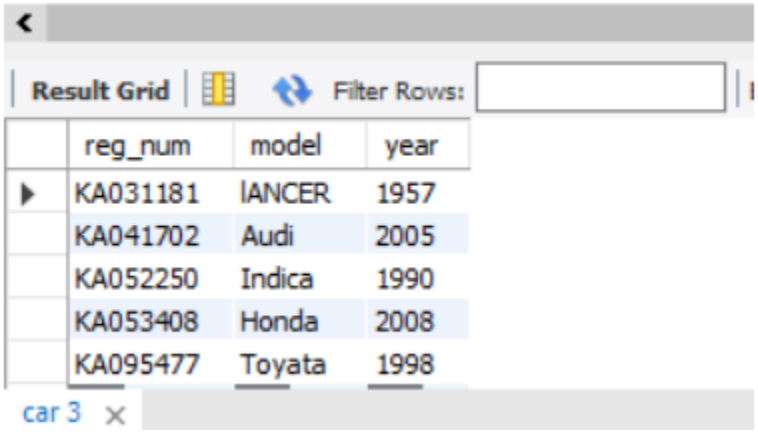
INSERTING VALUES TO TABLE

```
insert into person values('A01', 'Richard','Srinivasnagar');
insert into person values('A02', 'Pradeep','Rajajinagar');
insert into person values('A03', 'Smith','Ashoknagar');
insert into person values('A04', 'Venu','NR Colony');
insert into person values('A05', 'Jhon','Hanumanthnagar');
select*from person;
```




```
insert into car values('KA053350','Indica', 1990);
insert into car values('KA031181','Lancer', 1957);
insert into car values('KA095477','Toyata', 1998);
```

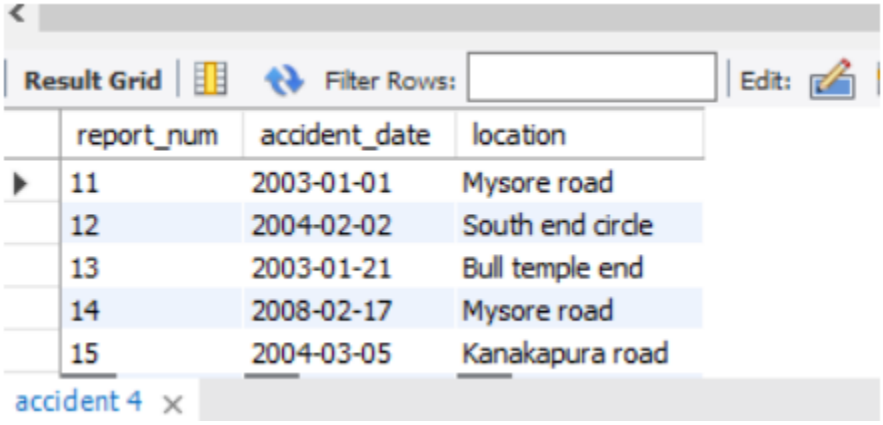
```
insert into car values('KA53408','Honda', 2008);
insert into car values('KA05459','Audi', 2005);
select*from car;
```



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

car 3

```
insert into accident values(11, '2003-01-01', 'Mysore road');
insert into accident values(12, '2004-02-02', 'South end circle');
insert into accident values(13, '2003-01-21', 'NR colony');
insert into accident values(14, '2004-03-05', 'Kanakapura road');
insert into accident values(15, '2005-01-05', 'Bull temple');
select * from accident;
```

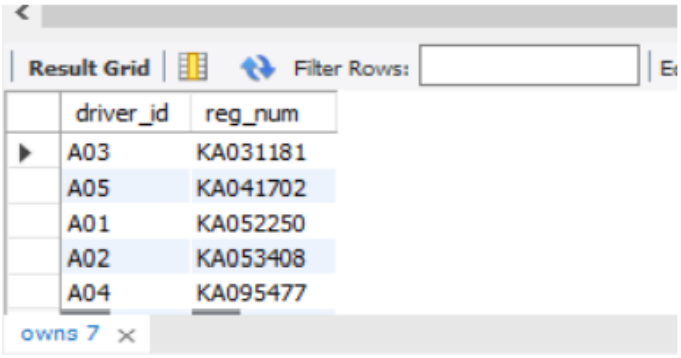


report_num	accident_date	location
11	2003-01-01	Mysore road
12	2004-02-02	South end circle
13	2003-01-21	Bull temple end
14	2008-02-17	Mysore road
15	2004-03-05	Kanakapura road

accident 4

```
insert into owns values('A01', 'KA053350');
insert into owns values('A02', 'KA031181');
```

```
insert into owns values('A03', 'KA095477');
insert into owns values('A04', 'KA53408');
insert into owns values('A05', 'KA05459');
select * from owns;
```

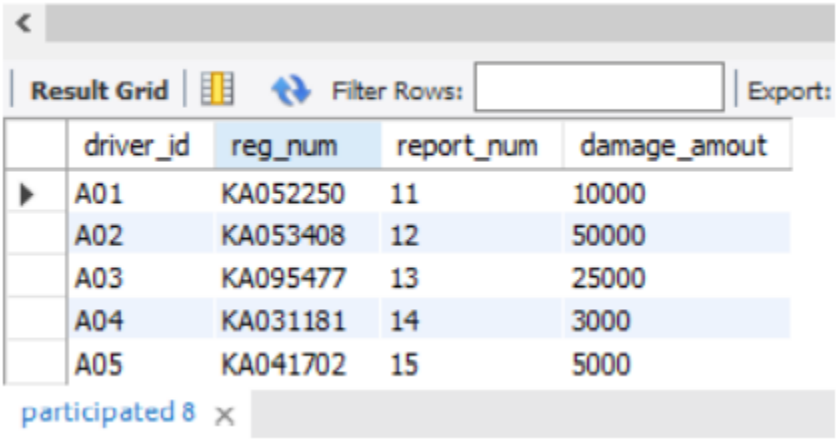


Result Grid | Filter Rows:

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

owns 7

```
Insert into participated values ('A01', 'KA053350',11,10000);
Insert into participated values ('A02', 'KA031181', 12, 50000);
Insert into participated values ('A03', 'KA095477',13,25-000);
Insert into participated values ('A04', 'KA53408',14,3000);
insert into participated values('A05', 'KA05459',15,5000);
select * from participated;
```



Result Grid | Filter Rows: | Export:

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

participated 8

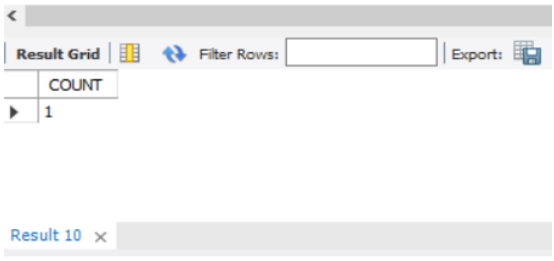
Queries

1. Update the damage amount to 25000 for the car with a specific reg-num (example '053408') for which the accident report number was 12

update participated set damage_amount=25000 where reg_num='053408' and report_num=12;

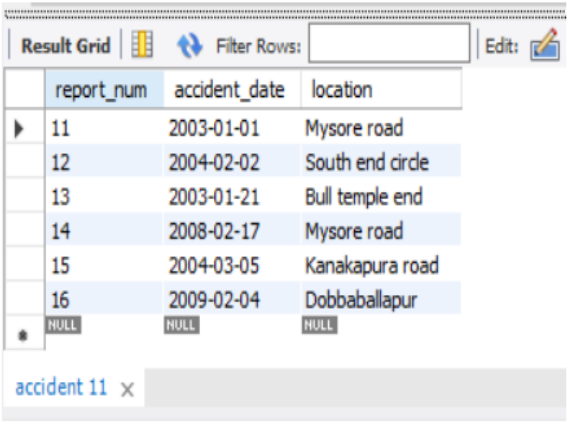
2. Find the total number of people who owned cars that were involved in accidents in 2008.

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



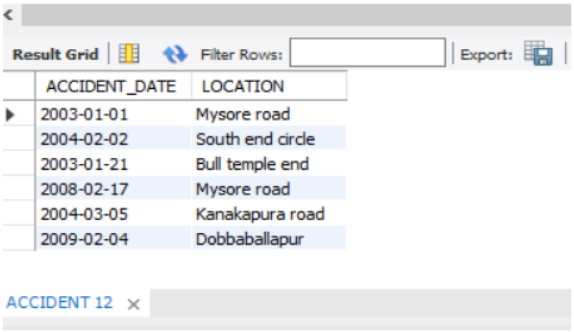
3. Add a new accident to the database.

insert into accident values(16,'2008-03-08',"Doddaballapura"); select * from accident;



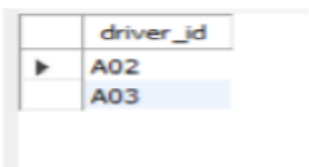
4. Display Accident date and location

select accident_date, location from accident;



5. 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



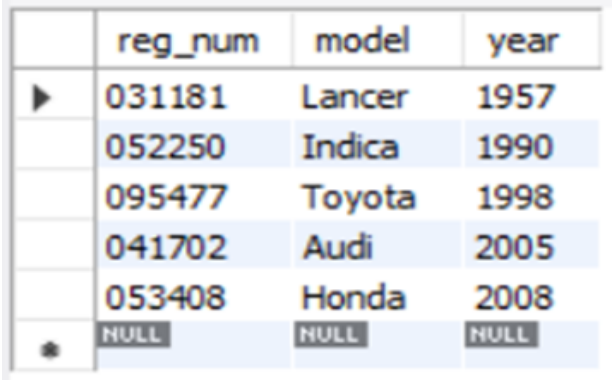
More Queries on INSURANCE DATABASE

- LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.
- FIND THE AVERAGE DAMAGE AMOUNT
- DELETE THE TUPLE 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.

QUERIES

1.List the entire participated relation in the descending order of damage amount.

select * from participated order by damage_amount desc;



	reg_num	model	year
▶	031181	Lancer	1957
	052250	Indica	1990
	095477	Toyota	1998
	041702	Audi	2005
	053408	Honda	2008
*	NULL	NULL	NULL

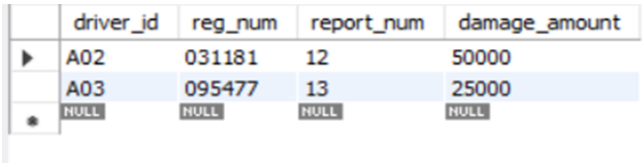
2. Find the average damage amount

select avg(damage_amount) from participated;

	count(report_num)
▶	1

3. Delete the tuple whose damage amount is below the average damage amount

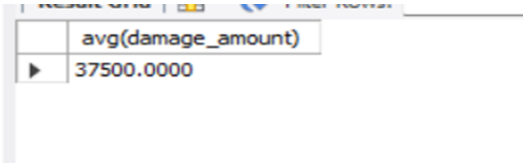
delete from participated where damage_amount<(select p.damage_amount from(select avg(damage_amount)as damage_amount from participated) p);



	driver_id	reg_num	report_num	damage_amount
▶	A02	031181	12	50000
	A03	095477	13	25000
*	NULL	NULL	NULL	NULL

4. List the name of drivers whose damage is greater than the average damage amount.

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



	avg(damage_amount)
▶	37500.0000

5. Find maximum damage amount. select damage_amount from participated having max(damage_amount)

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,

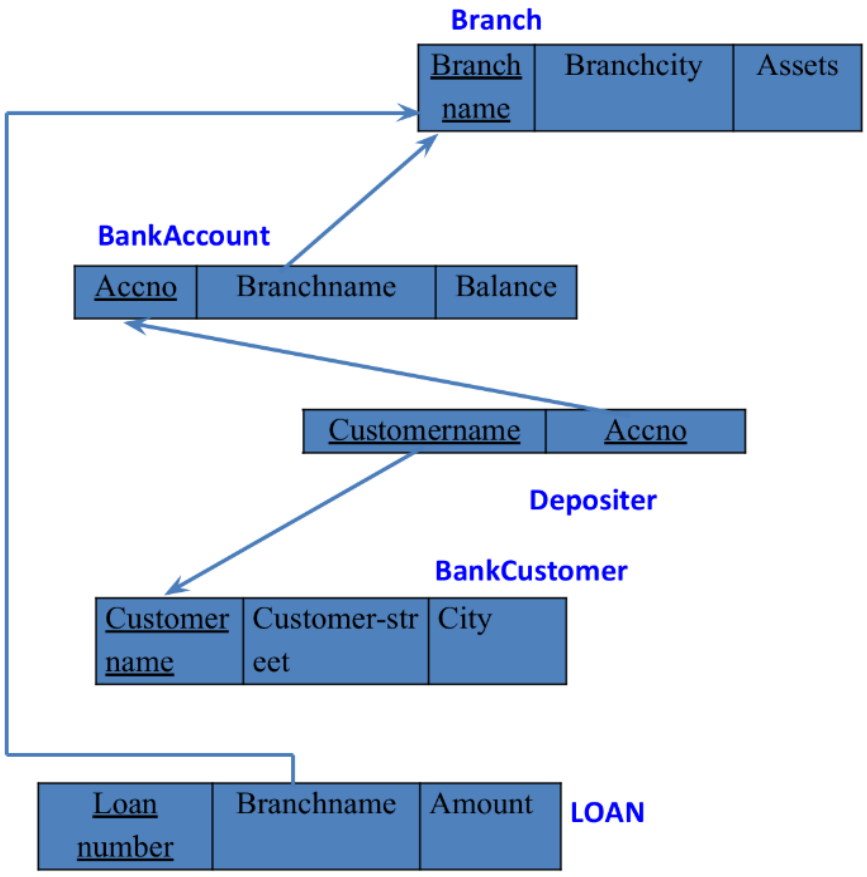
customer-city: String)

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

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

- 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.

SCHEMA DIAGRAM



CREATION OF TABLES

```
create database 1bm21cs050_bank;
use 1bm21cs050_bank;
create table branch(
 branch_name varchar(10),
branch_city varchar(10),
assests int,
PRIMARY KEY(branch_name)
);


create table bank_account(
acc_no varchar(10),
branch_name varchar(10),
 balance int,
PRIMARY KEY(acc_no),
FOREIGN KEY(branch_name) REFERENCES branch(branch_name)
);

create table depositer(
customer_name varchar(10),
acc_no varchar(10),
PRIMARY KEY(customer_name, acc_no),
 FOREIGN KEY(acc_no) REFERENCES bank_account(acc_no)
);

 create table bank_customer(
customer_name varchar(10),
customer_street varchar(10),
 city varchar(10),
PRIMARY KEY(customer_name),
FOREIGN KEY (customer_name) REFERENCES
depositer(customer_name)
```

);

```
 create table loan(
loan_number int,
branch_name varchar(10),
amount int ,
PRIMARY KEY(loan_number),
FOREIGN KEY(branch_name) REFERENCES branch
(branch_name)
);
```

STRUCTURE OF TABLE

Field	Type	Null	Key	Default	Extra
branchname	varchar(50)	NO	PRI	NULL	
branchcity	varchar(50)	YES		NULL	
assets	int	YES		NULL	

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

Field	Type	Null	Key	Default	Extra
accno	int	NO	PRI	NULL	
branchname	varchar(50)	YES	MUL	NULL	
balance	int	YES		NULL	

Field	Type	Null	Key	Default	Extra
customername	varchar(50)	NO	PRI	NULL	
customerstreet	varchar(50)	YES		NULL	
city	varchar(50)	YES		NULL	

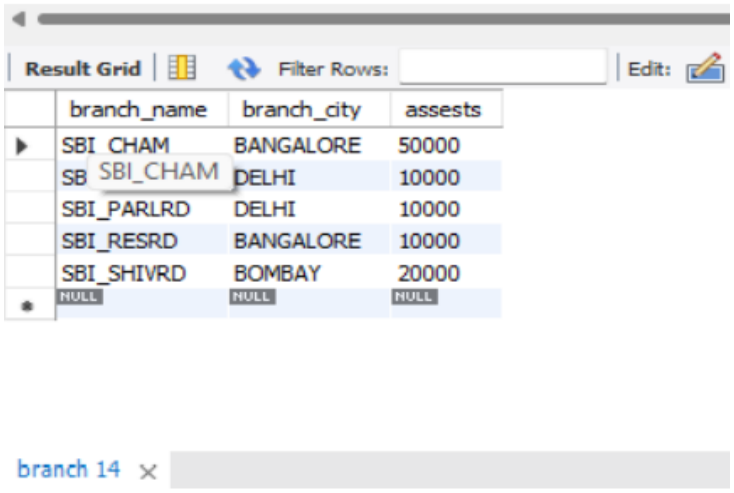
Result Grid | Filter Rows: | Export: | Wrap Cell

Field	Type	Null	Key	Default	Extra
customername	varchar(50)	YES	MUL	NULL	
accno	int	NO	PRI	NULL	

Field	Type	Null	Key	Default	Extra
loannumber	int	NO	PRI	NULL	
branchname	varchar(50)	YES	MUL	NULL	
amount	int	YES		NULL	

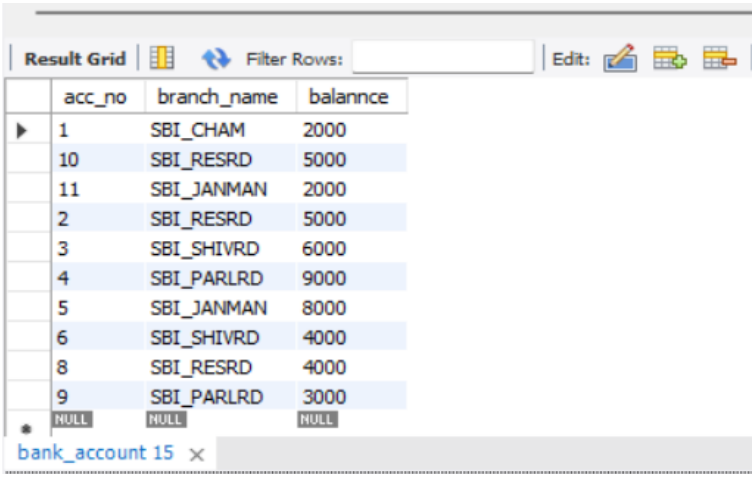
INSERTION OF VALUES

insert into branch values("SBI_CHAM", "BANGALORE", 50000);
insert into branch values("SBI_RESRD", "BANGALORE", 10000);
insert into branch values("SBI_SHIVRD", "BOMBAY", 20000);
insert into branch values("SBI_PARLRD", "DELHI", 10000);
insert into branch values("SBI_JANMAN", "DELHI", 10000);
select * from branch;



insert into bank_account values("1", "SBI_CHAM", 2000);
insert into bank_account values("2", "SBI_RESRD", 5000);

```
insert into bank_account values("3", "SBI_SHIVRD", 6000);
insert into bank_account values("4", "SBI_PARLRD", 9000);
insert into bank_account values("5","SBI_JANMAN", 8000);
insert into bank_account values("6", "SBI_SHIVRD", 4000);
 insert into bank_account values("8", "SBI_RESRD", 4000);
insert into bank_account values("9", "SBI_PARLRD", 3000);
 insert into bank_account values("10", "SBI_RESRD",5000);
insert into bank_account values("11", "SBI_JANMAN", 2000);
select * from bank_account;
```

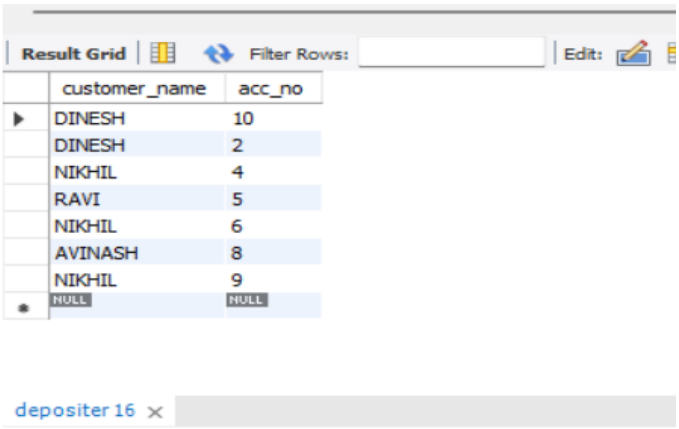


Result Grid | Filter Rows: | Edit:

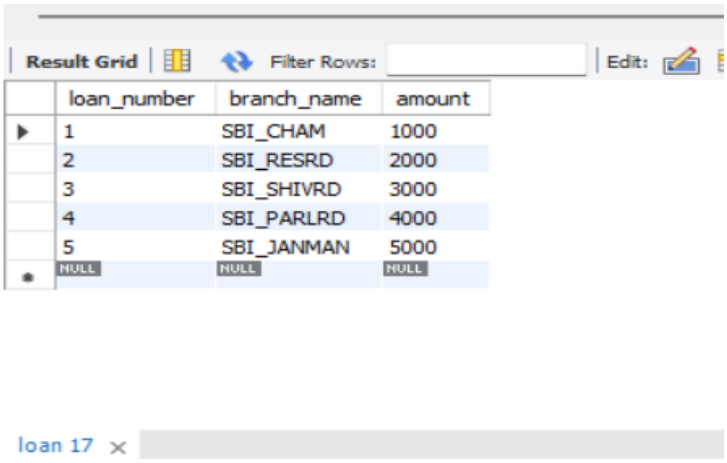
acc_no	branch_name	balannce
1	SBI_CHAM	2000
10	SBI_RESRD	5000
11	SBI_JANMAN	2000
2	SBI_RESRD	5000
3	SBI_SHIVRD	6000
4	SBI_PARLRD	9000
5	SBI_JANMAN	8000
6	SBI_SHIVRD	4000
8	SBI_RESRD	4000
9	SBI_PARLRD	3000
NULL	NULL	NULL

bank_account 15

```
insert into depositer values("DINESH", 2);
insert into depositer values("NIKHIL", 4);
 insert into depositer values("RAVI", 5);
insert into depositer values("AVINASH", 8);
insert into depositer values("NIKHIL", 9);
insert into depositer values("DINESH", 10);
insert into depositer values("NIKHIL", 6);
select * from depositer;
```

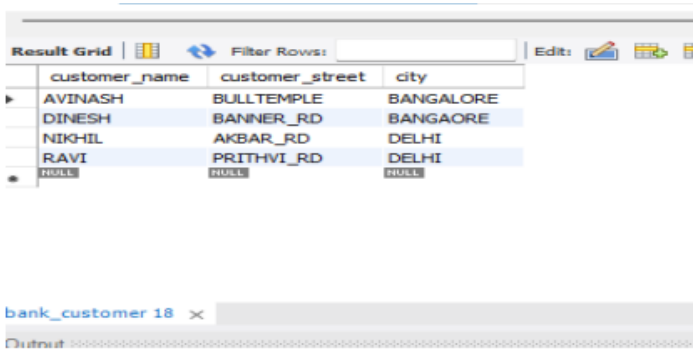



```
 insert into loan values(1, "SBI_CHAM" ,1000);
 insert into loan values(2, "SBI_RESRD" ,2000);
insert into loan values(3, "SBI_SHIVRD" ,3000);
insert into loan values(4, "SBI_PARLRD" ,4000);
 insert into loan values(5, "SBI_JANMAN" ,5000);
select * from loan;
```




```
 insert into bank_customer values("AVINASH","BULLTEMPLE",
"BANGALORE");
insert into bank_customer values("DINESH", "BANNER_RD",
"BANGALORE");
```

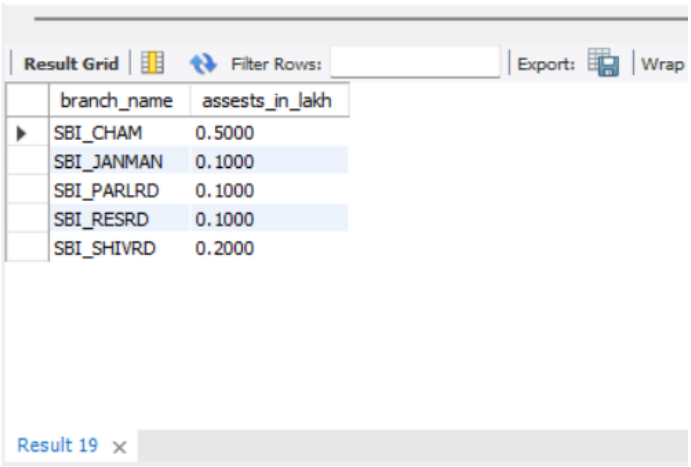
insert into bank_customer values("AVINASH", "NATCLG_RD", "BANGALORE");
insert into bank_customer values("NIKHIL", "AKBAR_RD","DELHI");
insert into bank_customer values("RAVI", "PRITHVI_RD", "DELHI");
select * from bank_customer;



QUERIES

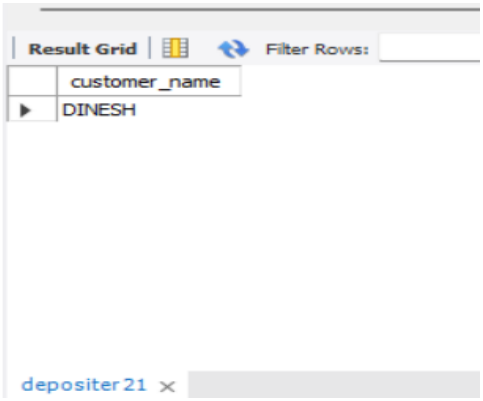
1.Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.

select branch_name, assests/100000 as assests_in_lakh from branch;



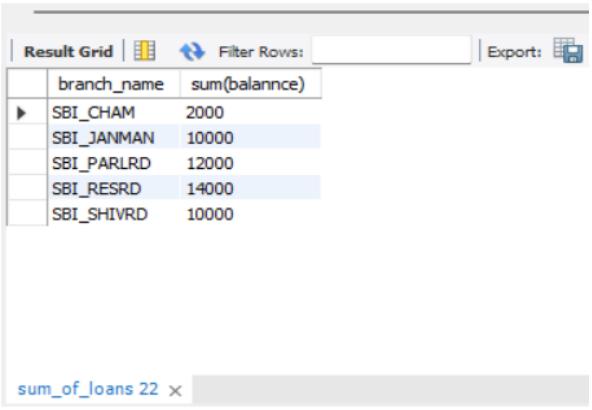
2.Find all the customers who have at least two accounts at the same branch (ex.SBI_ResidencyRoad).

select customer_name from depositer where acc_no IN (select acc_no from bank_account where branch_name="SBI_RESRD" group by customer_name having count(acc_no)>=2);



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

create view sum_of_loans as select branch_name, sum(balannce) from bank_account group by branch_name; select * from sum_of_loans



More Queries on BANK DATABASE

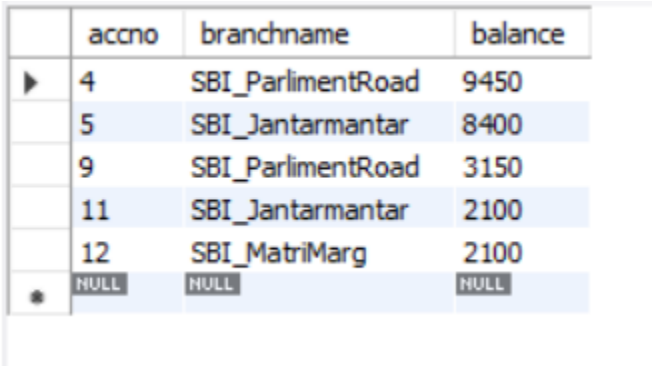
- Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).
- Find all customers who have a loan at the bank but do not have an account.
- Find all customers who have both an account and a loan at the Bangalore branch
- Find the names of all branches that 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%

INSERTION OF VALUES

```
insert into bankaccount values(12,"SBI_MatriMarg",2000);
insert into branch values("SBI_MatriMarg","Delhi",200000);
 insert into depositer values("Nikil",12)
```

branchname	branchcity	assets
SBI_Jantarmantar	Delhi	20000
SBI_MatriMarg	Delhi	200000
SBI_ParlimentRoad	Delhi	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

;



	accno	branchname	balance
▶	4	SBI_ParlimentRoad	9450
	5	SBI_Jantarmantar	8400
	9	SBI_ParlimentRoad	3150
	11	SBI_Jantarmantar	2100
	12	SBI_MatriMarg	2100
*	NULL	NULL	NULL

```
create table borrower(
customername varchar(50),
loannumber int,
foreign key(customername) references bankcustomer(customername),
foreign key(loannumber) references loan(loannumber));
insert into borrower
values("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.customername from branch b, depositer d, bankaccount ba where
b.branchcity='Delhi' and d.accno=ba.accno and
b.branchname=ba.branchname group by d.customername having
count(customername)>1;
```

	customername
▶	Nikil

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

select distinct b.customername from borrower b, depositer d where b.Customername not in(select d.customername from loan l,depositer d, borrower b where l.loannumber=b.loannumber and d.customername=b.customername);

	customername
▶	Mohan

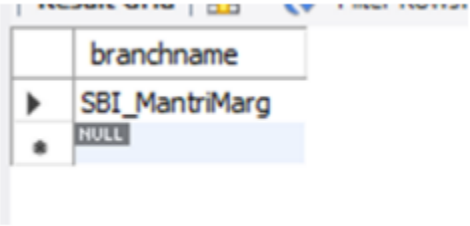
• Find all customers who have both an account and a loan at the Bangalore branch

select distinct d.customername from depositer d where d.customername in(select d.customername from branch br,depositer d, bankaccount ba where br.branchcity="Banglore" and br.branchname=ba.branchname and ba.accno=d.accno and d.customername in(select customername from borrower));

	customername
▶	Avinash
	Dinesh

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

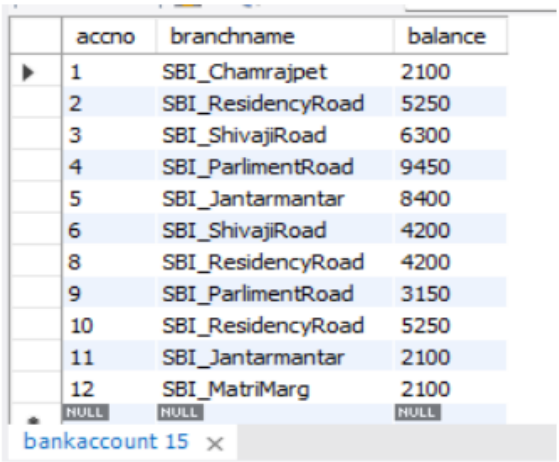
select b.branchname from branch b where b.assets> all (select sum(b.assets) from branch b where b.branchcity='Banglore');



branchname
SBI_MantriMarg
NULL

• Update the Balance of all accounts by 5% update bankaccount set balance=(balance+(balance*0.05));

select * from bankaccount;

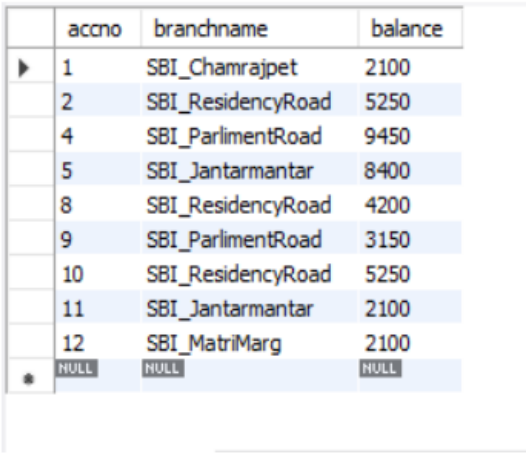


accno	branchname	balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
3	SBI_ShivajiRoad	6300
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
6	SBI_ShivajiRoad	4200
8	SBI_ResidencyRoad	4200
9	SBI_ParlimentRoad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
12	SBI_MatriMarg	2100
NULL	NULL	NULL

bankaccount 15

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

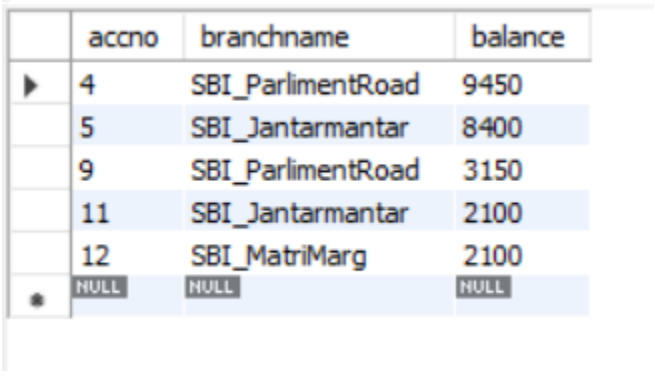
delete ba.* from bankaccount ba, branch b where branchcity='Bombay' and ba.branchname=b.branchname; select * from bankaccount;



accno	branchname	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_MatriMarg	2100
NULL	NULL	NULL

• Demonstrate how to delete branches located in banglore

delete b.*,ba.* from branch b, bankaccount ba,loan l Where b.branchcity=”Banglore” and b.branchname=ba.branchname And l.branchname=ba.branchname; select * from branch; select * from bankaccount



accno	branchname	balance
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
9	SBI_ParlimentRoad	3150
11	SBI_Jantarmantar	2100
12	SBI_MatriMarg	2100
NULL	NULL	NULL

EMPLOYEE DATABASE

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

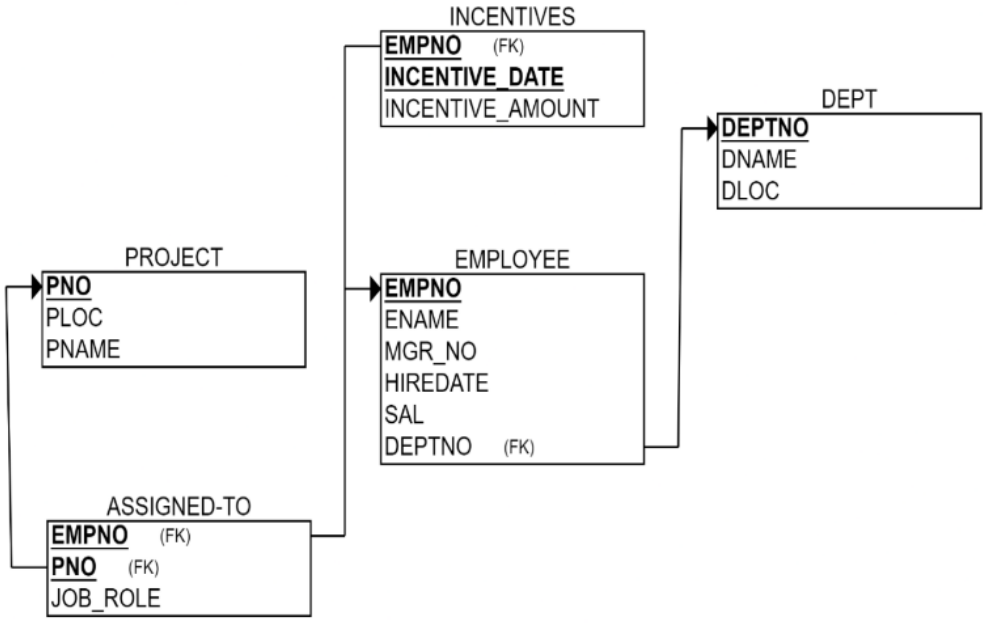
2. Enter greater than five tuples for each table.

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

4. Get Employee ID’s of those employees who didn’t receive incentives

5. 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

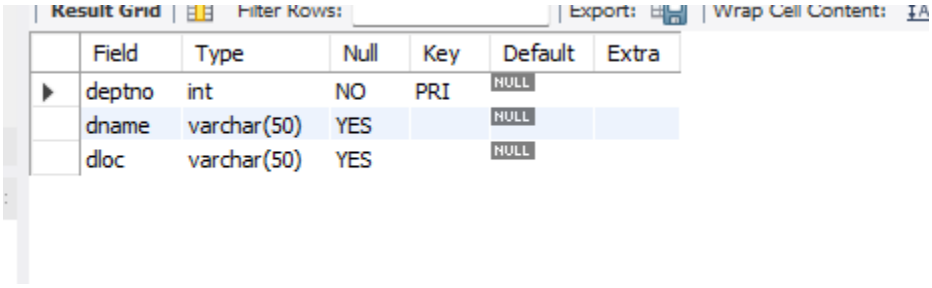


CREATION OF DATABASE

```
create database 1BM21CS050_EMPLOYEE;
use 1BM21CS050_EMPLOYEE;
 create table Dept(
deptno int, dname varchar (20),
dloc varchar(20),
PRIMARY KEY (deptno) );

create table Employee(
empno int, ename varchar(20),
mgr_no int,
hiredate date,
salary int,
```

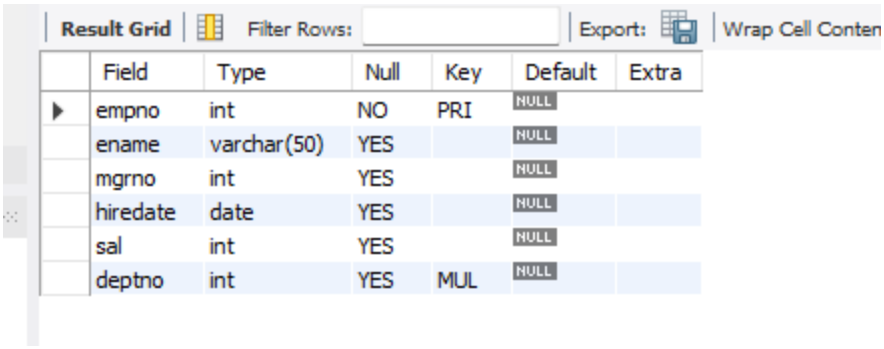
```
deptno int,
PRIMARY KEY (empno),
FOREIGN KEY (deptno) REFERENCES Dept(deptno)
);

 create table Project(
pno int, ploc varchar(30),
pname varchar(20),
PRIMARY KEY (pno)
 ); create table Incentives(
empno int,
Incentives_date int,
 Incentives_amount int,
PRIMARY KEY (Incentives_date),
FOREIGN KEY (empno) references Employee (empno)
);

create table Assigned_to(
empno int,
pno int,
 job_role varchar(20),
PRIMARY KEY (empno, pno),
FOREIGN KEY (empno) REFERENCES Employee(empno),
FOREIGN KEY (pno) REFERENCES Project(pno)
);
```

STRUCTURE OF TABLE

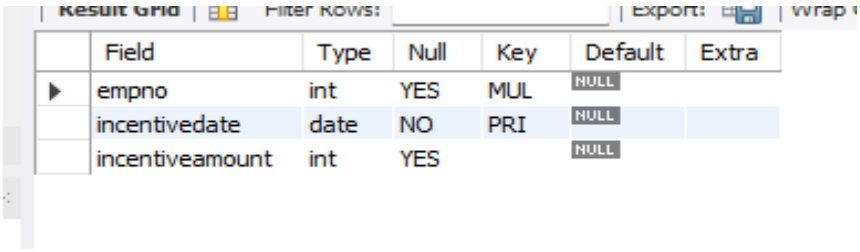


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

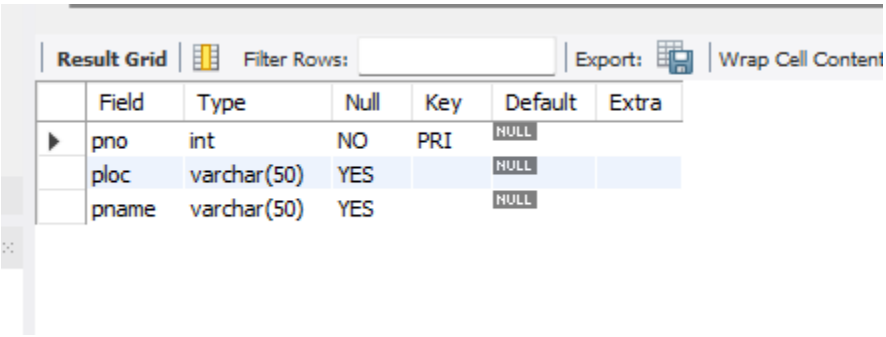
Field	Type	Null	Key	Default	Extra
deptno	int	NO	PRI	NULL	
dname	varchar(50)	YES		NULL	
dloc	varchar(50)	YES		NULL	



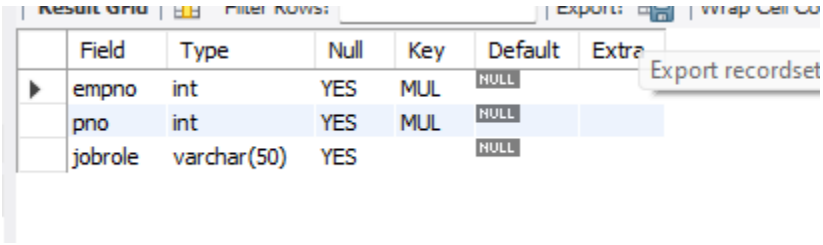
Field	Type	Null	Key	Default	Extra
empno	int	NO	PRI	NULL	
ename	varchar(50)	YES		NULL	
mgrno	int	YES		NULL	
hiredate	date	YES		NULL	
sal	int	YES		NULL	
deptno	int	YES	MUL	NULL	



Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
incentivedate	date	NO	PRI	NULL	
incentiveamount	int	YES		NULL	



Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
ploc	varchar(50)	YES		NULL	
pname	varchar(50)	YES		NULL	

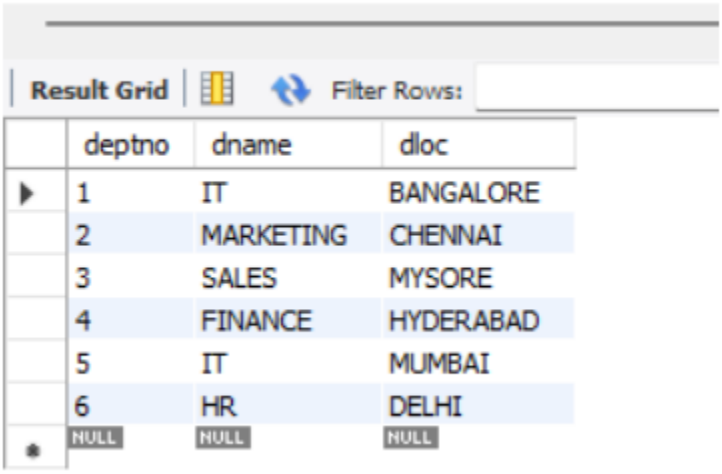


Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
pno	int	YES	MUL	NULL	
jobrole	varchar(50)	YES		NULL	

INSERTION OF VALUES

insert into Dept values(1, "IT", "BANGALORE");
insert into Dept values(2, "MARKETING", "CHENNAI");

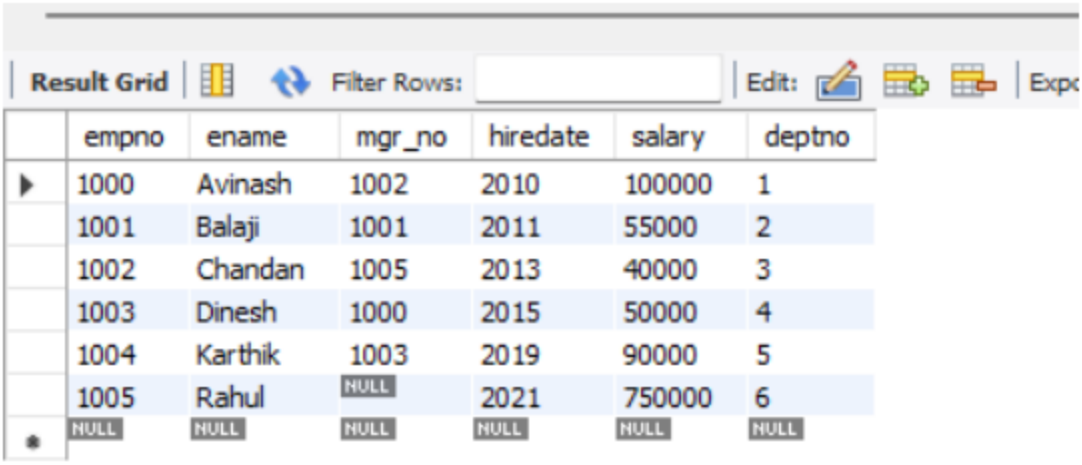
```
insert into Dept values(3, "SALES", "MYSORE");
insert into Dept values(4, "FINANCE", "HYDERABAD");
insert into Dept values(5, "IT", "MUMBAI");
insert into Dept values(6, "HR", "DELHI");
select * Dept;
```



Result Grid | Filter Rows:

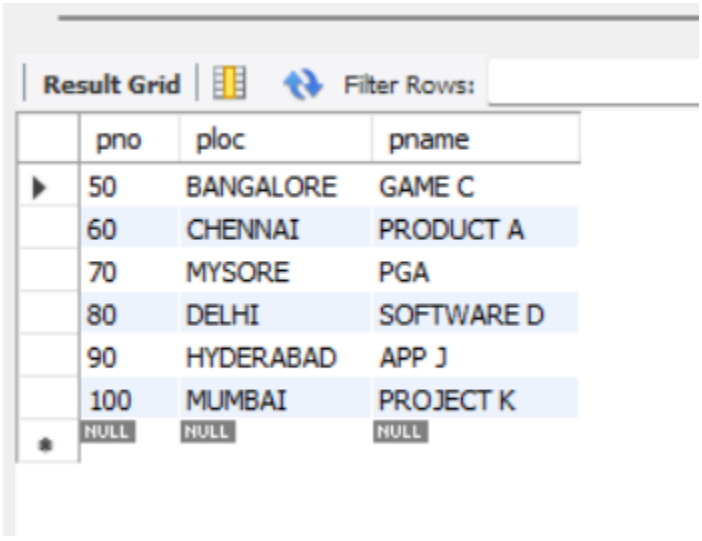
deptno	dname	dloc
1	IT	BANGALORE
2	MARKETING	CHENNAI
3	SALES	MYSORE
4	FINANCE	HYDERABAD
5	IT	MUMBAI
6	HR	DELHI
NULL	NULL	NULL

```
insert into Employee values(1000, "Avinash", 1002, 2010, 100000, 1);
insert into Employee values(1001, "Balaji", 1001, 2011, 55000, 2);
insert into Employee values(1002, "Chandan", 1005, 2013, 40000, 3);
insert into Employee values(1003, "Dinesh", 1000, 2015, 50000, 4);
insert into Employee values(1004, "Karthik", 1003, 2019, 90000, 5);
insert into Employee values(1005, "Rahul", NULL, 2021, 750000, 6);
select * from Employee;
```



empno	ename	mgr_no	hiredate	salary	deptno
1000	Avinash	1002	2010	100000	1
1001	Balaji	1001	2011	55000	2
1002	Chandan	1005	2013	40000	3
1003	Dinesh	1000	2015	50000	4
1004	Karthik	1003	2019	90000	5
1005	Rahul	NULL	2021	750000	6
NULL	NULL	NULL	NULL	NULL	NULL

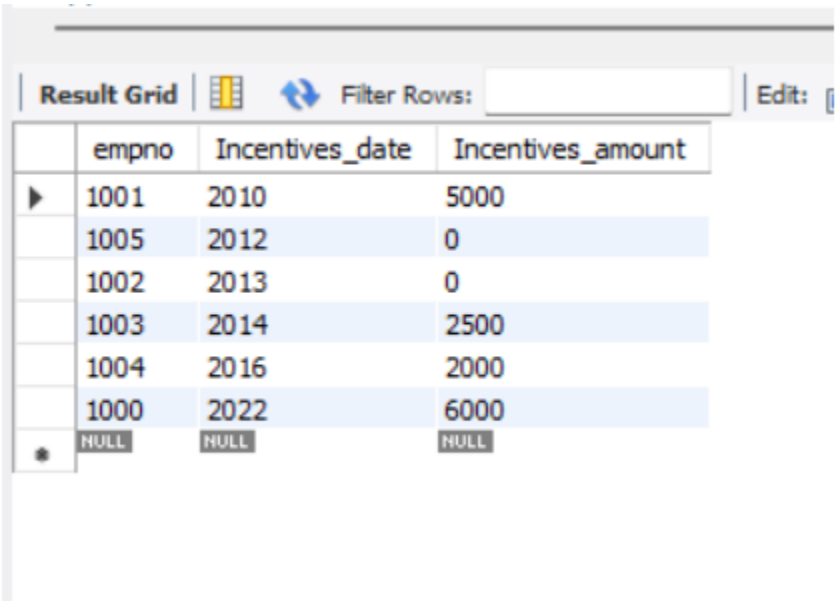
```
insert into Project values(50, "BANGALORE", "GAME C");
insert into Project values(60, "CHENNAI", "PRODUCT A");
insert into Project values(70, "MYSORE", "PGA");
insert into Project values(80, "DELHI", "SOFTWARE D");
insert into Project values(90, "HYDERABAD", "APP J");
 insert into Project values(100, "MUMBAI", "PROJECT K");
select * from Project;
```



pno	ploc	pname
50	BANGALORE	GAME C
60	CHENNAI	PRODUCT A
70	MYSORE	PGA
80	DELHI	SOFTWARE D
90	HYDERABAD	APP J
100	MUMBAI	PROJECT K
NULL	NULL	NULL

```
insert into Incentives values(1001, 2010, 5000);
insert into Incentives values(1002, 2013, 0);
insert into Incentives values(1003, 2014, 2500);
```

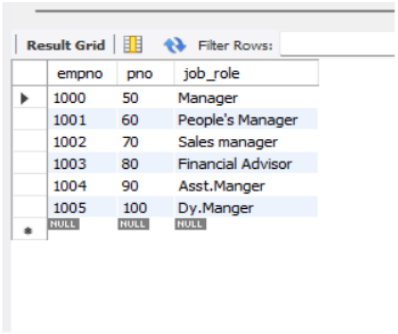
```
insert into Incentives values(1004, 2016, 2000);
 insert into Incentives values(1005, 2012, 0);
insert into Incentives values(1000, 2022, 6000);
 select * from Incentives;
```



Result Grid | Filter Rows: | Edit:

empno	Incentives_date	Incentives_amount
1001	2010	5000
1005	2012	0
1002	2013	0
1003	2014	2500
1004	2016	2000
1000	2022	6000
NULL	NULL	NULL

```
insert into Assigned_to values(1000, 50, "Manager");
insert into Assigned_to values(1001, 60, "People's Manager");
insert into Assigned_to values(1002, 70, "Sales manager");
insert into Assigned_to values(1003, 80, "Financial Advisor");
insert into Assigned_to values(1004, 90, "Asst.Manger");
insert into Assigned_to values(1005, 100, "Dy.Manger");
select * from Assigned_to;
```



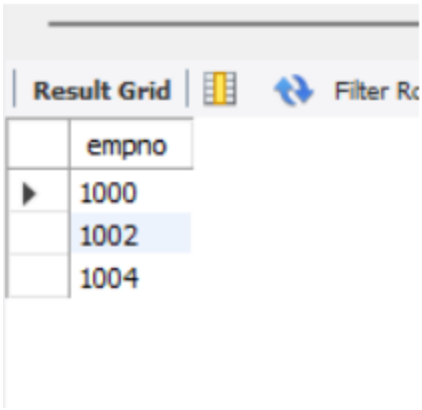
Result Grid | Filter Rows:

empno	pno	job_role
1000	50	Manager
1001	60	People's Manager
1002	70	Sales manager
1003	80	Financial Advisor
1004	90	Asst.Manger
1005	100	Dy.Manger
NULL	NULL	NULL

QUERIES

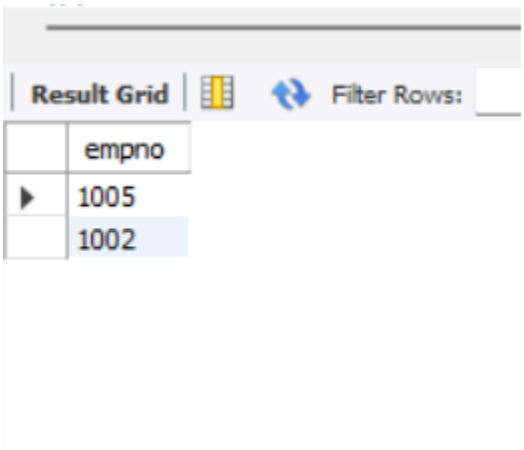
1. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru

select empno from Assigned_to where pno= ANY(select pno from Project where ploc="HYDERABAD" OR ploc = "BANGALORE" OR ploc="MYSORE");



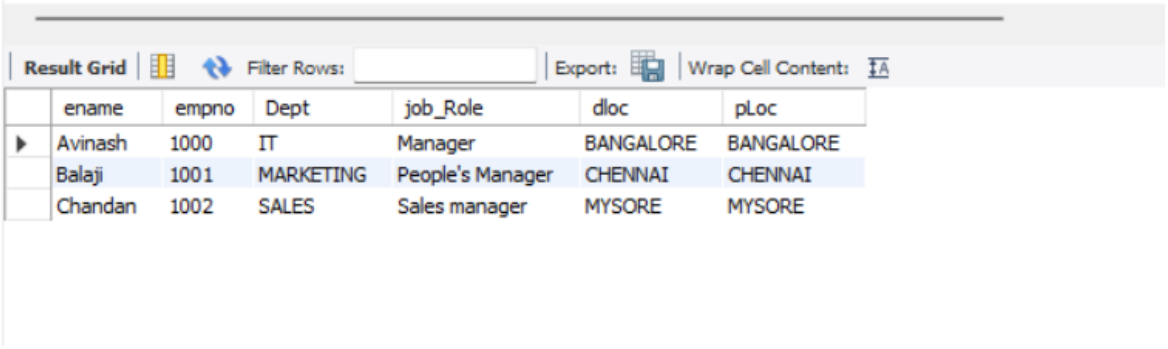
2. Get Employee ID’s of those employees who didn’t receive incentives

select e.empno from employee e where e.empno NOT IN (select i.empno from Incentives i);



3. 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.ename ename, e.empno empno, d.Dname Dept, a.job_role job_Role, d.Dloc dloc, p.ploc pLoc from Project p, Dept d, Employee e, Assigned_to a where e.empno=a.empno and p.pno=a.pno and e.deptno=d.deptno and p.ploc=d.dloc;



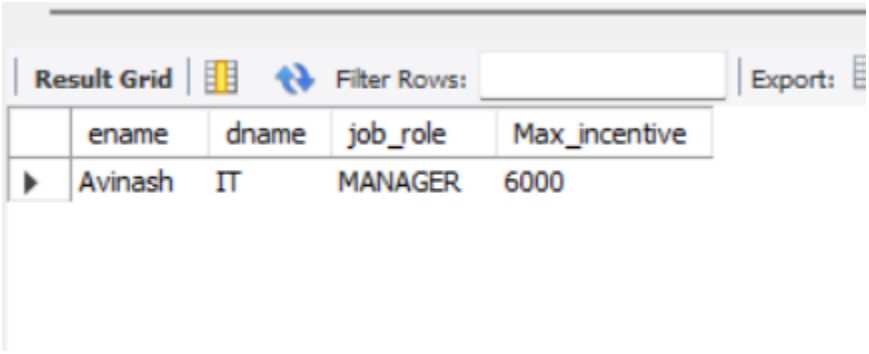
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

ename	empno	Dept	job_Role	dloc	pLoc
Avinash	1000	IT	Manager	BANGALORE	BANGALORE
Balaji	1001	MARKETING	People's Manager	CHENNAI	CHENNAI
Chandan	1002	SALES	Sales manager	MYSORE	MYSORE

SPOT QUERY

Find the employee name, dept name and job_role of an employee who received max incentive in year 2021

select e.ename, d.dname, a.job_role, MAX(i.Incentives_amount) MAX_Incentive from Employee e, Dept d, Incentives i, Assigned_to a where Incentives_date between '2016' and '2022';



Result Grid | Filter Rows: | Export:

ename	dname	job_role	Max_incentive
Avinash	IT	MANAGER	6000

More queries on EMPLOYEE DATABASE

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

2. Enter greater than five tuples for each table.

3. List the name of the managers with the maximum employees

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

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

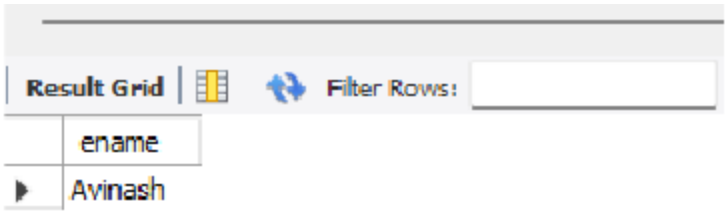
6. Find the employee details who got second maximum incentive in January 2019.

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

QUERIES

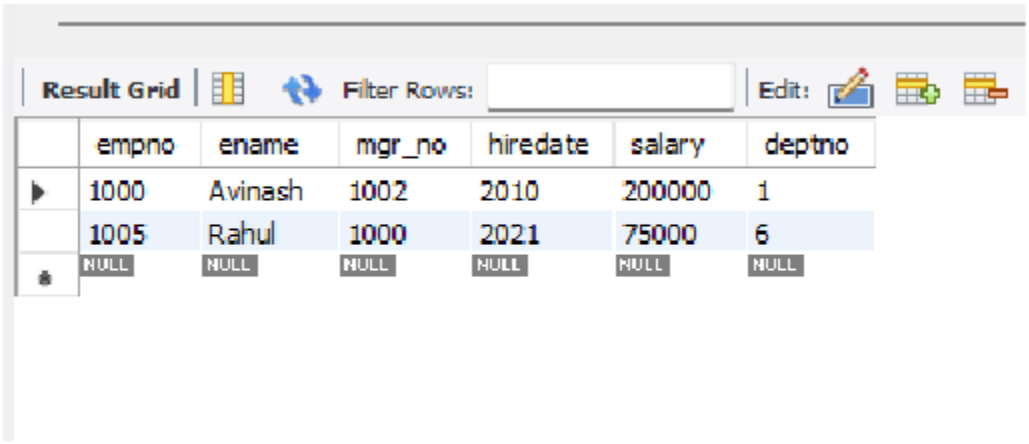
1. List the name of the managers with the maximum employees

```
select e.ename from employee e,Employee f where e.empno =
f.mgr_no group by e.empno having count(*)=(select max(mycount)
from (select count(*) mycount from Employee group by mgr_no) a);
```



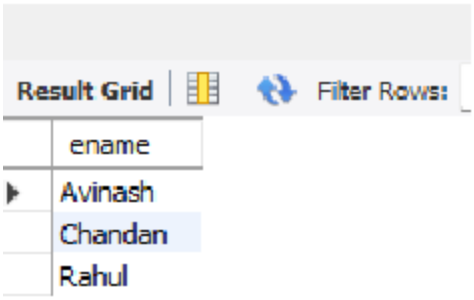
2. Display those managers name whose salary is more than average salary of his employee.

select * from employee m where m.empno in (select mgr_no from employee) and m.salary>(select avg(n.salary) from Employee n where n.mgr_no=m.empno);



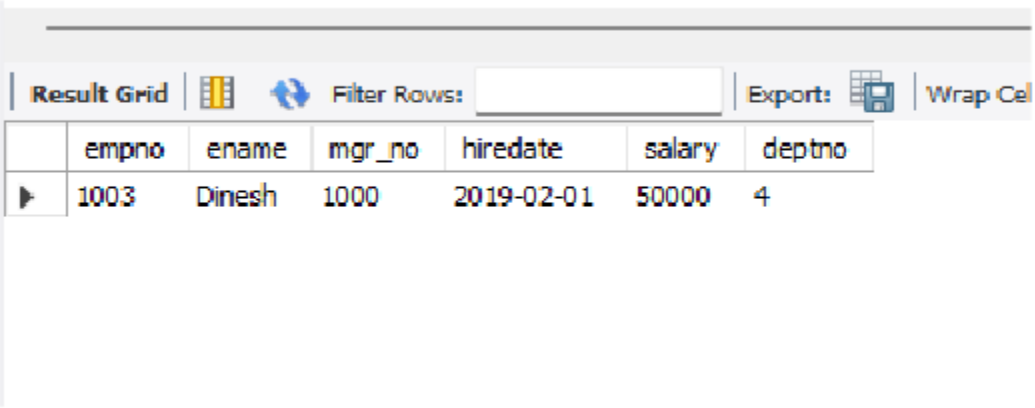
3.Find the name of second top level managers of each department

select ename from employee where empno in(select distinct mgr_no from employee where empno in (select distinct mgr_no from employee where empno in(select distinct mgr_no from employee)));



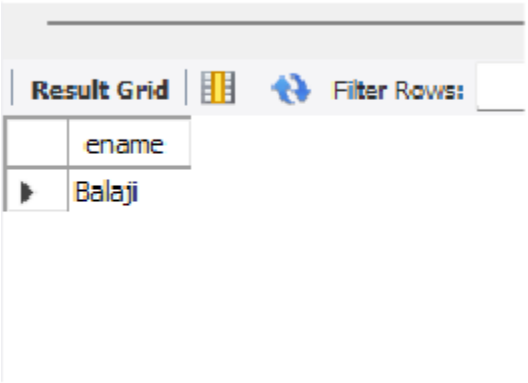
4.Find the employee details who got second maximum incentive in January 2019.

select * from Employee where empno= (select iii.empno from incentives iii where iii.incentives_amount=(select max(ii.incentives_amount) from incentives ii where ii.incentives_amount<(select max(i.incentives_amount) from incentives i where i.incentives_date between "2019-01-01" and "2019-12-31") and incentives_date between "2019-01-01" and "2019-12-31"));



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

select e.ename from Employee e where e.Deptno=(select Deptno from Employee where e.mgr_no=empno);



SUPPLIER DATABASE

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.

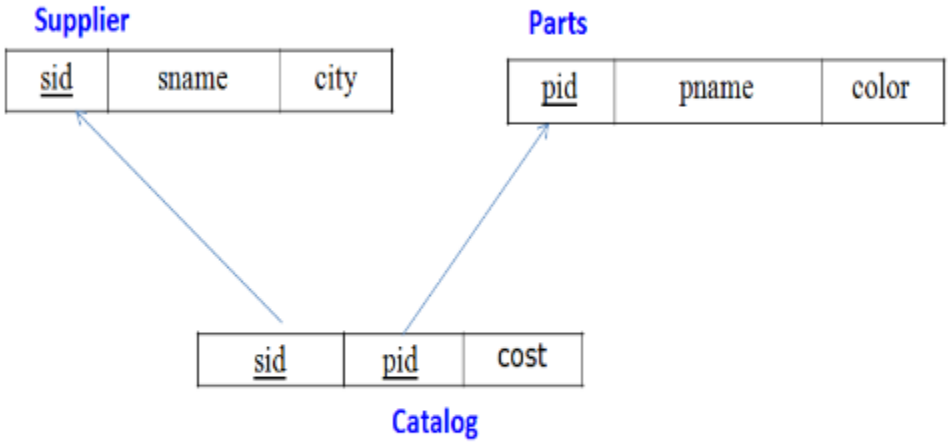
5. Find the snames of suppliers who supply every red part.

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.

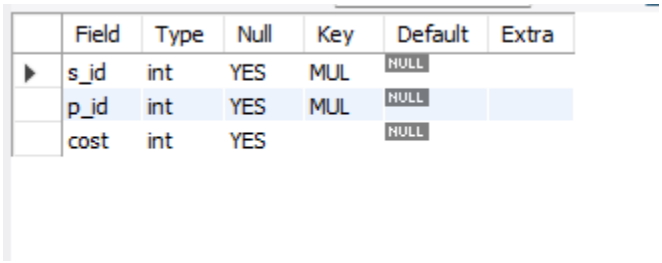
SCHEMA DIAGRAM



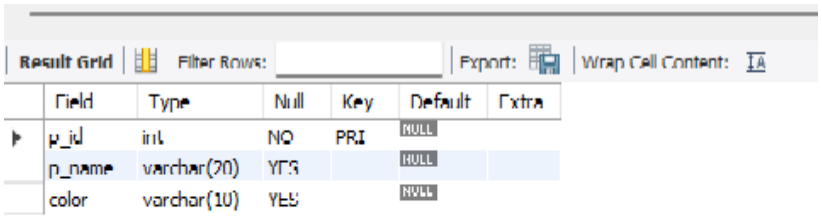
CREATION OF TABLES

```
create table Supplier(
s_id int,
s_name varchar(20),
city varchar(15),
 PRIMARY KEY (s_id) );

 create table Parts(
p_id int,
p_name varchar(20),
color varchar(10),
PRIMARY KEY (p_id) );

create table Catalog(
```

```
s_id int,
 p_id int,
cost int,
FOREIGN KEY (s_id) REFERENCES Supplier(s_id),
FOREIGN KEY (p_id) REFERENCES Parts(p_id)
);
```

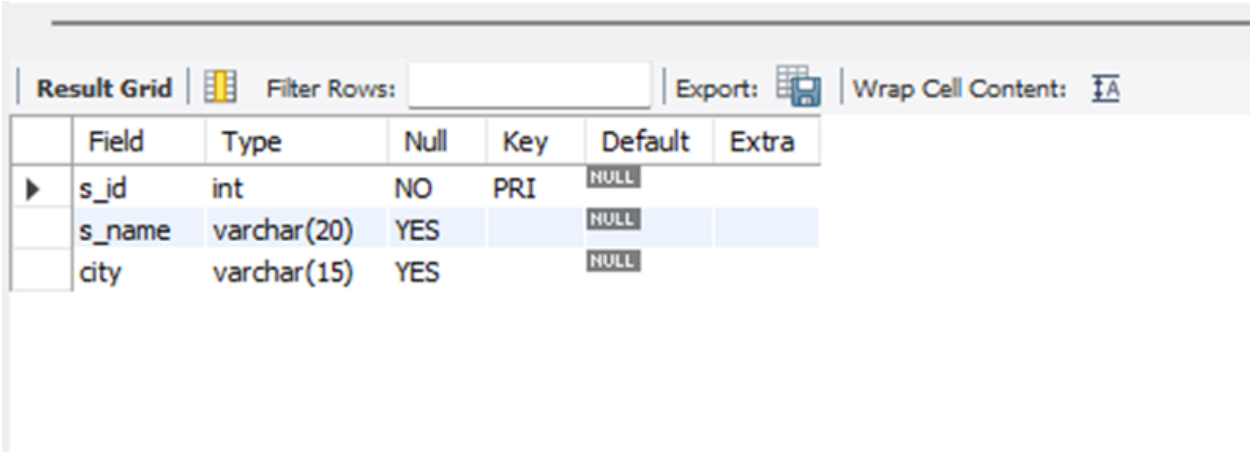


Field	Type	Null	Key	Default	Extra
s_id	int	YES	MUL	NULL	
p_id	int	YES	MUL	NULL	
cost	int	YES		NULL	



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

Field	Type	Null	Key	Default	Extra
p_id	int	NO	PRI	NULL	
p_name	varchar(20)	YES		NULL	
color	varchar(10)	YES		NULL	

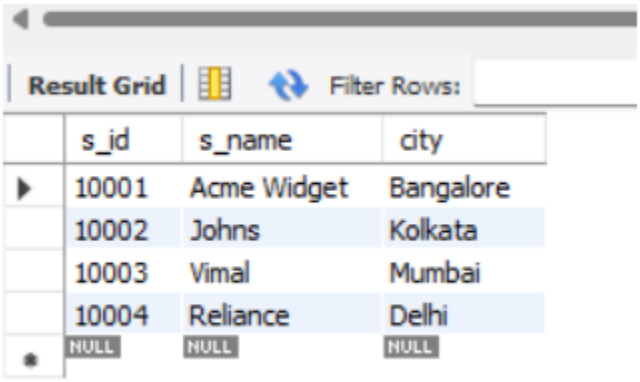


Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
s_id	int	NO	PRI	NULL	
s_name	varchar(20)	YES		NULL	
city	varchar(15)	YES		NULL	

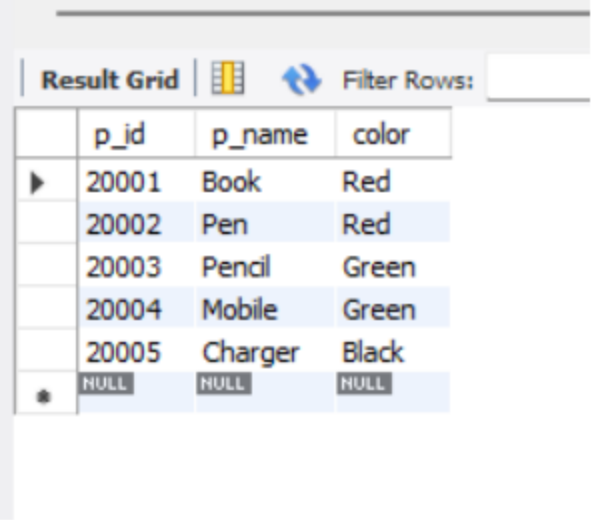
INSERTION OF VALUES

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



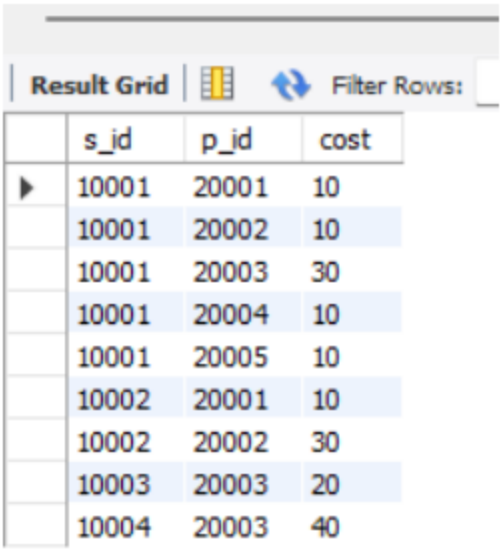
s_id	s_name	city
10001	Acme Widget	Bangalore
10002	Johns	Kolkata
10003	Vimal	Mumbai
10004	Reliance	Delhi
NULL	NULL	NULL

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



p_id	p_name	color
20001	Book	Red
20002	Pen	Red
20003	Pencil	Green
20004	Mobile	Green
20005	Charger	Black
NULL	NULL	NULL

```
insert into Catalog values(10001, 20001, 10);
insert into Catalog values(10001, 20002, 10);
insert into Catalog values(10001, 20003, 30);
insert into Catalog values(10001, 20004, 10);
insert into Catalog values(10001, 20005, 10);
insert into Catalog values(10002, 20001, 10);
insert into Catalog values(10002, 20002, 30);
insert into Catalog values(10003, 20003, 20);
insert into Catalog values(10004, 20003, 40);
select * from Catalog;
```



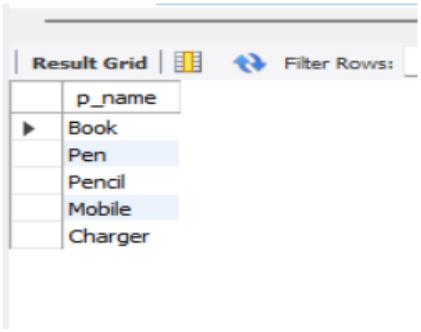
Result Grid | Filter Rows:

s_id	p_id	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10001	20005	10
10002	20001	10
10002	20002	30
10003	20003	20
10004	20003	40

QUERIES

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

select p_name from Parts where p_id IN (select p_id from Catalog);

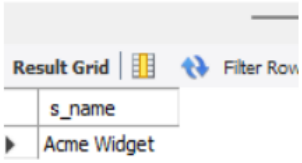


Result Grid | Filter Rows:

p_name
Book
Pen
Pencil
Mobile
Charger

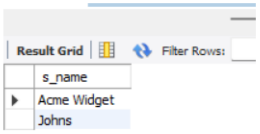
2. Find the snames of suppliers who supply every part.

select s_name from Supplier where s_id in(select s_id from Catalog group by s_id having count(s_id)=(select count(p_id) from Parts));



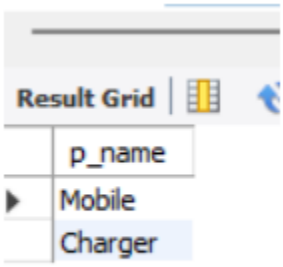
3. Find the snames of suppliers who supply every red part.

select s_name from Supplier where s_id in(select s_id from Catalog where p_id in(select p_id from parts where color='Red'));



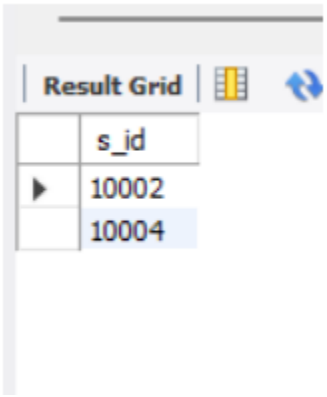
4. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.

select p_name from parts where p_id in(select p_id from catalog where s_id in(select s_id from supplier where s_name='Acme widget')) and p_id not in(select p_id from catalog where s_id in(select s_id from supplier where s_name!='Acme widget'));



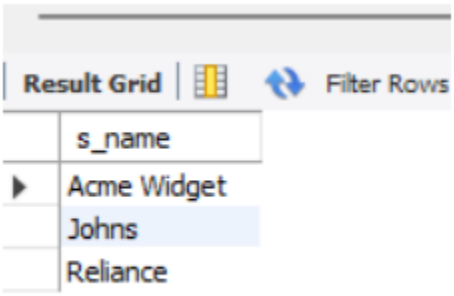
5. Find the sids of suppliers who charge more for some part than the average cost of that part (averaged over all the suppliers who supply that part).

select c.s_id from catalog c where c.cost >(select avg(cc.cost) from catalog cc where c.p_id=cc.p_id group by cc.p_id);



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

select sname from supplier where sid in(select sid from catalog where cost in(select max(cost) from catalog group by pid));



FLIGHT DATABASE

FLIGHTS(flno: integer, from: string, to: string, distance: integer,
departs: time, arrives: time,
price: integer)

AIRCRAFT(aid: integer, aname: string, cruising_range: integer)

CERTIFIED(eid: integer, aid: integer)

EMPLOYEES(eid: integer, ename: string, salary: integer)

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

Create database table and insert appropriate data

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

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

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

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

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

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

CREATION OF TABLES

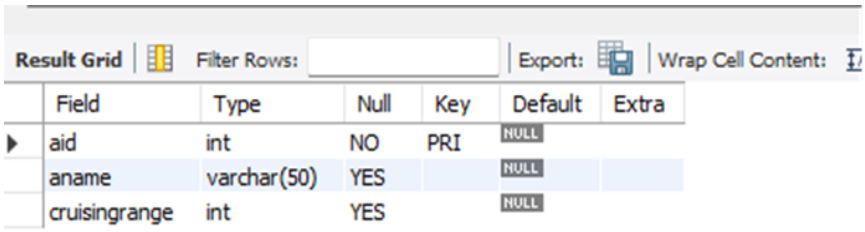
Creation of tables create table flights(
flno int,
ffrom varchar(50),
tto varchar(50),
distance int,
departs time,
arrives time,
price int,
primary key(flno)
);

create table aircraft(
aid int,
aname varchar(50),
cruisingrange int,
primary key(aid)
);

create table employee(
eid int,
ename varchar(50),
salary int,
primary key(eid)
);

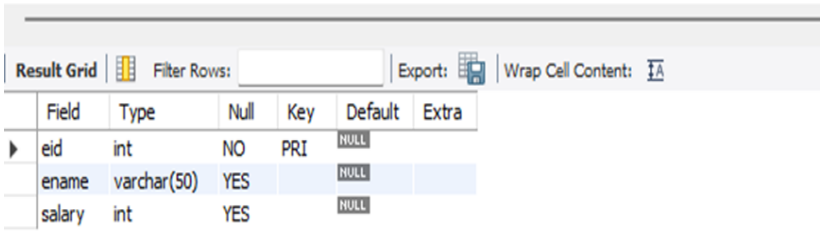
create table certified(
eid int,
aid int,
foreign key(aid) references aircraft(aid) on update cascade on delete cascade,
foreign key(eid) references employee(eid) on update cascade on delete cascade);

STRUCTURE OF TABLES



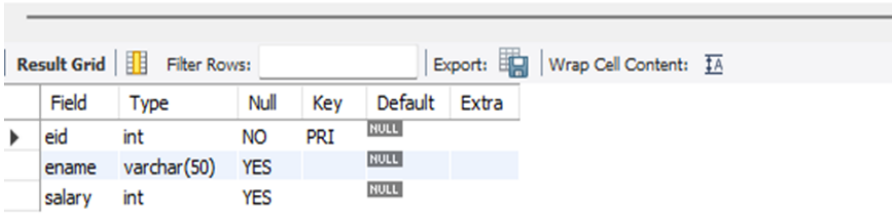
Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Field	Type	Null	Key	Default	Extra
aid	int	NO	PRI	NULL	
aname	varchar(50)	YES		NULL	
cruisingrange	int	YES		NULL	



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

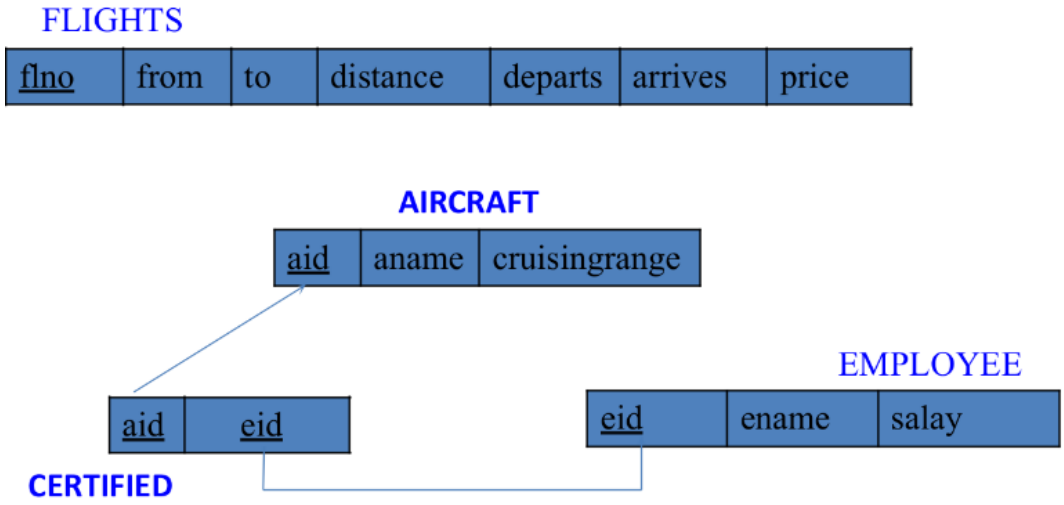
Field	Type	Null	Key	Default	Extra
eid	int	NO	PRI	NULL	
ename	varchar(50)	YES		NULL	
salary	int	YES		NULL	



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

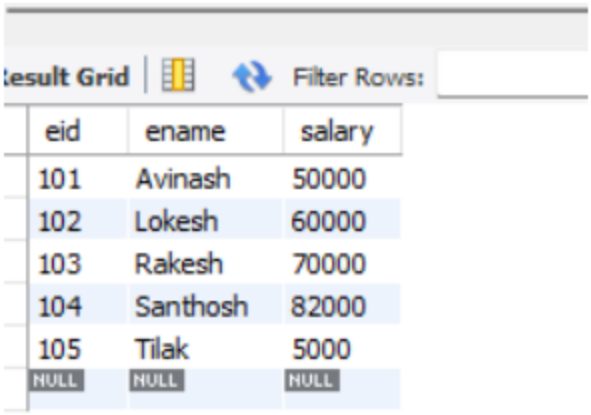
Field	Type	Null	Key	Default	Extra
eid	int	NO	PRI	NULL	
ename	varchar(50)	YES		NULL	
salary	int	YES		NULL	

SCHEMA DIAGRAM



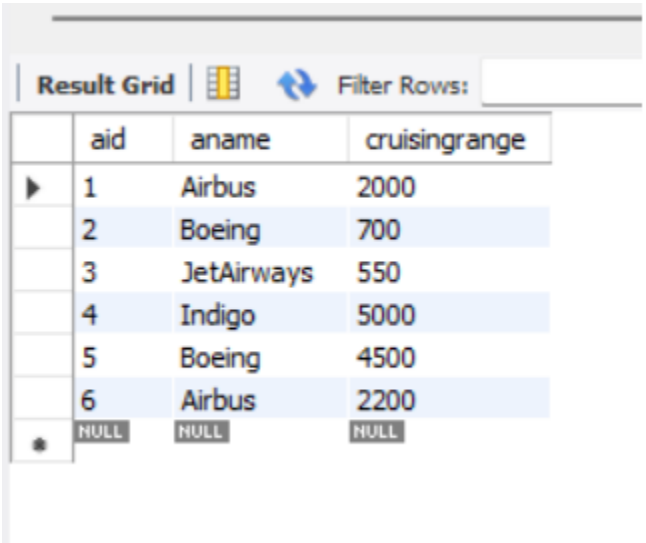
INSERTION OF VALUES

Insertion of values insert into employee values(101,'Avinash',50000);
insert into employee values(102,'Lokesh',60000);
insert into employee values(103,'Rakesh',70000);
insert into employee values(104,'Santhosh',82000);
insert into employee values(105,'Tilak',5000);
select * from employee;



eid	ename	salary
101	Avinash	50000
102	Lokesh	60000
103	Rakesh	70000
104	Santhosh	82000
105	Tilak	5000
NULL	NULL	NULL

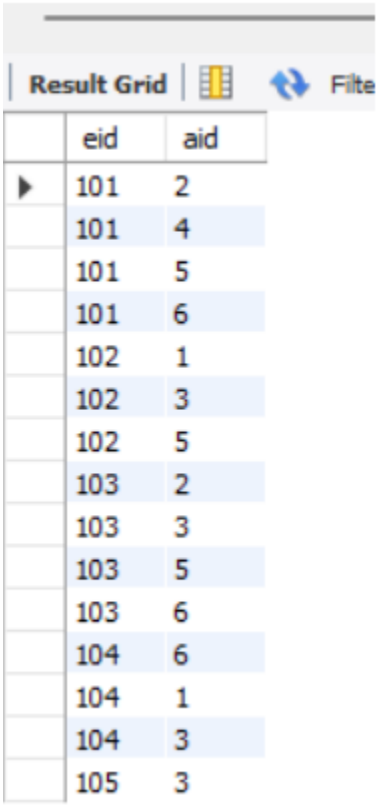
```
insert into aircraft values(1,'Airbus',2000);
insert into aircraft values(2,'Boeing',700);
insert into aircraft values(3,'JetAirways',550);
insert into aircraft values(4,'Indigo',5000);
insert into aircraft values(5,'Boeing',4500);
insert into aircraft values(6,'Airbus',2200);
select * from aircraft;
```



aid	aname	cruisingrange
1	Airbus	2000
2	Boeing	700
3	JetAirways	550
4	Indigo	5000
5	Boeing	4500
6	Airbus	2200
NULL	NULL	NULL

```
insert into certified values(101,2);
insert into certified values(101,4);
```

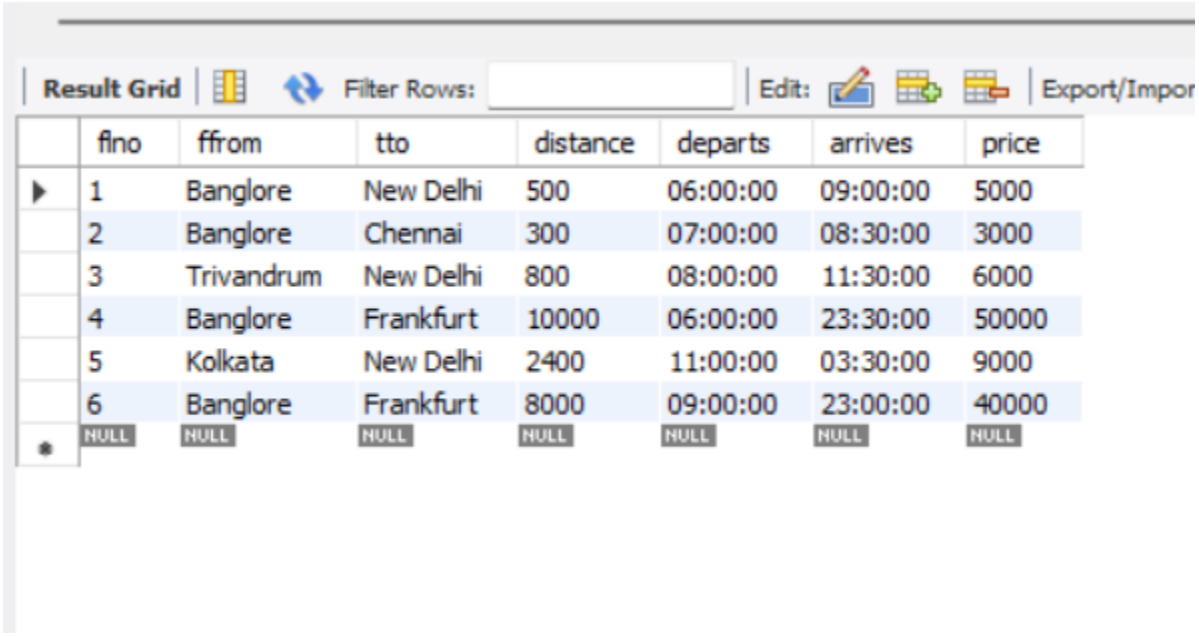
```
insert into certified values(101,5);
insert into certified values(101,6);
insert into certified values(102,1);
 insert into certified values(102,3);
insert into certified values(102,5);
insert into certified values(103,2);
insert into certified values(103,3);
insert into certified values(103,5);
insert into certified values(103,6);
 insert into certified values(104,6);
insert into certified values(104,1);
 insert into certified values(104,3);
insert into certified values(105,3);
select * from certified;
```



eid	aid
101	2
101	4
101	5
101	6
102	1
102	3
102	5
103	2
103	3
103	5
103	6
104	6
104	1
104	3
105	3

```
insert into flights
values(1,'Banglore','NewDelhi',500,'6:00','9:00',5000);
```

```
insert into flights values(2,'Banglore','Chennai',300,'7:00','8:30',3000);
insert into flights
values(3,'Trivandrum','NewDelhi',800,'8:00','11:30',6000);
insert into flights
values(4,'Banglore','Frankfurt',10000,'6:00','23:30',50000);
 insert into flights values(5,'Kolkata','New
Delhi',2400,'11:00','3:30',9000);
insert into flights
values(6,'Banglore','Frankfurt',8000,'9:00','23:00',40000);
select * from flights;
```



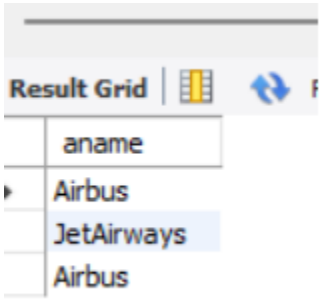
Result Grid | Filter Rows: | Edit: | Export/Impor

fino	ffrom	tto	distance	departs	arrives	price
1	Banglore	New Delhi	500	06:00:00	09:00:00	5000
2	Banglore	Chennai	300	07:00:00	08:30:00	3000
3	Trivandrum	New Delhi	800	08:00:00	11:30:00	6000
4	Banglore	Frankfurt	10000	06:00:00	23:30:00	50000
5	Kolkata	New Delhi	2400	11:00:00	03:30:00	9000
6	Banglore	Frankfurt	8000	09:00:00	23:00:00	40000
NULL	NULL	NULL	NULL	NULL	NULL	NULL

QUERIES

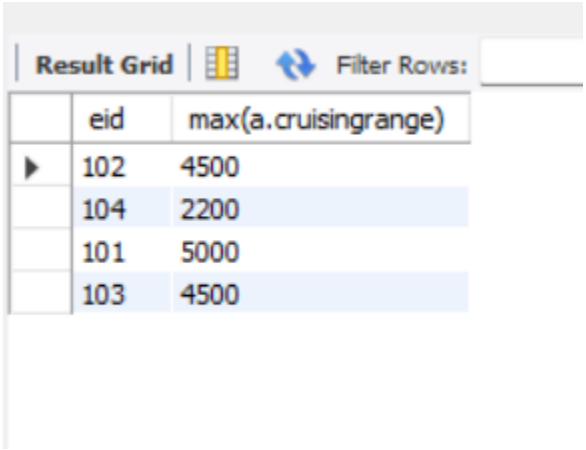
1. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

select a.aname from aircraft a where a.aid in(select c.aid from certified c where c.eid in(select e.eid from employee e where salary>80000));



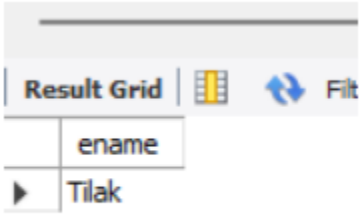
2. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.

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



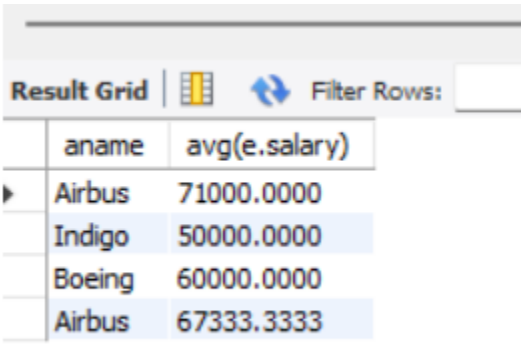
3. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

select e.ename from employee e where e.salary<(select min(f.price) from flights f where f.ffrom='Banglore' and f.tto='Frankfurt');



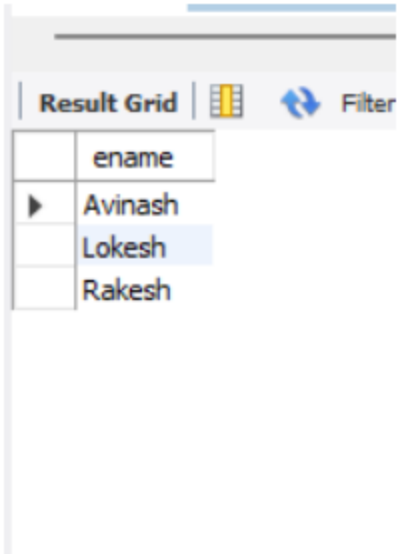
4. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the average salary of all pilots certified for this aircraft.

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



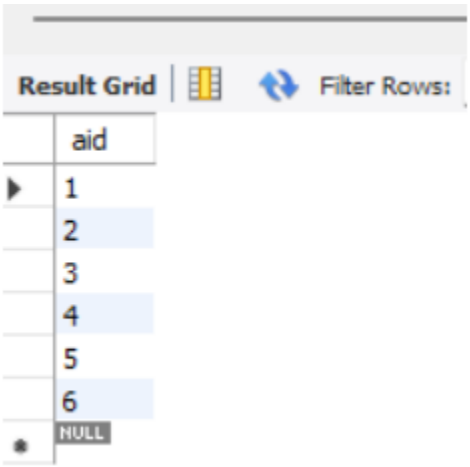
5. Find the names of pilots certified for some Boeing aircraft

select e.ename from employee e where e.eid in (select c.eid from certified c where c.aid in (select a.aid from aircraft a where a.aname='Boeing'));



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

select aid from aircraft where cruisingrange > ALL (select distance from flights where ffrom = 'Bangalore' AND tto='New Delhi');



NO SQL

STUDENT DATABASE

1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, Email-Id.

2. Insert appropriate values

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

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

5. Export the created table into local file system

6. Drop the table

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

QUERIES

1. Create a database “Student” with the following attributes Rollno, Age, ContactNo, Email-Id.

db.createCollection(“Student”);

2. Insert appropriate values

```
db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});

db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});

db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});

db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});

db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
```

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

```
db.Student.update({RollNo:10},{$set:{
email:"Abhinav@gmail.com"}})
```

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

```
db.Student.insert({RollNo:11,Age:22,Name:
"ABC",Cont:2276,email:"rea.de9@gmail.com"});
db.Student.update({RollNo:11,Name:"ABC"},{$se
t:{Name:"FEM"}})
```

5. Export the created table into local file system

```
mongoexport
mongodb+srv://NSQ:DEEPINISURESH@cluster0.mfnfeys.
```

mongodb.net/myDB --collection=Student --out C:\deepi\Downloads\output.json

6. Drop the table

db.Student.drop();

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

mongoimport mongodb+srv://NSQ:DEEPINISURESH@cluster0.mfnfeys.mongodb.net/myDB --collection=New_Student --type json --file C:\deepi\Downloads\output.json

```
Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-76fkrr-shard-0 [primary] NSQ> 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("63cfe9d1578f04a88f66485c") }
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ>

Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cfe9d1578f04a88f66485d") }
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ>

Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cfe9d2578f04a88f66485e") }
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ>

Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cfe9d2578f04a88f66485f") }
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ>

Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cfe9d4578f04a88f664860") }
}
```

```
Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cfe9d4578f04a88f664860") }
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.find()
[
  {
    _id: ObjectId("63cfe9d1578f04a88f66485c"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("63cfe9d1578f04a88f66485d"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("63cfe9d2578f04a88f66485e"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("63cfe9d2578f04a88f66485f"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("63cfe9d4578f04a88f664860"),
    RollNo: 10,
    Age: 23,
```

```
  },
  {
    _id: ObjectId("63cfe9d4578f04a88f664860"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.update({RollNo:10},{$set:{
... email:"Abhinav@gmail.com"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.insert({RollNo:11,Age:22,Name:
... "ABC",Cont:2276,email:"rea.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cfea89578f04a88f664861") }
}
```

```
Atlas atlas-76fkrr-shard-0 [primary] NSQ> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-76fkrr-shard-0 [primary] NSQ>
```

```
C:\Users\BMSCECSE>mongoexport mongodb+srv://antararc:Test1234@cluster0.mfnfeys.mongodb.net/myDB  --collection=Student --out C:\Users\BMSCECSE\Downloads\output.json
2023-01-12T15:15:56.383+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.mfnfeys.mongodb.net/myDB
2023-01-12T15:15:56.497+0530    exported 7 records
```

```
Atlas atlas-mdgaz1-shard-0 [primary] myDB> db.New_Student.find()
[
  {
    _id: ObjectId("63bfcfd156eba0e23c3a5c74"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("63bfcf9a56eba0e23c3a5c72"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("63bfcfe456eba0e23c3a5c75"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("63bfd41356eba0e23c3a5c77"),
    RollNo: 10,
    Age: 24,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
  {
    _id: ObjectId("63bfcff656eba0e23c3a5c76"),
    RollNo: 5,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  },
  {
    _id: ObjectId("63bfcfb456eba0e23c3a5c73"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("63bfd4de56eba0e23c3a5c78"),
    RollNo: 11,
    Age: 22,
    Name: 'FEM',
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  }
]
```

CUSTOMER DATABASE

1. Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type

2. Insert at least 5 values into the table

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

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

5. Export the created collection into local file system

6. Drop the table

7. Import a given csv dataset from local file system into mongodb Collection.

QUERIES

1. Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type

db.createCollection(“Customers”);

2. Insert at least 5 values into the table

db.Customers.insert({Cust_id:1,Acc_Bal:5000,Acc_Type:”Current”});

db.Customers.insert({Cust_id:2,Acc_Bal:2800,Acc_Type:”Current”});

db.Customers.insert({Cust_id:3,Acc_Bal:3000,Acc_Type:”Savings”});

db.Customers.insert({Cust_id:4,Acc_Bal:4500,Acc_Type:”Savings”});

db.Customers.insert({Cust_id:5,Acc_Bal:3000,Acc_Type:”Current”});

3. 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.find({Acc_Type:”Savings”,Acc_Bal:{$gt:1200}});

4. Determine Minimum and Maximum account balance for each Customer_id.

db.Customers.find().sort({Acc_Bal:+1}).limit(1);
db.Customers.find().sort({Acc_Bal:-1}).limit(1);

5. Export the created collection into local file system

6. Drop the table

db.Customers.drop();

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

```
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.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("63cff4ad2367415816e1a7ce") }
}
```

```
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.insert({Cust_id:2,Acc_Bal:2800,Acc_Type:"Current"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cff66d2367415816e1a7cf") }
}
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.insert({Cust_id:3,Acc_Bal:3000,Acc_Type:"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cff6a02367415816e1a7d0") }
}
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.insert({Cust_id:4,Acc_Bal:4500,Acc_Type:"Current"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cff6ca2367415816e1a7d1") }
}
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customer.insert({Cust_id:5,Acc_Bal:6000,Acc_Type:"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cff6ef2367415816e1a7d2") }
```

```
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.insert({Cust_id:5,Acc_Bal:3000,Acc_Type:"Savings"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("63cff7d22367415816e1a7d3") }
}
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.find();
[
  {
    _id: ObjectId("63cff4ad2367415816e1a7ce"),
    Cust_id: 1,
    Acc_Bal: 5000,
    Acc_Type: 'Savings'
  },
  {
    _id: ObjectId("63cff66d2367415816e1a7cf"),
    Cust_id: 2,
    Acc_Bal: 2800,
    Acc_Type: 'Current'
  },
  {
    _id: ObjectId("63cff6a02367415816e1a7d0"),
    Cust_id: 3,
    Acc_Bal: 3000,
    Acc_Type: 'Savings'
  },
  {
    _id: ObjectId("63cff6ca2367415816e1a7d1"),
    Cust_id: 4,
    Acc_Bal: 4500,
    Acc_Type: 'Current'
  },
  {
    _id: ObjectId("63cff7d22367415816e1a7d3"),
    Cust_id: 5,
    Acc_Bal: 3000,
    Acc_Type: 'Savings'
  }
]
Atlas atlas-6qm0d4-shard-0 [primary] Customers>
```

```
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.find({Acc_Type:"Savings",Acc_Bal:{$gt:1200}});
[
  {
    _id: ObjectId("63cff4ad2367415816e1a7ce"),
    Cust_id: 1,
    Acc_Bal: 5000,
    Acc_Type: 'Savings'
  },
  {
    _id: ObjectId("63cff6a02367415816e1a7d0"),
    Cust_id: 3,
    Acc_Bal: 3000,
    Acc_Type: 'Savings'
  },
  {
    _id: ObjectId("63cff7d22367415816e1a7d3"),
    Cust_id: 5,
    Acc_Bal: 3000,
    Acc_Type: 'Savings'
  }
]
```

```
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.find().sort({Acc_Bal:-1}).limit(1);
[
  {
    _id: ObjectId("63cff4ad2367415816e1a7ce"),
    Cust_id: 1,
    Acc_Bal: 5000,
    Acc_Type: 'Savings'
  }
]
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.find().sort({Acc_Bal:+1}).limit(1);
[
  {
    _id: ObjectId("63cff66d2367415816e1a7cf"),
    Cust_id: 2,
    Acc_Bal: 2800,
    Acc_Type: 'Current'
  }
]
Atlas atlas-6qm0d4-shard-0 [primary] Customers>
```

```
2023-01-24T18:33:21.636+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.dyo62sf.mongodb.net/wee
2023-01-24T18:33:22.074+0530    8 document(s) imported successfully. 0 document(s) failed to import.
```

```
Atlas atlas-6qm0d4-shard-0 [primary] Customers> db.Customers.drop();
true
Atlas atlas-6qm0d4-shard-0 [primary] Customers>
```