VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

PRAJWAL S

(1BM24CS209)

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-2025 to Jan-2026

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 **PRAJWAL S(1BM24CS209),** 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 during the year 2025-2026. 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.

Ms. Rashmi H Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

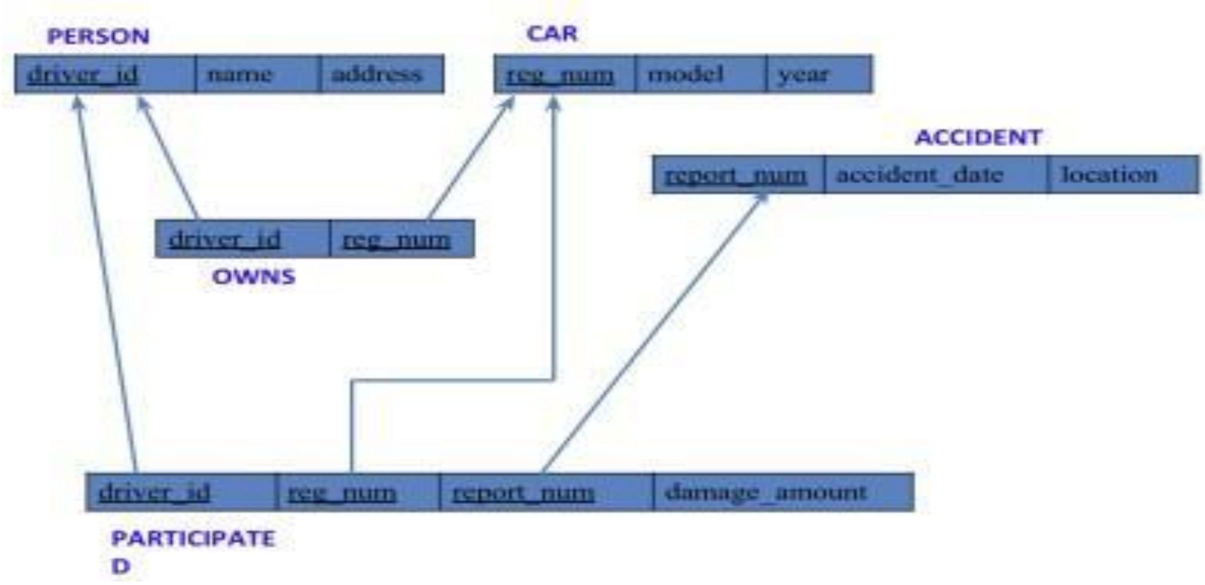
Index

Experiment 1: Insurance Database

Specification of Insurance Database Application

The insurance database must maintain information about drivers, the cars they own, the accidents reported, and the participation of each driver and car in those accidents. Each driver in the system is uniquely identified by a driver ID, along with their name and address, and each car is uniquely identified by its registration number together with details such as model and manufacturing year. The system must allow storing ownership information that links a driver to one or more cars, while also allowing a car to be linked to one or more drivers if shared ownership occurs; duplicate ownership records for the same driver and car must not exist. Accident information must be stored using a unique report number assigned to each accident, along with the date on which the accident occurred and the location where it happened. Every accident reported in the system must have at least one participating driver and car, and this participation is recorded by linking the driver, the involved car, and the accident report together with the corresponding damage amount for that particular involvement. A participation record must reference an existing driver, an existing car, and an existing accident, and no two participation entries may repeat the same combination of driver, car, and accident report. The database must ensure that damage amounts are non-negative, accident dates are valid calendar dates, and car manufacturing years fall within reasonable limits. It must also preserve referential integrity so that ownership or participation entries cannot exist without valid driver, car, and accident information already present in the system. Deletion policies must prevent removal of drivers or cars that appear in past accident participation records unless historical consistency is preserved through controlled deletion rules or archival mechanisms. The system should maintain accurate links between drivers, cars, and accidents at all times, ensuring reliable retrieval of ownership histories, accident histories, and damage information for administrative, legal, and insurance-related purposes.

Schema Diagram



Create database

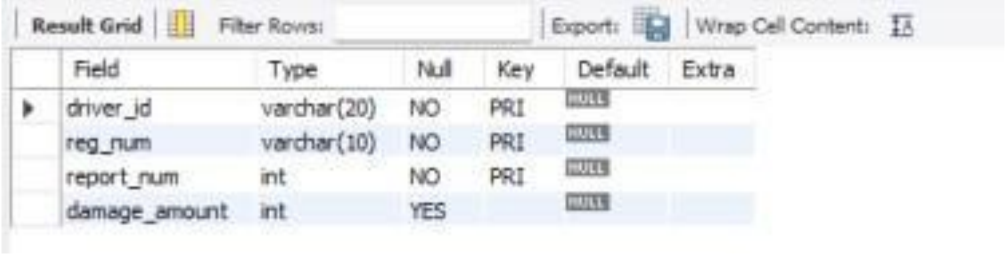
```
CREATE DATABASE IF NOT EXISTS INSURANCE ;
SHOW DATABASES ;
USE INSURANCE ;
```

Create table

```
CREATE TABLE IF NOT EXISTS  PERSON
(     driver_id VARCHAR(10) PRIMARY
KEY,     name VARCHAR(50) NOT NULL,
address VARCHAR(100)
);
CREATE TABLE IF NOT EXISTS CAR (
reg_num VARCHAR(10) PRIMARY KEY,
   model VARCHAR(20),
year INT
);
CREATE TABLE IF NOT EXISTS ACCIDENT (
   report_num INT PRIMARY KEY,
   accident_date DATE,
   location VARCHAR(50)
);
CREATE TABLE IF NOT EXISTS OWNS
(     driver_id VARCHAR(10),     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 IF NOT EXISTS PARTICIPATED
(     driver_id VARCHAR(10),     reg_num
VARCHAR(10),     report_num INT,
damage_amount INT,
   PRIMARY KEY (driver_id, reg_num, report_num),
   FOREIGN KEY (driver_id) REFERENCES PERSON(driver_id),
   FOREIGN KEY (reg_num) REFERENCES CAR(reg_num),
   FOREIGN KEY (report_num) REFERENCES ACCIDENT(report_num)
);
```

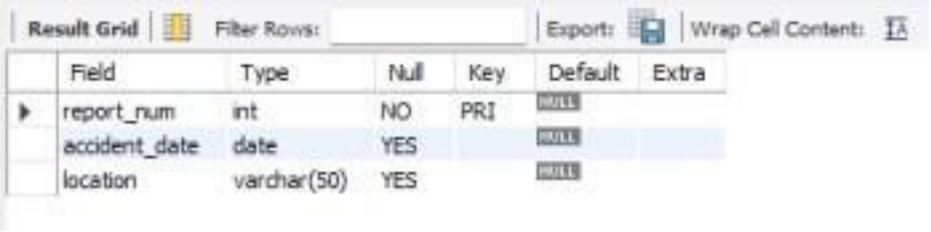
Structure of the table

desc person;



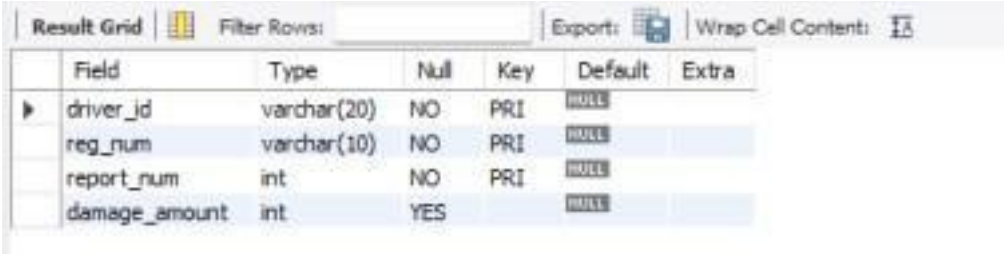
Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

desc accident;



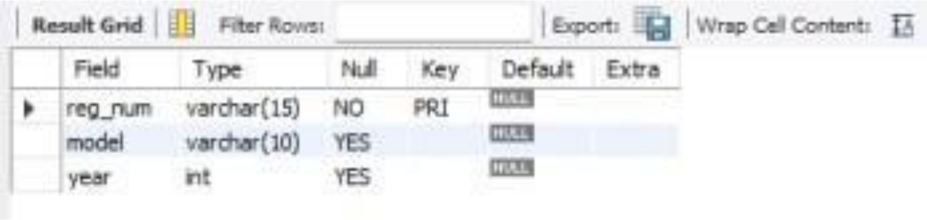
Field	Type	Null	Key	Default	Extra
report_num	int	NO	PRI	NULL	
accident_date	date	YES		NULL	
location	varchar(50)	YES		NULL	

desc participated;



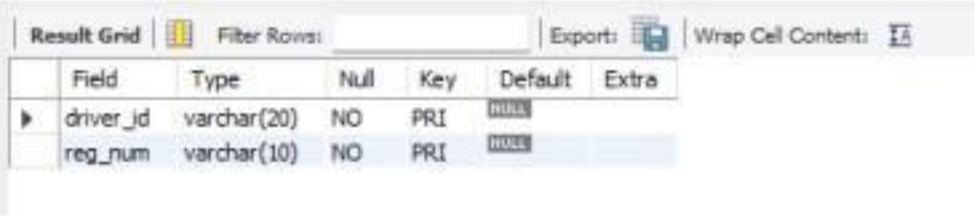
Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
reg_num	varchar(10)	NO	PRI	NULL	
report_num	int	NO	PRI	NULL	
damage_amount	int	YES		NULL	

desc car;



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

desc owns;



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

Inserting Values to the table

insert into person values("A01","Richard", "Srinivas nagar");

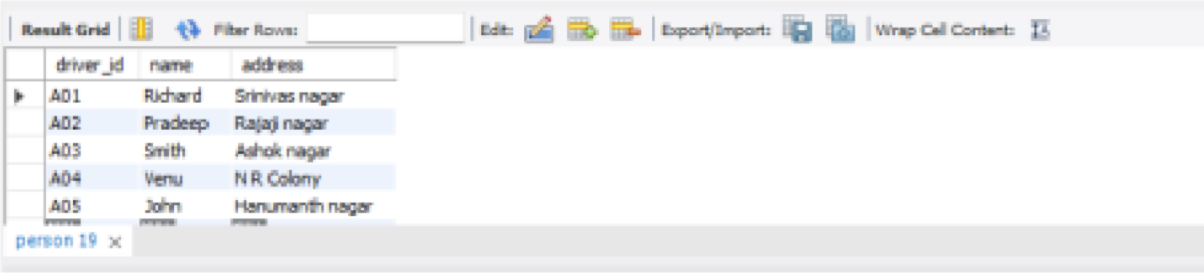
insert into person values("A02","Pradeep", "Rajaji nagar");

insert into person values("A03","Smith", "Ashok nagar");

insert into person values("A04","Venu", "N R Colony"); insert

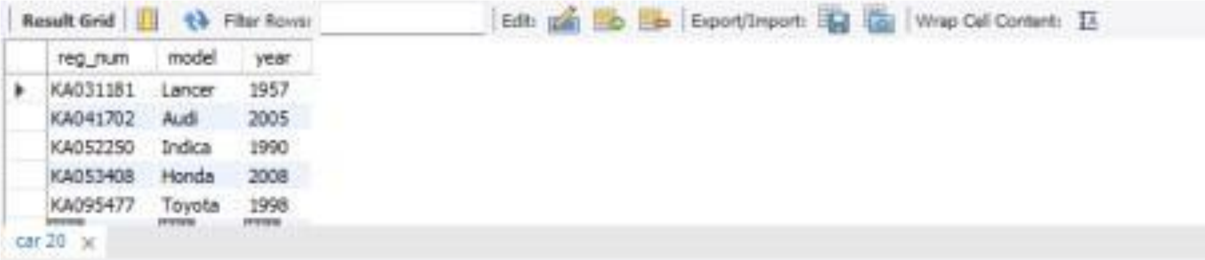
into person values("A05","John", "Hanumanth nagar"); select

* from person;



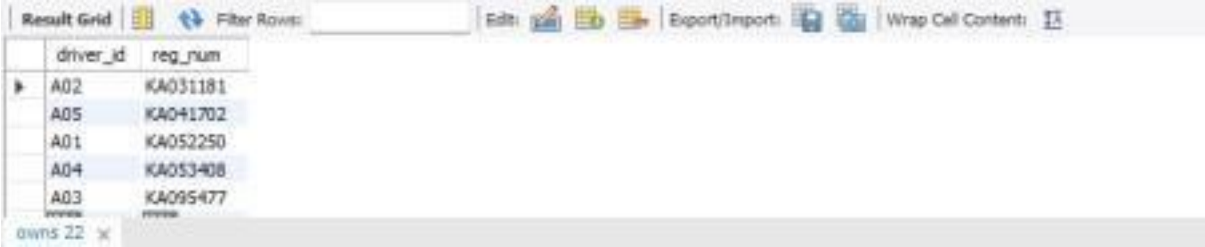
driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R Colony
A05	John	Hanumanth nagar

```
insert into car values("KA052250","Indica", "1990");
insert into car values("KA031181","Lancer", "1957");
insert into car values("KA095477","Toyota", "1998");
insert into car values("KA053408","Honda", "2008");
insert into car values("KA041702","Audi", "2005");  select
* from car;
```



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

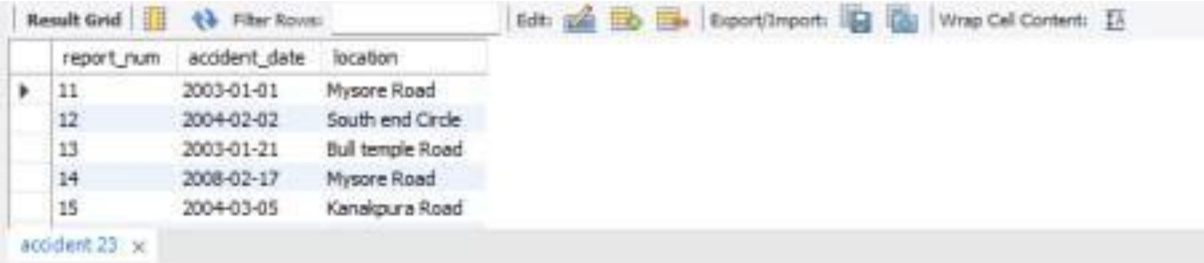
```
insert into owns values("A01","KA052250");  insert into owns values("A02","KA031181");
insert into owns values("A03","KA095477");  insert into owns values("A04","KA053408");
insert into owns values("A05","KA041702");  select * from owns;
```



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

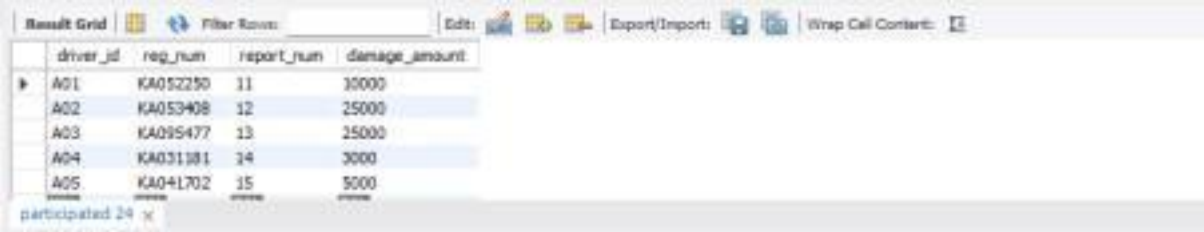
```
insert into accident values(11,'2003-01-01',"Mysore Road");  insert
into accident values(12,'2004-02-02',"South end Circle");  insert
into accident values(13,'2003-01-21',"Bull temple Road");  insert
into accident values(14,'2008-02-17',"Mysore Road");  insert into
accident values(15,'2004-03-05',"Kanakpura Road");
```

```
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 Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road

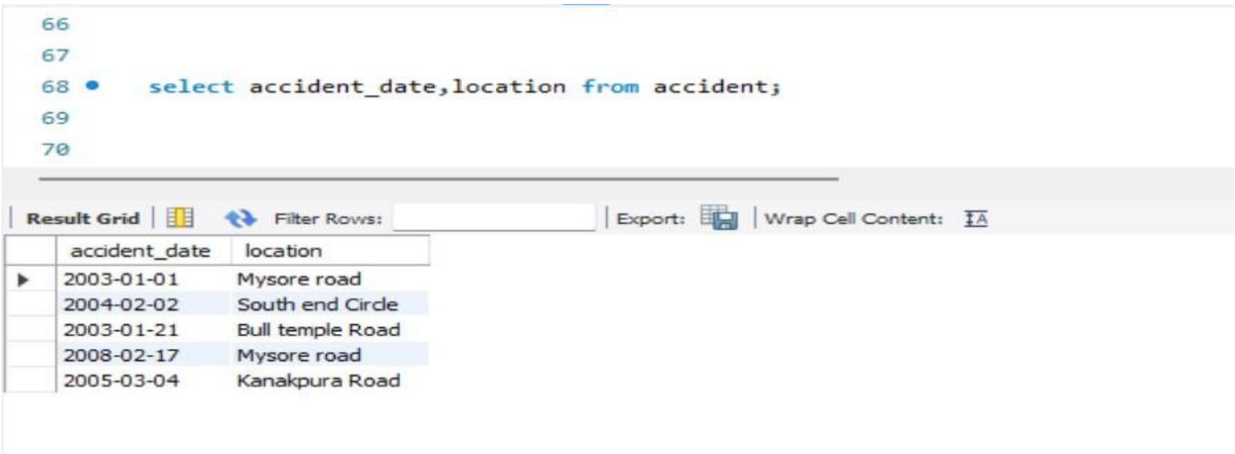
```
insert into participated values("A01","KA052250",11,10000);
insert into participated values("A02","KA053408",12,50000);
insert into participated values("A03","KA095477",13,25000);
insert into participated values("A04","KA031181",14,3000);  insert
into participated values("A05","KA041702",15,5000);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



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

Queries :-

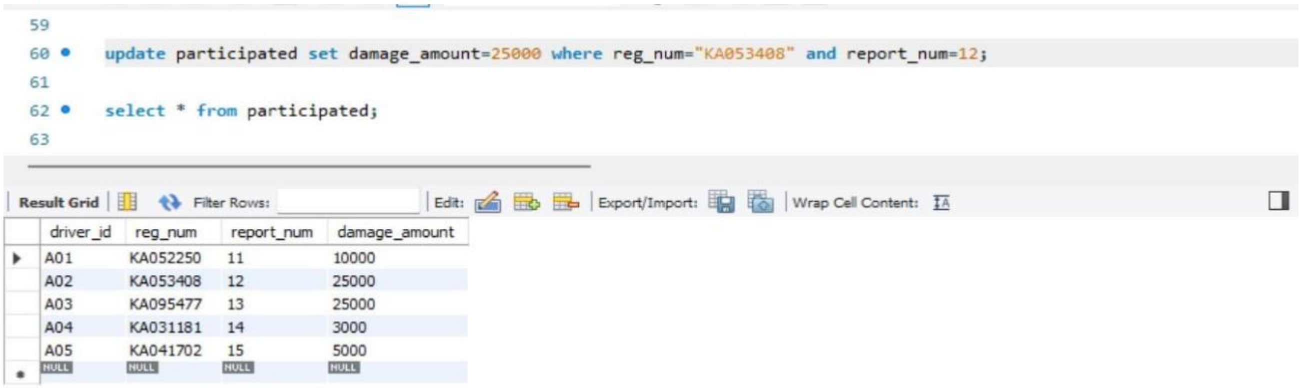
Display Accident date and location



```
select accident_date,location from accident;
```

accident_date	location
2003-01-01	Mysore road
2004-02-02	South end Circle
2003-01-21	Bull temple Road
2008-02-17	Mysore road
2005-03-04	Kanakpura Road

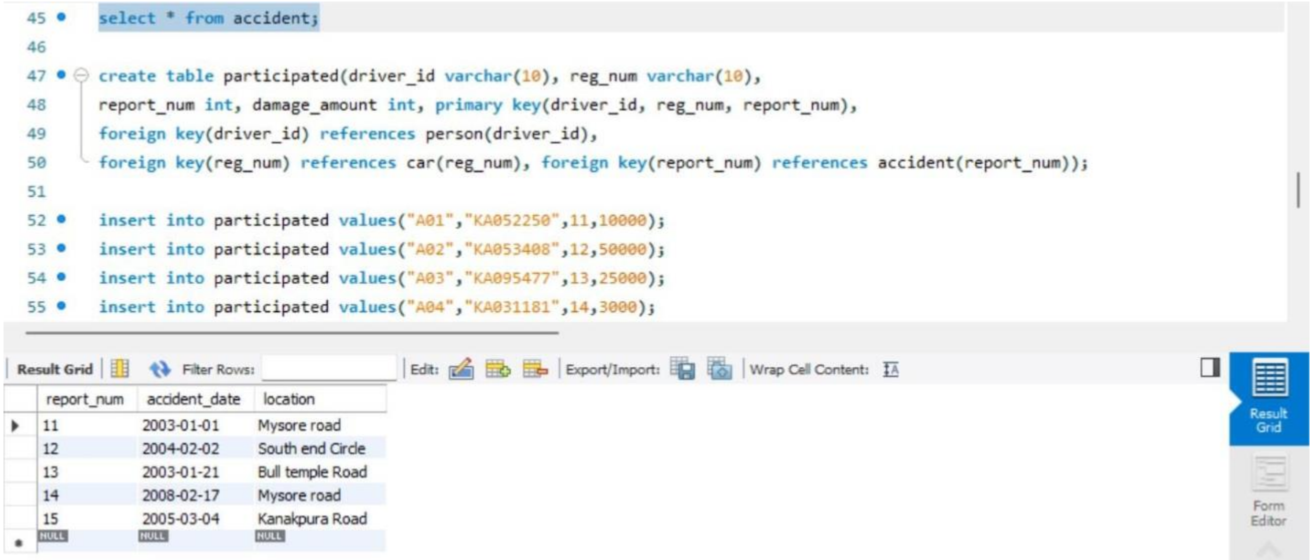
Update the damage amount to 25000 for the car with a specific reg_num (example 'KA053408') for which the accident report number was 12.



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

select * from participated;
```

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

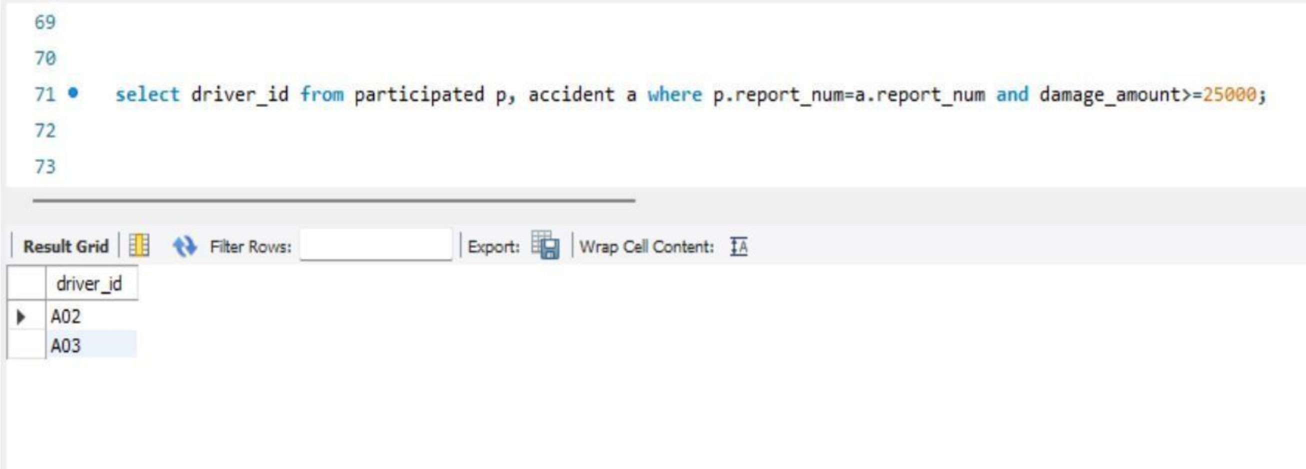
Display Accident date and location



```sql
select * from accident;

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

insert into participated values("A01","KA052250",11,10000);
insert into participated values("A02","KA053408",12,50000);
insert into participated values("A03","KA095477",13,25000);
insert into participated values("A04","KA031181",14,3000);
```

report_num	accident_date	location
11	2003-01-01	Mysore road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore road
15	2005-03-04	Kanakpura Road
NULL	NULL	NULL

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



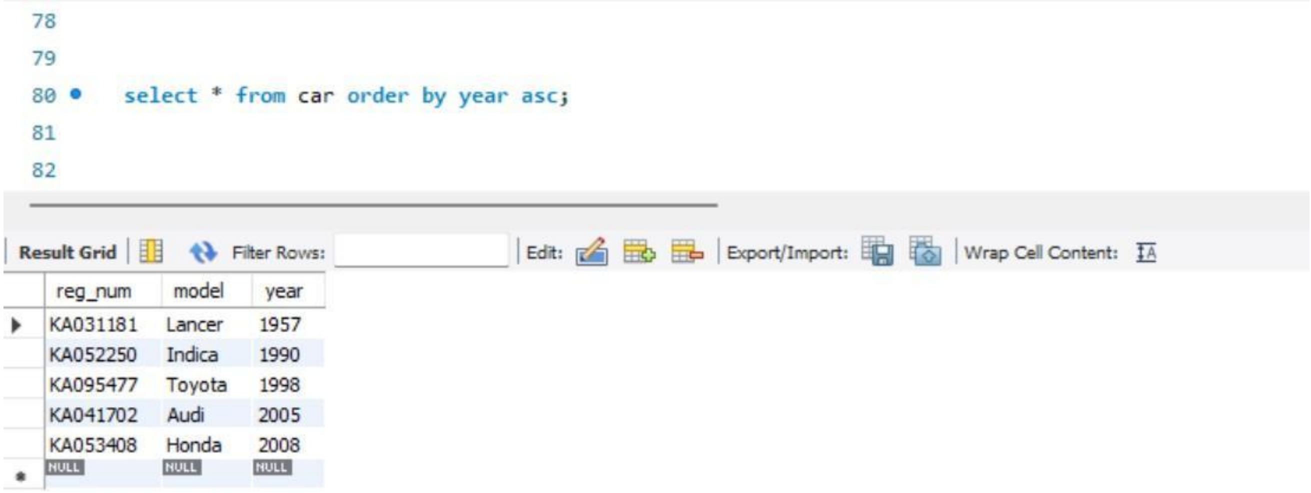
```sql
select driver_id from participated p, accident a where p.report_num=a.report_num and damage_amount>=25000;
```

driver_id
A02
A03

Experiment 2: More Queries on Insurance Database

Queries :-

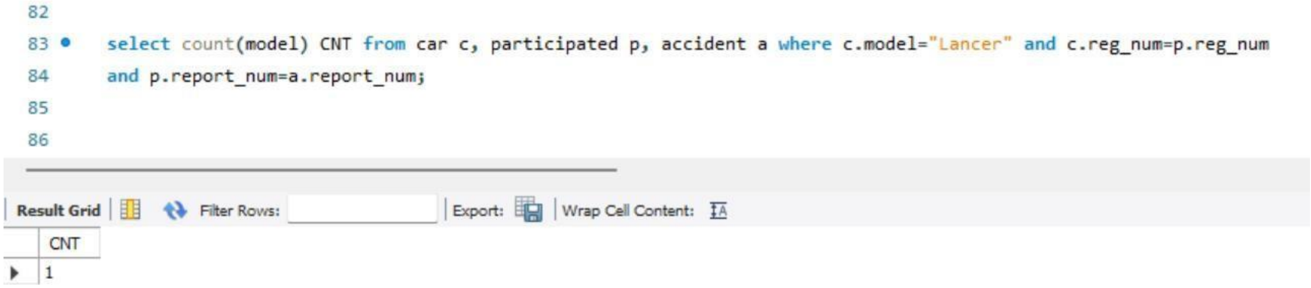
Display the entire CAR relation in the ascending order of manufacturing year.



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

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

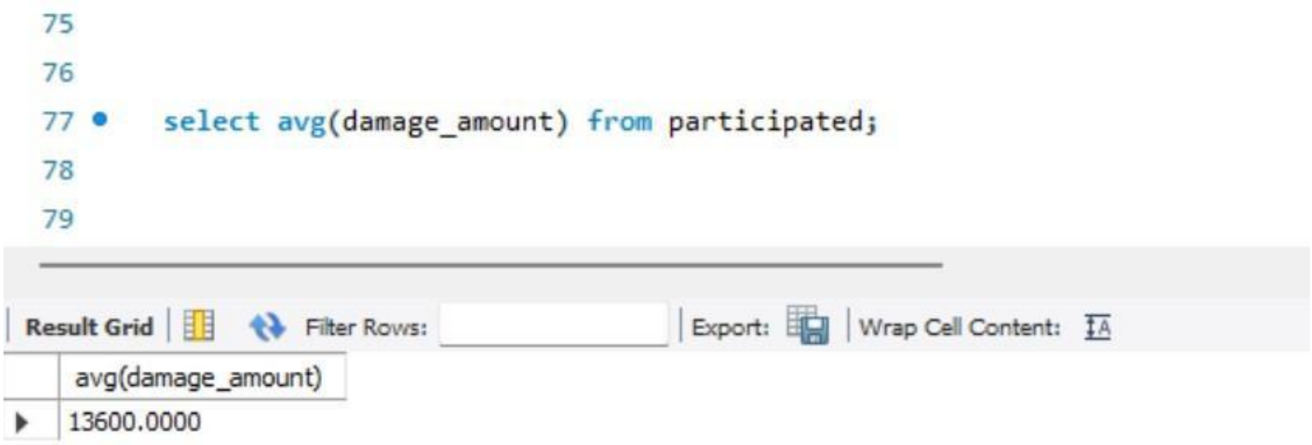
Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved



```sql
select count(model) CNT from car c, participated p, accident a where c.model="Lancer" and c.reg_num=p.reg_num
and p.report_num=a.report_num;
```

CNT
1

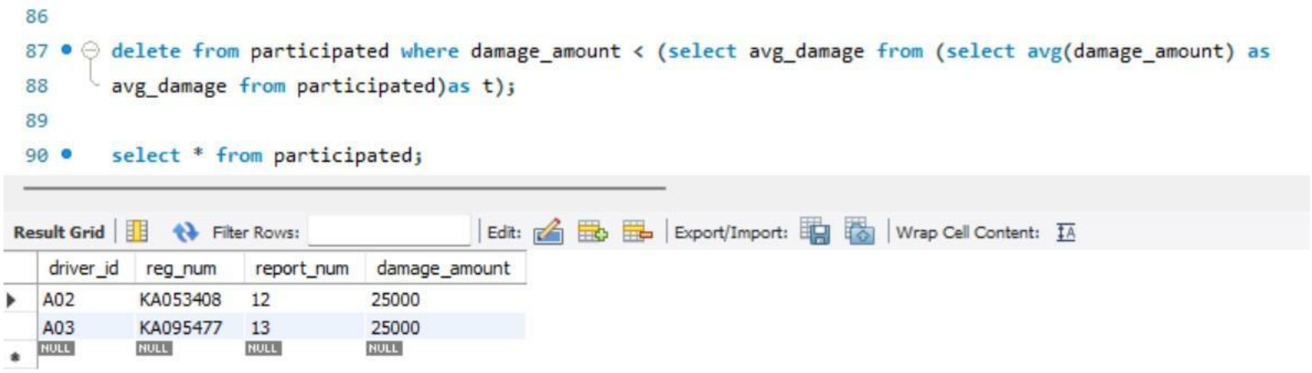
Find the Average Damage Amount



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

avg(damage_amount)
13600.0000

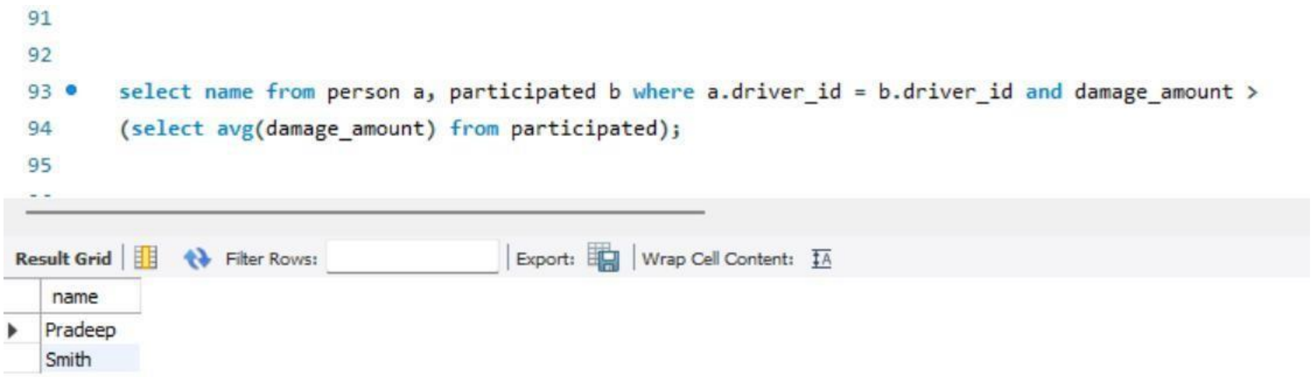
Delete the tuple whose Damage Amount is below the Average Damage Amount



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

select * from participated;
```

driver_id	reg_num	report_num	damage_amount
A02	KA053408	12	25000
A03	KA095477	13	25000
NULL	NULL	NULL	NULL

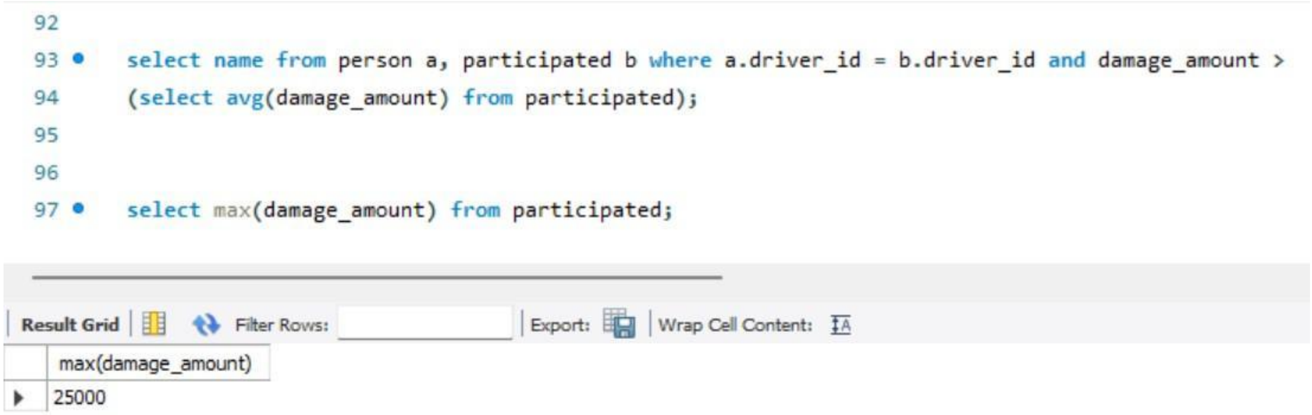
List the name of drivers whose Damage is Greater than the Average Damage Amount



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

```
name
Pradeep
Smith
```

Find Maximum Damage Amount.



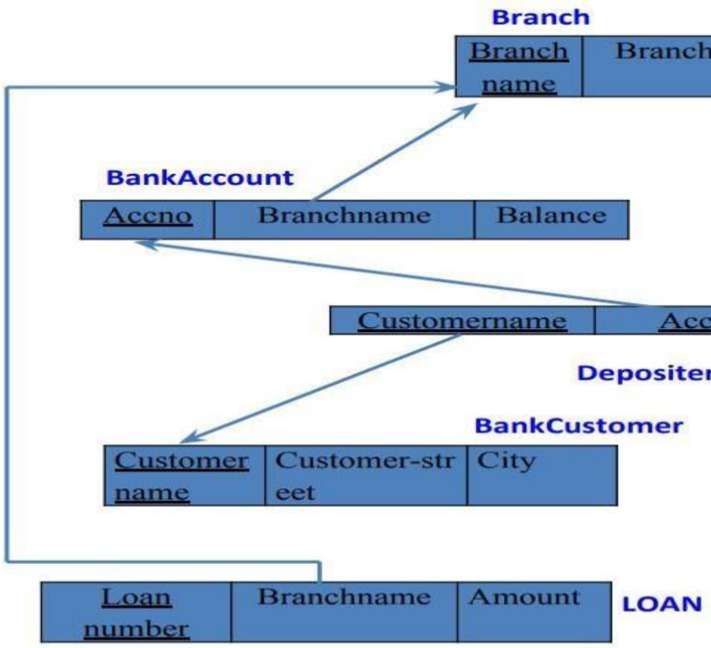
```
select name from person a, participated b where a.driver_id = b.driver_id and damage_amount >
(select avg(damage_amount) from participated);


select max(damage_amount) from participated;
```

```
max(damage_amount)
25000
```

Experiment 3: BANKING DATABASE

The banking system must store information about branches, bank accounts, customers, deposit ,relationships, and loans so that branch details (identified by branch name together with city and total assets) are linked to accounts and loans, each account (identified by an account number) records the branch it belongs to and the current balance, customers are recorded with their name, street , city and a depositor relationship associates a customer with an account; loans are recorded by a unique loan number together with the branch name that issued the loan and the loan amount. Account numbers and loan numbers must be unique identifiers, branch names are used to associate accounts and loans to a branch, and customer names (as model) are used to identify customers referenced by depositor entries; every depositor entry must reference an existing customer and an existing account so that ownership and access relationships are always valid, and duplicate depositor records linking the same customer and account are disallowed. The system must maintain referential integrity so accounts cannot reference a non-existent branch, depositor rows cannot reference missing customers or accounts, and loans must reference an existing branch; deletion of a branch, account, or customer that is referenced by dependent records should be controlled (either disallowed or handled by archival/controlled reassignment) to preserve historical transaction and loan consistency. Numeric and temporal constraints must be enforced: account balances should be constrained to valid values (for example non-negative where overdraft is not allowed), branch assets and loan amounts must be non- negative and within specified business limits, and updates to balance or loan amounts should be auditable. Cardinality rules implied by the schema are enforced: a branch may host many accounts and issue many loans, an account belongs to exactly one branch, a customer may be linked to many accounts through depositor relationships, and an account may have many depositors if joint accounts are permitted by policy. Implementation must prevent orphaned records, ensure uniqueness where required, and rely on application logic or database-level triggers to enforce complex rules such as cascading effects on deletion, business rules about allowed balance operations or overdrafts, and any required validation when transferring accounts between branches or when converting a customer’s identifying details; the database should thus reliably support queries for branch-wise account lists, customer account ownership, account balances, and loan portfolios while preserving historical and referential integrity for auditing and regulatory reporting.

Schema Diagram



Create Database

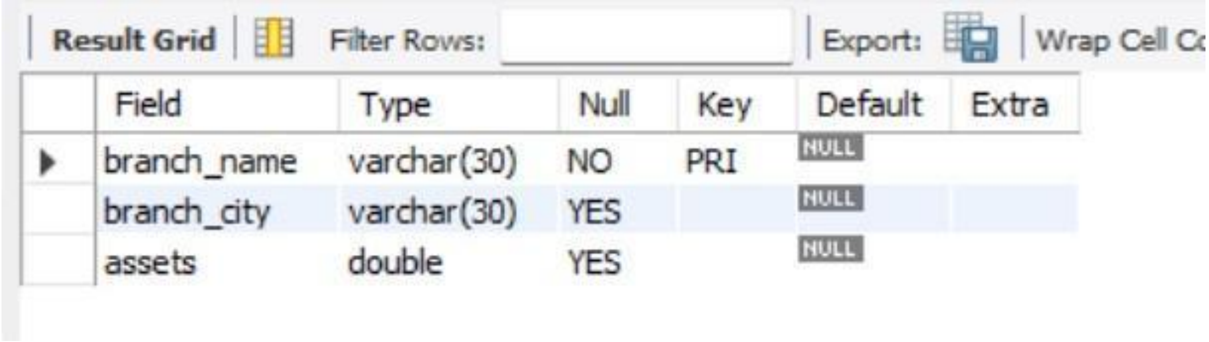
```
create database bank_database;
use bank_database;
```

Create Table

```
create table branch(
branch_name varchar(50),
branch_city varchar(50),
assests real, primary
key(branch_name)
);

create table bankAccount( accno int, branch_name
varchar(50), balance real, primary key (accno), foreign
key(branch_name) references branch(branch_name)
);

create table bankCustomer(
customer_name varchar(50),
customer_street varchar(50),
customer_city varchar(50) ,
primary key (customer_name)
);

create table depositor(
customer_name varchar(50),
accno int,
primary key (customer_name,accno), foreign key(customer_name)
references bankCustomer(customer_name), foreign key(accno)
references bankAccount(accno)
);
desc depositor;

create table loan( loan_number int, branch_name
varchar(50), amount real, primary key(loan_number),
foreign key(branch_name) references
branch(branch_name)
);
```

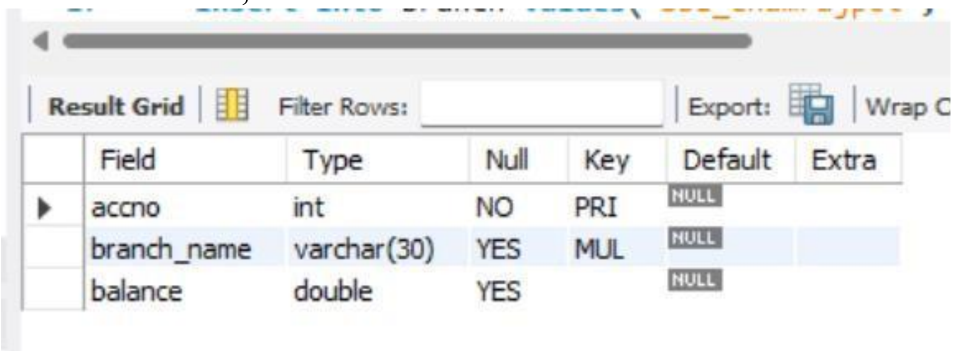
Structure of the table

desc branch;



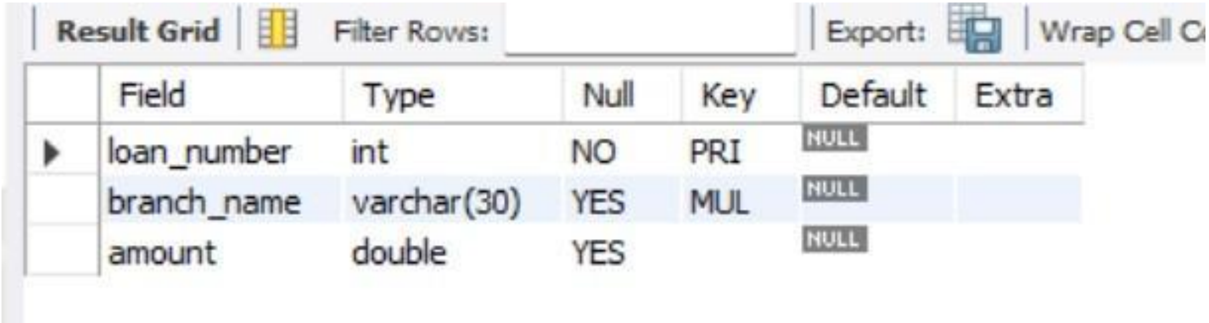
Field	Type	Null	Key	Default	Extra
branch_name	varchar(30)	NO	PRI	NULL	
branch_city	varchar(30)	YES		NULL	
assets	double	YES		NULL	

desc bankaccount;



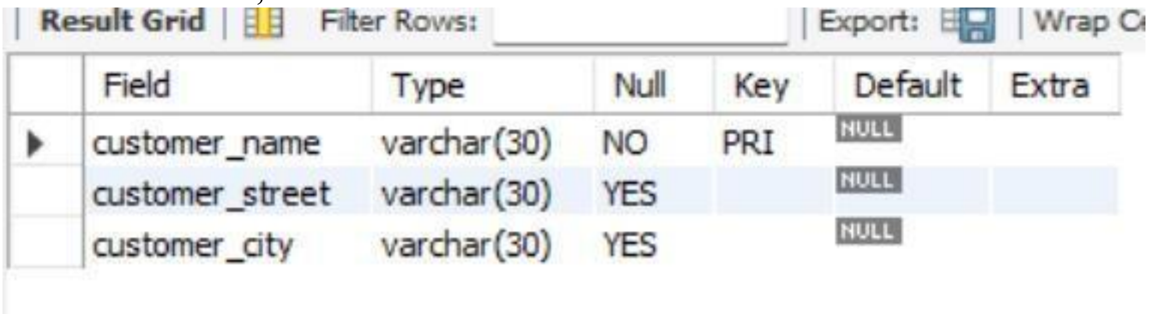
Field	Type	Null	Key	Default	Extra
accno	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
balance	double	YES		NULL	

desc loan;



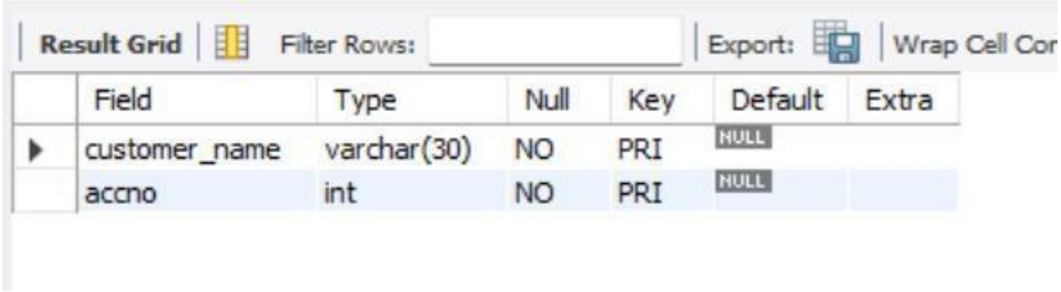
Field	Type	Null	Key	Default	Extra
loan_number	int	NO	PRI	NULL	
branch_name	varchar(30)	YES	MUL	NULL	
amount	double	YES		NULL	

desc bankcustomer;



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

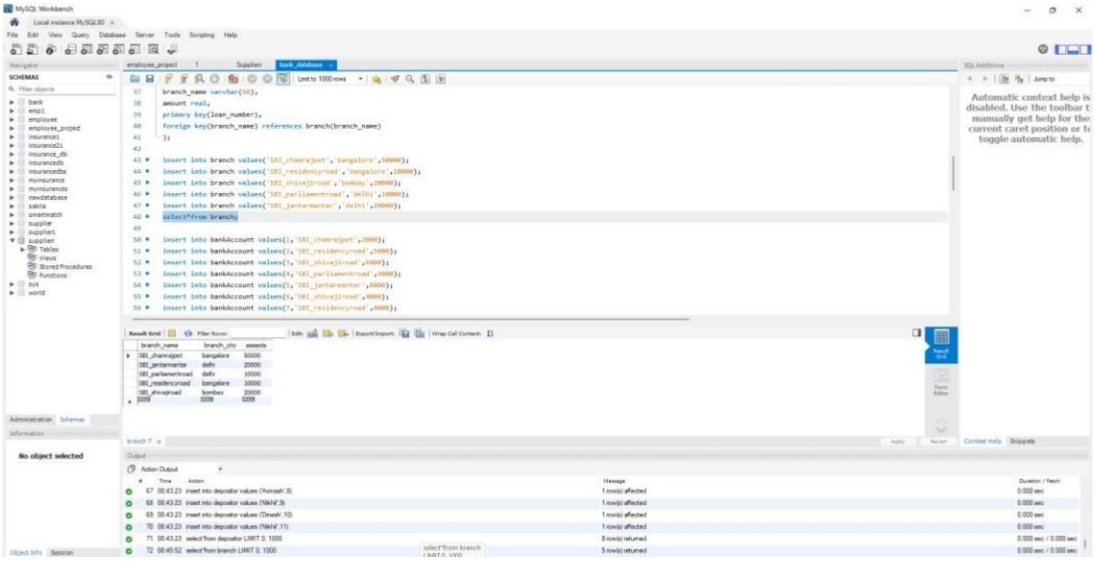
desc depositer;



	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(30)	NO	PRI	NULL	
	accno	int	NO	PRI	NULL	

Insertion of values into tables

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_parliamentroad','delhi',10000); insert into branch values('SBI_jantarmantar','delhi',20000); select*from branch;

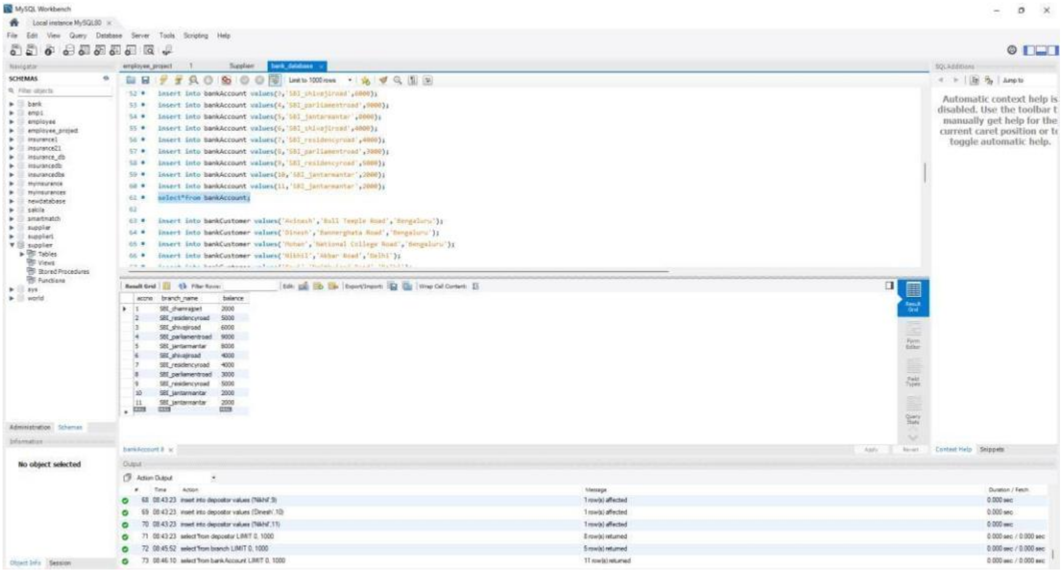


```
branch_name varchar(50),
amount real,
primary key(loan_number),
foreign key(branch_name) references branch(branch_name)
);

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_parliamentroad','delhi',10000);
insert into branch values('SBI_jantarmantar','delhi',20000);
select*from branch;

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_parliamentroad',9000);
insert into bankAccount values(5,'SBI_jantarmantar',8000);
insert into bankAccount values(6,'SBI_shivajiroad',4000);
insert into bankAccount values(7,'SBI_residencyroad',4000);
```

	branch_name	branch_city	assets
▶	SBI_chamrajpet	bangalore	50000
	SBI_jantarmantar	delhi	20000
	SBI_parliamentroad	delhi	10000
	SBI_residencyroad	bangalore	10000
	SBI_shivajiroad	bombay	20000
*	NULL	NULL	NULL

```
#   Time      Action                                        Message            Duration / Fetch
67  08:43:23  insert into depositor values ('Avinash',8)    1 row(s) affected  0.000 sec
68  08:43:23  insert into depositor values ('Nikhil',9)     1 row(s) affected  0.000 sec
69  08:43:23  insert into depositor values ('Dinesh',10)    1 row(s) affected  0.000 sec
70  08:43:23  insert into depositor values ('Nikhil',11)    1 row(s) affected  0.000 sec
71  08:43:23  select*from depositor LIMIT 0, 1000           8 row(s) returned  0.000 sec / 0.000 sec
72  08:45:52  select*from branch LIMIT 0, 1000              5 row(s) returned  0.000 sec / 0.000 sec
```

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_parliamentroad',9000); insert into bankAccount values(5,'SBI_jantarmantar',8000); insert into bankAccount values(6,'SBI_shivajiroad',4000); insert into bankAccount values(7,'SBI_residencyroad',4000); insert into bankAccount values(8,'SBI_parliamentroad',3000); insert into bankAccount values(9,'SBI_residencyroad',5000); insert into bankAccount values(10,'SBI_jantarmantar',2000); insert into bankAccount values(11,'SBI_jantarmantar',2000); select*from bankAccount;

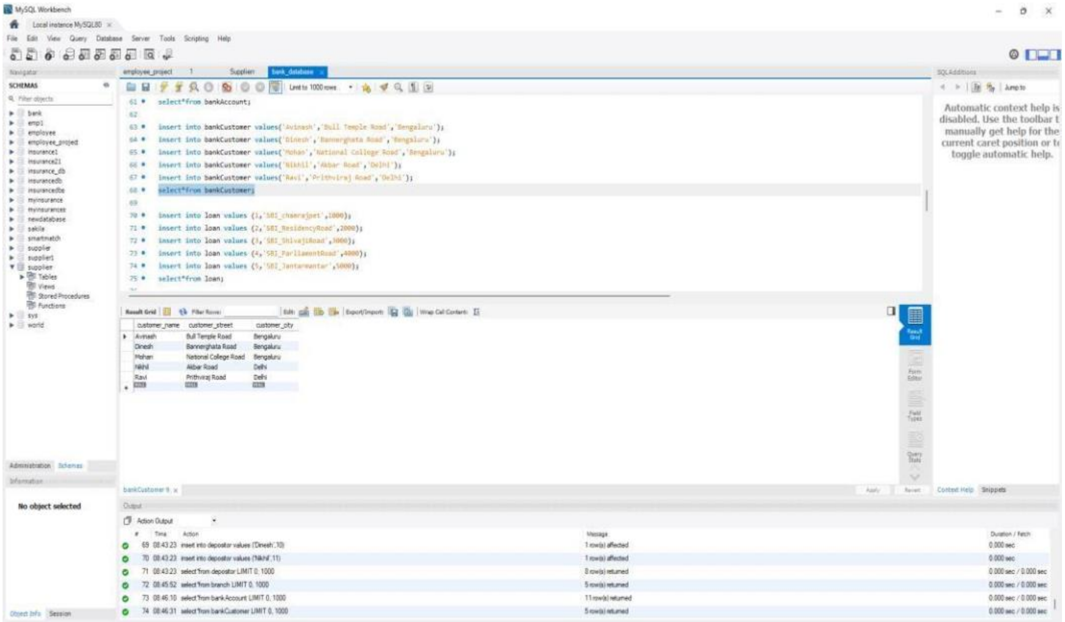


```
insert into bankAccount values(3,'SBI_shivajiroad',6000);
insert into bankAccount values(4,'SBI_parliamentroad',9000);
insert into bankAccount values(5,'SBI_jantarmantar',8000);
insert into bankAccount values(6,'SBI_shivajiroad',4000);
insert into bankAccount values(7,'SBI_residencyroad',4000);
insert into bankAccount values(8,'SBI_parliamentroad',3000);
insert into bankAccount values(9,'SBI_residencyroad',5000);
insert into bankAccount values(10,'SBI_jantarmantar',2000);
insert into bankAccount values(11,'SBI_jantarmantar',2000);
select*from bankAccount;

insert into bankCustomer values('Avinash','Bull Temple Road','Bengaluru');
insert into bankCustomer values('Dinesh','Bannerghata Road','Bengaluru');
insert into bankCustomer values('Mohan','National College Road','Bengaluru');
insert into bankCustomer values('Nikhil','Akbar Road','Delhi');
```

accno	branch_name	balance
1	SBI_chamrajpet	2000
2	SBI_residencyroad	5000
3	SBI_shivajiroad	6000
4	SBI_parliamentroad	9000
5	SBI_jantarmantar	8000
6	SBI_shivajiroad	4000
7	SBI_residencyroad	4000
8	SBI_parliamentroad	3000
9	SBI_residencyroad	5000
10	SBI_jantarmantar	2000
11	SBI_jantarmantar	2000

```
#   Time      Action                                        Message            Duration / Fetch
68  08:43:23  insert into depositor values ('Nikhil',9)     1 row(s) affected  0.000 sec
69  08:43:23  insert into depositor values ('Dinesh',10)    1 row(s) affected  0.000 sec
70  08:43:23  insert into depositor values ('Nikhil',11)    1 row(s) affected  0.000 sec
71  08:43:23  select*from depositor LIMIT 0, 1000           8 row(s) returned  0.000 sec / 0.000 sec
72  08:45:52  select*from branch LIMIT 0, 1000              5 row(s) returned  0.000 sec / 0.000 sec
73  08:46:10  select*from bankAccount LIMIT 0, 1000         11 row(s) returned 0.000 sec / 0.000 sec
```

insert into bankCustomer values('Avinash','Bull Temple Road','Bengaluru'); insert into bankCustomer values('Dinesh','Bannerghata Road','Bengaluru'); insert into bankCustomer values('Mohan','National College Road','Bengaluru'); insert into bankCustomer values('Nikhil','Akbar Road','Delhi'); insert into bankCustomer values('Ravi','Prithviraj Road','Delhi'); select*from bankCustomer;

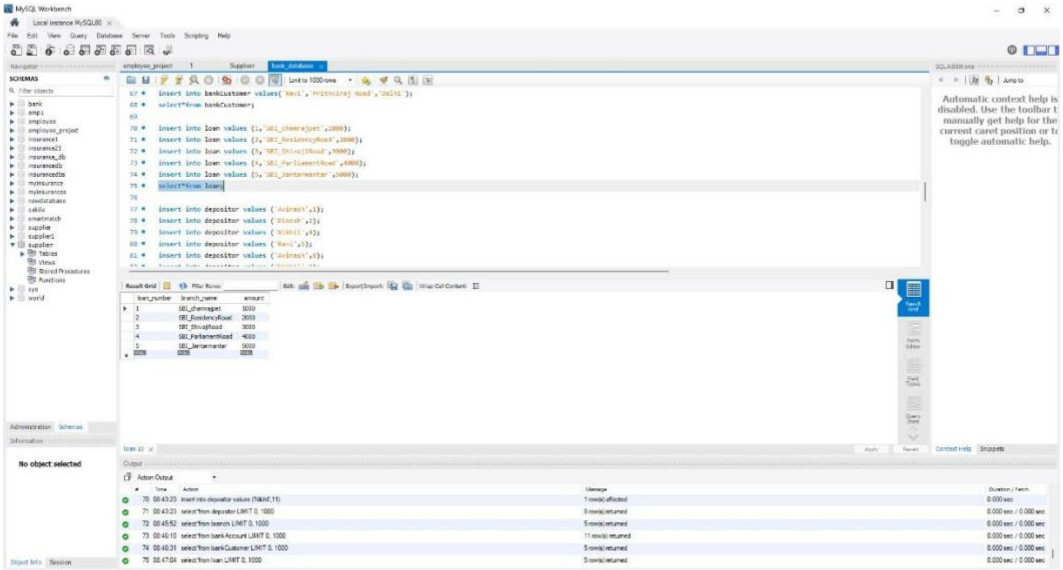


```
select*from bankAccount;

insert into bankCustomer values('Avinash','Bull Temple Road','Bengaluru');
insert into bankCustomer values('Dinesh','Bannerghata Road','Bengaluru');
insert into bankCustomer values('Mohan','National College Road','Bengaluru');
insert into bankCustomer values('Nikhil','Akbar Road','Delhi');
insert into bankCustomer values('Ravi','Prithviraj Road','Delhi');
select*from bankCustomer;

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_ParliamentRoad',4000);
insert into loan values (5,'SBI_Jantarmantar',5000);
select*from loan;
```

customer_name	customer_street	customer_city
Avinash	Bull Temple Road	Bengaluru
Dinesh	Bannerghata Road	Bengaluru
Mohan	National College Road	Bengaluru
Nikhil	Akbar Road	Delhi
Ravi	Prithviraj Road	Delhi

```
#   Time      Action                                        Message            Duration / Fetch
69  08:43:23  insert into depositor values ('Dinesh',10)    1 row(s) affected  0.000 sec
70  08:43:23  insert into depositor values ('Nikhil',11)    1 row(s) affected  0.000 sec
71  08:43:23  select*from depositor LIMIT 0, 1000           8 row(s) returned  0.000 sec / 0.000 sec
72  08:45:52  select*from branch LIMIT 0, 1000              5 row(s) returned  0.000 sec / 0.000 sec
73  08:46:10  select*from bankAccount LIMIT 0, 1000         11 row(s) returned 0.000 sec / 0.000 sec
74  08:46:31  select*from bankCustomer LIMIT 0, 1000        5 row(s) returned  0.000 sec / 0.000 sec
```

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_ParliamentRoad',4000); insert into loan values (5,'SBI_Jantarmantar',5000); select*from loan;



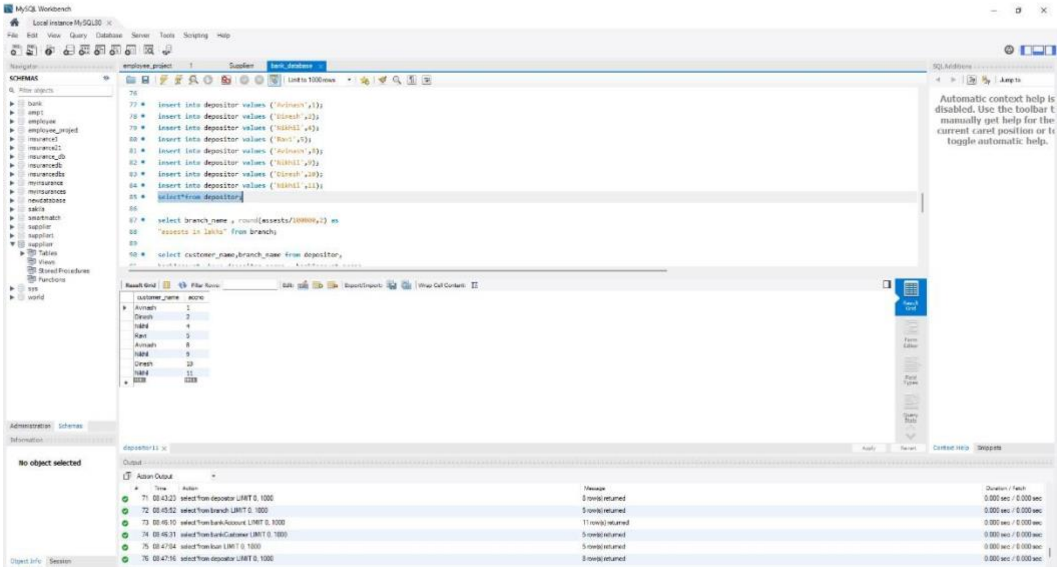
```
insert into bankCustomer values('Ravi','Prithviraj Road','Delhi');
select*from bankCustomer;

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_ParliamentRoad',4000);
insert into loan values (5,'SBI_Jantarmantar',5000);
select*from loan;

insert into depositor values ('Avinash',1);
insert into depositor values ('Dinesh',2);
insert into depositor values ('Nikhil',4);
insert into depositor values ('Ravi',5);
insert into depositor values ('Avinash',8);
```

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

```
#   Time      Action                                          Message             Duration / Fetch
70  08:43:23  insert into depositor values ('Nikhil',11)      1 row(s) affected   0.000 sec
71  08:43:23  select*from depositor LIMIT 0, 1000             8 row(s) returned   0.000 sec / 0.000 sec
72  08:45:52  select*from branch LIMIT 0, 1000                5 row(s) returned   0.000 sec / 0.000 sec
73  08:46:10  select*from bankAccount LIMIT 0, 1000           11 row(s) returned  0.000 sec / 0.000 sec
74  08:46:31  select*from bankCustomer LIMIT 0, 1000          5 row(s) returned   0.000 sec / 0.000 sec
75  08:47:04  select*from loan LIMIT 0, 1000                  5 row(s) returned   0.000 sec / 0.000 sec
```

insert into depositor values ('Avinash',1);
insert into depositor values ('Dinesh',2);
insert into depositor values ('Nikhil',4);
insert into depositor values ('Ravi',5);
insert into depositor values ('Avinash',8);
insert into depositor values ('Nikhil',9);
insert into depositor values ('Dinesh',10);
insert into depositor values ('Nikhil',11);

select*from depositor;



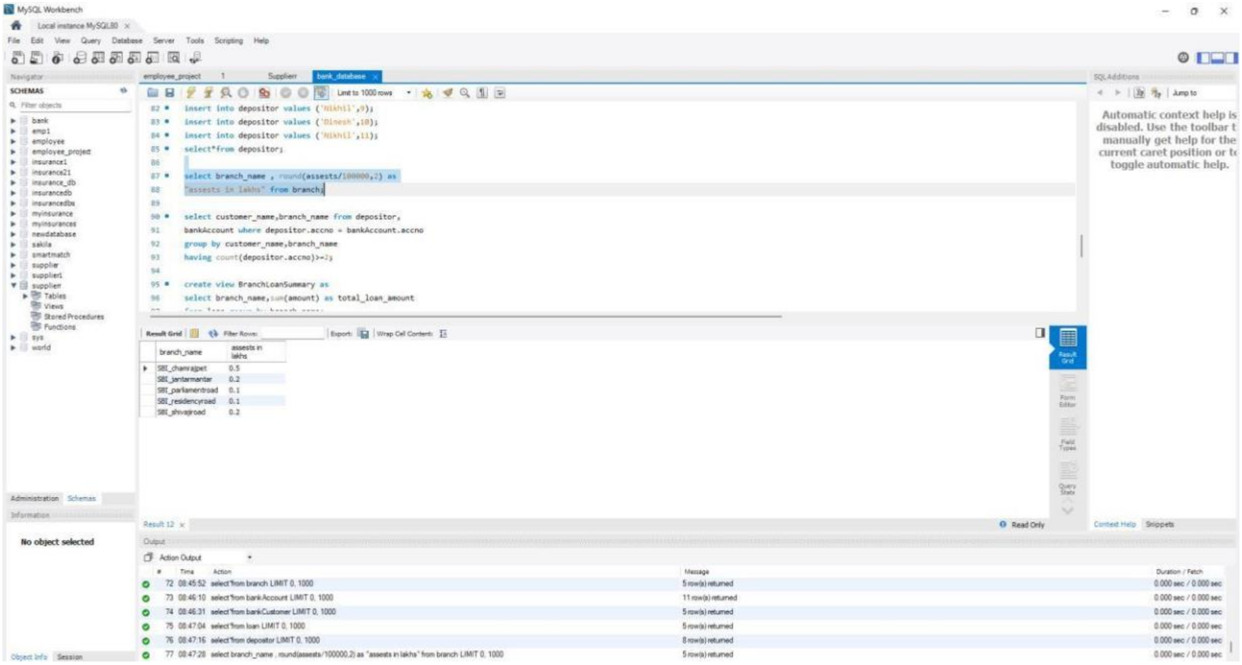
```sql
insert into depositor values ('Avinash',1);
insert into depositor values ('Dinesh',2);
insert into depositor values ('Nikhil',4);
insert into depositor values ('Ravi',5);
insert into depositor values ('Avinash',8);
insert into depositor values ('Nikhil',9);
insert into depositor values ('Dinesh',10);
insert into depositor values ('Nikhil',11);
select*from depositor;

select branch_name , round(assests/100000,2) as
"assests in lakhs" from branch;

select customer_name,branch_name from depositor,
```

customer_name	accno
Avinash	1
Dinesh	2
Nikhil	4
Ravi	5
Avinash	8
Nikhil	9
Dinesh	10
Nikhil	11
NULL	NULL

```
#   Time      Action                                  Message            Duration / Fetch
71  08:43:23  select*from depositor LIMIT 0, 1000     8 row(s) returned  0.000 sec / 0.000 sec
72  08:45:52  select*from branch LIMIT 0, 1000        5 row(s) returned  0.000 sec / 0.000 sec
73  08:46:10  select*from bankAccount LIMIT 0, 1000   11 row(s) returned 0.000 sec / 0.000 sec
74  08:46:31  select*from bankCustomer LIMIT 0, 1000  5 row(s) returned  0.000 sec / 0.000 sec
75  08:47:04  select*from loan LIMIT 0, 1000          5 row(s) returned  0.000 sec / 0.000 sec
76  08:47:16  select*from depositor LIMIT 0, 1000     8 row(s) returned  0.000 sec / 0.000 sec
```

Queries :-

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



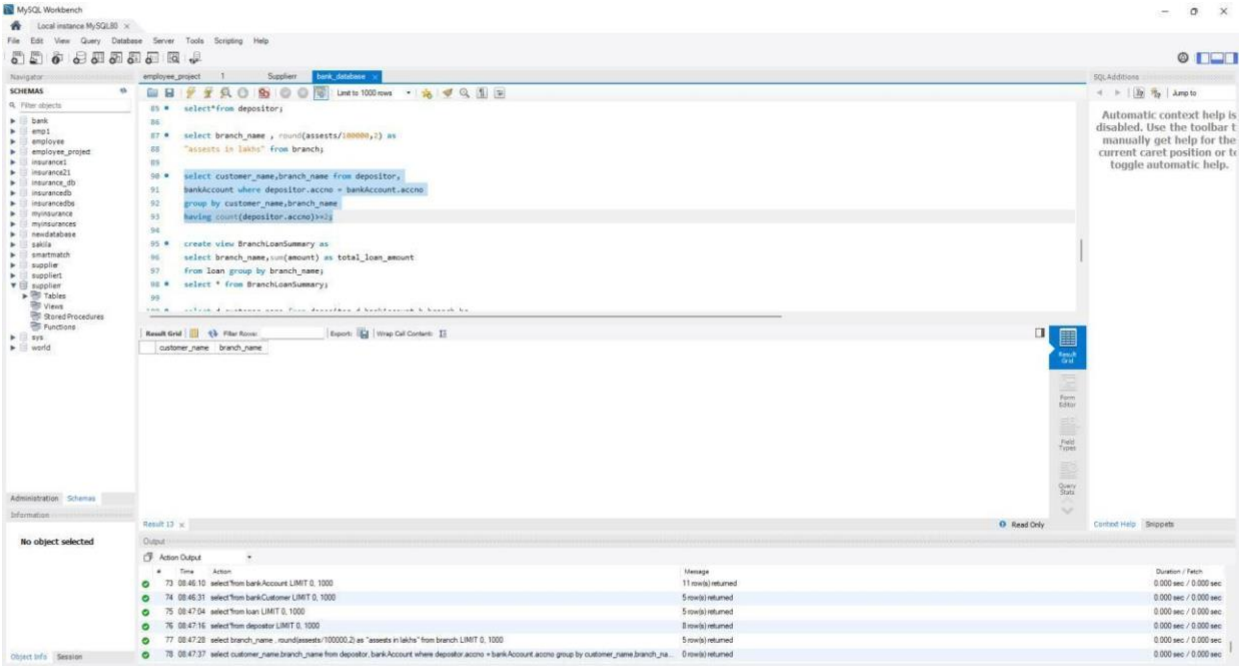
```sql
insert into depositor values ('Nikhil',9);
insert into depositor values ('Dinesh',10);
insert into depositor values ('Nikhil',11);
select*from depositor;

select branch_name , round(assests/100000,2) as
"assests in lakhs" from branch;

select customer_name,branch_name from depositor,
bankAccount where depositor.accno = bankAccount.accno
group by customer_name,branch_name
having count(depositor.accno)>=2;

create view BranchLoanSummary as
select branch_name,sum(amount) as total_loan_amount
```

branch_name	assests in lakhs
SBI_chamrajpet	0.5
SBI_jantarmantar	0.2
SBI_parlamentroad	0.1
SBI_residencyroad	0.1
SBI_shivajroad	0.2

```
#   Time      Action                                                                                       Message            Duration / Fetch
72  08:45:52  select*from branch LIMIT 0, 1000                                                             5 row(s) returned  0.000 sec / 0.000 sec
73  08:46:10  select*from bankAccount LIMIT 0, 1000                                                        11 row(s) returned 0.000 sec / 0.000 sec
74  08:46:31  select*from bankCustomer LIMIT 0, 1000                                                       5 row(s) returned  0.000 sec / 0.000 sec
75  08:47:04  select*from loan LIMIT 0, 1000                                                               5 row(s) returned  0.000 sec / 0.000 sec
76  08:47:16  select*from depositor LIMIT 0, 1000                                                          8 row(s) returned  0.000 sec / 0.000 sec
77  08:47:28  select branch_name , round(assests/100000,2) as "assests in lakhs" from branch LIMIT 0, 1000  5 row(s) returned  0.000 sec / 0.000 sec
```

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

SBI_ResidencyRoad).

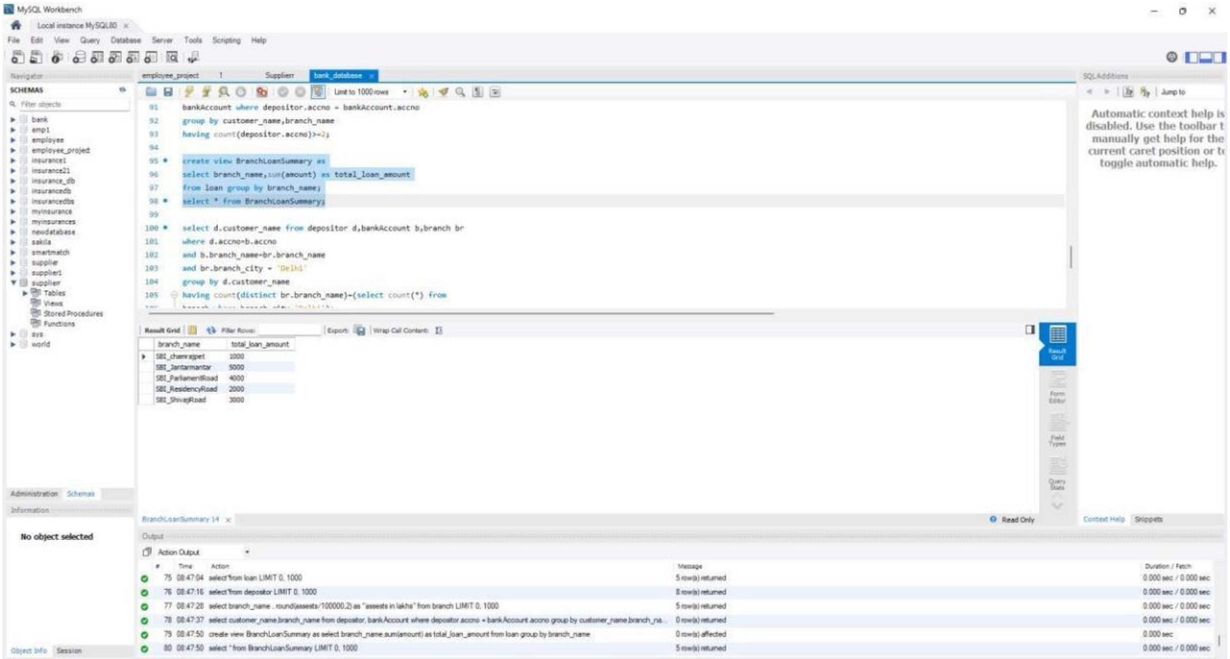


```
select*from depositor;

select branch_name , round(assests/100000,2) as
"assests in lakhs" from branch;

select customer_name,branch_name from depositor,
bankAccount where depositor.accno = bankAccount.accno
group by customer_name,branch_name
having count(depositor.accno)>=2;

create view BranchLoanSummary as
select branch_name,sum(amount) as total_loan_amount
from loan group by branch_name;
select * from BranchLoanSummary;
```

customer_name	branch_name

```
#   Time      Action                                                                                                              Message             Duration / Fetch
73  08:46:10  select*from bankAccount LIMIT 0, 1000                                                                                11 row(s) returned  0.000 sec / 0.000 sec
74  08:46:31  select*from bankCustomer LIMIT 0, 1000                                                                               5 row(s) returned   0.000 sec / 0.000 sec
75  08:47:04  select*from loan LIMIT 0, 1000                                                                                       5 row(s) returned   0.000 sec / 0.000 sec
76  08:47:16  select*from depositor LIMIT 0, 1000                                                                                  8 row(s) returned   0.000 sec / 0.000 sec
77  08:47:28  select branch_name , round(assests/100000,2) as "assests in lakhs" from branch LIMIT 0, 1000                          5 row(s) returned   0.000 sec / 0.000 sec
78  08:47:37  select customer_name,branch_name from depositor, bankAccount where depositor.accno = bankAccount.accno group by customer_name,branch_na...  0 row(s) returned   0.000 sec / 0.000 sec
```

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



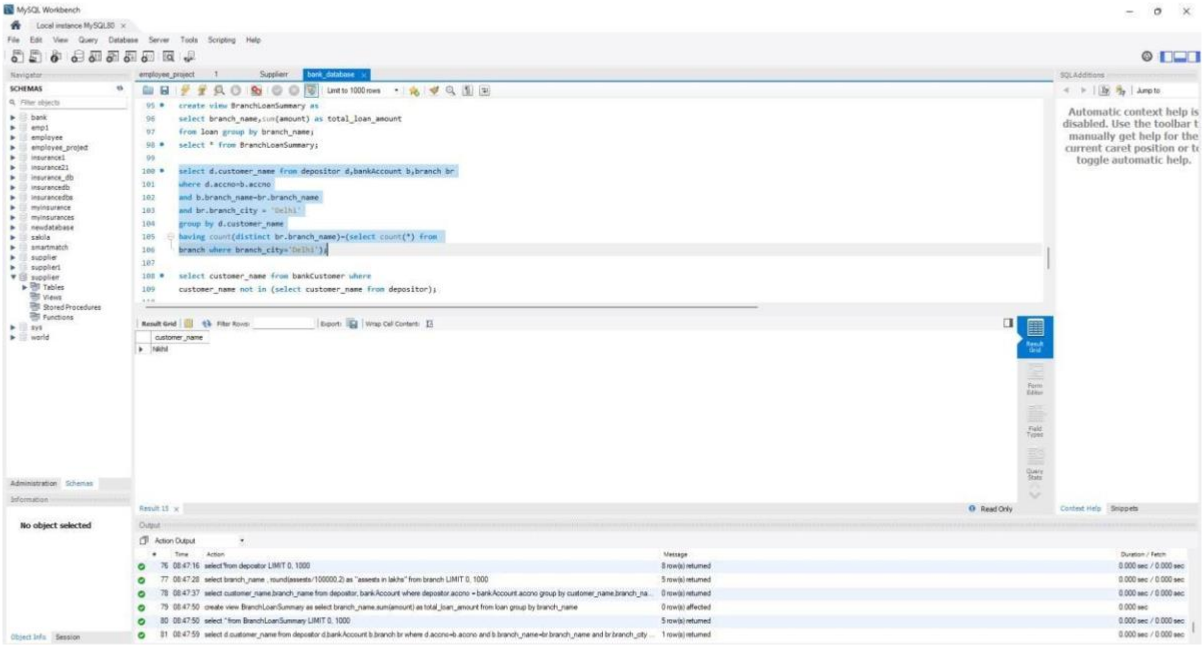
```
bankAccount where depositor.accno = bankAccount.accno
group by customer_name,branch_name
having count(depositor.accno)>=2;

create view BranchLoanSummary as
select branch_name,sum(amount) as total_loan_amount
from loan group by branch_name;
select * from BranchLoanSummary;

select d.customer_name from depositor d,bankAccount b,branch br
where d.accno=b.accno
and b.branch_name=br.branch_name
and br.branch_city = 'Delhi'
group by d.customer_name
having count(distinct br.branch_name)=(select count(*) from
```

branch_name	total_loan_amount
SBI_chamrajpet	1000
SBI_Jantarmantar	5000
SBI_ParlamentRoad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

```
#   Time      Action                                                                                                              Message             Duration / Fetch
75  08:47:04  select*from loan LIMIT 0, 1000                                                                                       5 row(s) returned   0.000 sec / 0.000 sec
76  08:47:16  select*from depositor LIMIT 0, 1000                                                                                  8 row(s) returned   0.000 sec / 0.000 sec
77  08:47:28  select branch_name , round(assests/100000,2) as "assests in lakhs" from branch LIMIT 0, 1000                          5 row(s) returned   0.000 sec / 0.000 sec
78  08:47:37  select customer_name,branch_name from depositor, bankAccount where depositor.accno = bankAccount.accno group by customer_name,branch_na...  0 row(s) returned   0.000 sec / 0.000 sec
79  08:47:50  create view BranchLoanSummary as select branch_name,sum(amount) as total_loan_amount from loan group by branch_name   0 row(s) affected   0.000 sec
80  08:47:50  select * from BranchLoanSummary LIMIT 0, 1000                                                                        5 row(s) returned   0.000 sec / 0.000 sec
```

Experiment 4: More Queries on Bank Database

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

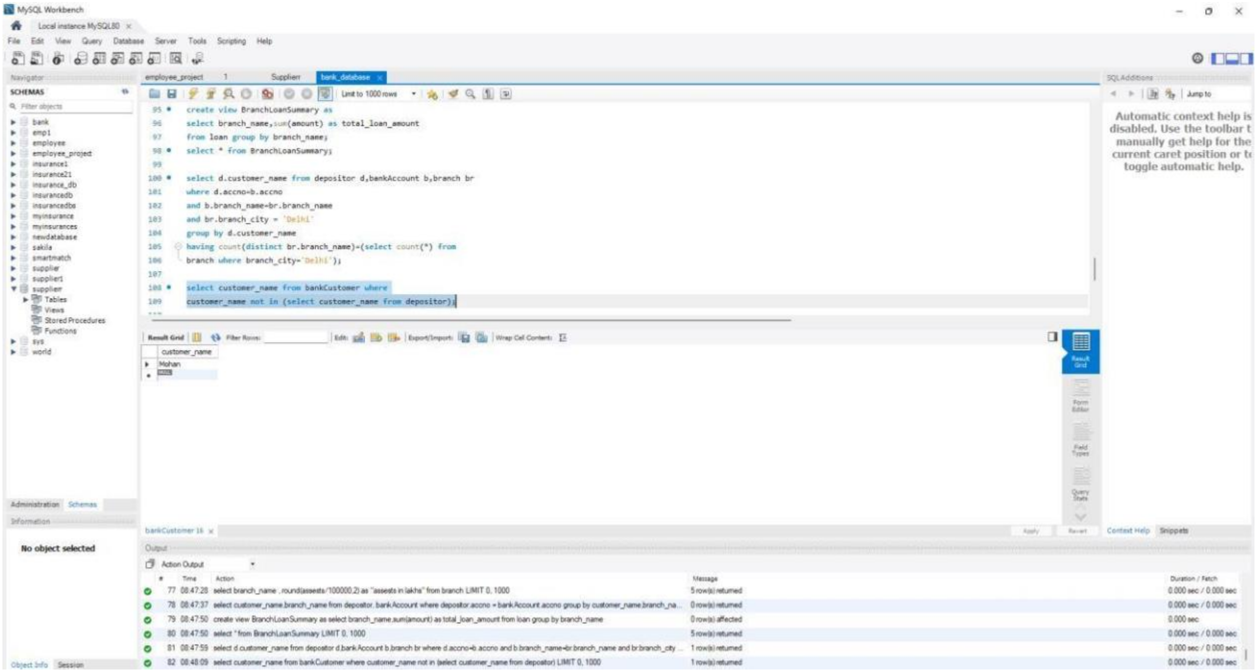


```
create view BranchLoanSummary as
select branch_name,sum(amount) as total_loan_amount
from loan group by branch_name;
select * from BranchLoanSummary;

select d.customer_name from depositor d,bankAccount b,branch br
where d.accno=b.accno
and b.branch_name=br.branch_name
and br.branch_city = 'Delhi'
group by d.customer_name
having count(distinct br.branch_name)=(select count(*) from
branch where branch_city='Delhi');

select customer_name from bankCustomer where
customer_name not in (select customer_name from depositor);
```

customer_name
Nikhil

```
#   Time      Action                                                                                                                          Message            Duration / Fetch
76  08:47:16  select*from depositor LIMIT 0, 1000                                                                                             8 row(s) returned  0.000 sec / 0.000 sec
77  08:47:28  select branch_name , round(assets/100000,2) as "assets in lakhs" from branch LIMIT 0, 1000                                      5 row(s) returned  0.000 sec / 0.000 sec
78  08:47:37  select customer_name,branch_name from depositor, bankAccount where depositor.accno = bankAccount.accno group by customer_name,branch_na...  0 row(s) returned  0.000 sec / 0.000 sec
79  08:47:50  create view BranchLoanSummary as select branch_name,sum(amount) as total_loan_amount from loan group by branch_name               0 row(s) affected  0.000 sec
80  08:47:50  select * from BranchLoanSummary LIMIT 0, 1000                                                                                   5 row(s) returned  0.000 sec / 0.000 sec
81  08:47:59  select d.customer_name from depositor d,bankAccount b,branch br where d.accno=b.accno and b.branch_name=br.branch_name and br.branch_city ...  1 row(s) returned  0.000 sec / 0.000 sec
```

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

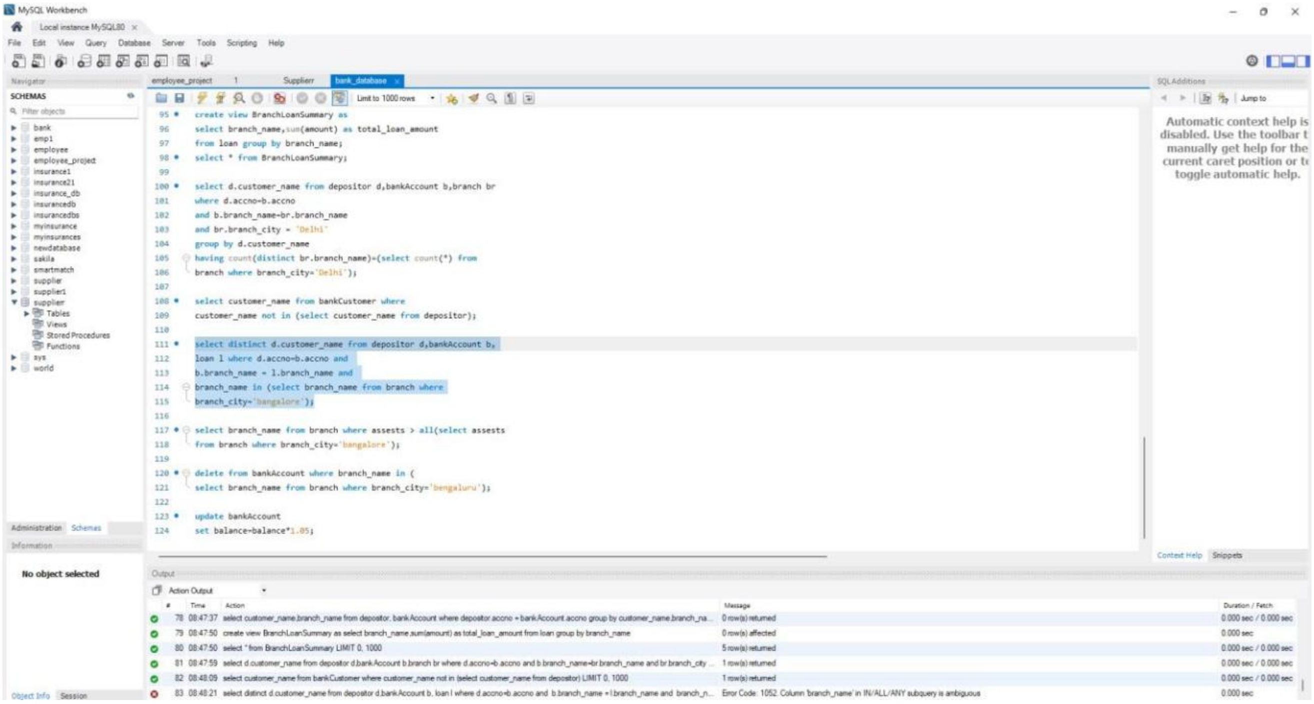


```sql
create view BranchLoanSummary as
select branch_name,sum(amount) as total_loan_amount
from loan group by branch_name;
select * from BranchLoanSummary;

select d.customer_name from depositor d,bankAccount b,branch br
where d.accno=b.accno
and b.branch_name=br.branch_name
and br.branch_city = 'Delhi'
group by d.customer_name
having count(distinct br.branch_name)=(select count(*) from
branch where branch_city='Delhi');

select customer_name from bankCustomer where
customer_name not in (select customer_name from depositor);
```

customer_name
Mohan
NULL

```
#   Time      Action                                                                                                          Message           Duration / Fetch
77  08:47:28  select branch_name , round(assests/100000,2) as "assets in lakhs" from branch LIMIT 0, 1000                    5 row(s) returned  0.000 sec / 0.000 sec
78  08:47:37  select customer_name,branch_name from depositor, bankAccount where depositor.accno = bankAccount.accno group by customer_name,branch_na...  0 row(s) returned  0.000 sec / 0.000 sec
79  08:47:50  create view BranchLoanSummary as select branch_name,sum(amount) as total_loan_amount from loan group by branch_name  0 row(s) affected  0.000 sec
80  08:47:50  select * from BranchLoanSummary LIMIT 0, 1000                                                                   5 row(s) returned  0.000 sec / 0.000 sec
81  08:47:59  select d.customer_name from depositor d,bankAccount b,branch br where d.accno=b.accno and b.branch_name=br.branch_name and br.branch_city ...  1 row(s) returned  0.000 sec / 0.000 sec
82  08:48:09  select customer_name from bankCustomer where customer_name not in (select customer_name from depositor) LIMIT 0, 1000  1 row(s) returned  0.000 sec / 0.000 sec
```

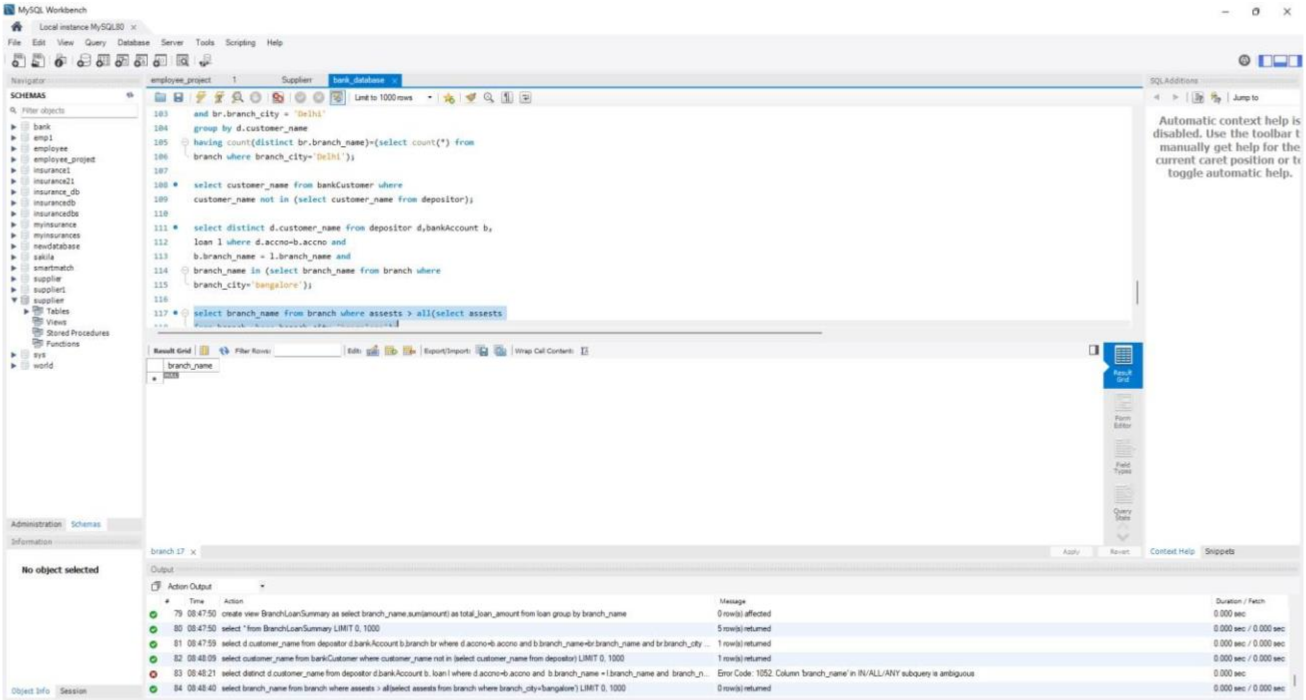
Find all customers who have both an account and a loan at the Bangalore branch.



```sql
create view BranchLoanSummary as
select branch_name,sum(amount) as total_loan_amount
from loan group by branch_name;
select * from BranchLoanSummary;

select d.customer_name from depositor d,bankAccount b,branch br
where d.accno=b.accno
and b.branch_name=br.branch_name
and br.branch_city = 'Delhi'
group by d.customer_name
having count(distinct br.branch_name)=(select count(*) from
branch where branch_city='Delhi');

select customer_name from bankCustomer where
customer_name not in (select customer_name from depositor);

select distinct d.customer_name from depositor d,bankAccount b,
loan l where d.accno=b.accno and
b.branch_name = l.branch_name and
branch_name in (select branch_name from branch where
branch_city='bangalore');

select branch_name from branch where assests > all(select assests
from branch where branch_city='bangalore');

delete from bankAccount where branch_name in (
select branch_name from branch where branch_city='bengaluru');

update bankAccount
set balance=balance*1.05;
```

```
#   Time      Action                                                                                                          Message           Duration / Fetch
78  08:47:37  select customer_name,branch_name from depositor, bankAccount where depositor.accno = bankAccount.accno group by customer_name,branch_na...  0 row(s) returned  0.000 sec / 0.000 sec
79  08:47:50  create view BranchLoanSummary as select branch_name,sum(amount) as total_loan_amount from loan group by branch_name  0 row(s) affected  0.000 sec
80  08:47:50  select * from BranchLoanSummary LIMIT 0, 1000                                                                   5 row(s) returned  0.000 sec / 0.000 sec
81  08:47:59  select d.customer_name from depositor d,bankAccount b,branch br where d.accno=b.accno and b.branch_name=br.branch_name and br.branch_city ...  1 row(s) returned  0.000 sec / 0.000 sec
82  08:48:09  select customer_name from bankCustomer where customer_name not in (select customer_name from depositor) LIMIT 0, 1000  1 row(s) returned  0.000 sec / 0.000 sec
83  08:48:21  select distinct d.customer_name from depositor d,bankAccount b, loan l where d.accno=b.accno and b.branch_name = l.branch_name and branch_n...  Error Code: 1052. Column 'branch_name' in IN/ALL/ANY subquery is ambiguous  0.000 sec
```

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



```sql
    and br.branch_city = 'Delhi'
    group by d.customer_name
    having count(distinct br.branch_name)=(select count(*) from
    branch where branch_city='Delhi');

    select customer_name from bankCustomer where
    customer_name not in (select customer_name from depositor);

    select distinct d.customer_name from depositor d,bankAccount b,
    loan l where d.accno=b.accno and
    b.branch_name = l.branch_name and
    branch_name in (select branch_name from branch where
    branch_city='bangalore');

    select branch_name from branch where assests > all(select assests
```

	branch_name
●	NULL

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

Output

Action Output

```
 #   Time      Action                                                                                                                      Message                                                           Duration / Fetch
79   08:47:50  create view BranchLoanSummary as select branch_name,sum(amount) as total_loan_amount from loan group by branch_name          0 row(s) affected                                                 0.000 sec
80   08:47:50  select * from BranchLoanSummary LIMIT 0, 1000                                                                               5 row(s) returned                                                 0.000 sec / 0.000 sec
81   08:47:59  select d.customer_name from depositor d,bankAccount b,branch br where d.accno=b.accno and b.branch_name=br.branch_name and br.branch_city ...   1 row(s) returned   0.000 sec / 0.000 sec
82   08:48:09  select customer_name from bankCustomer where customer_name not in (select customer_name from depositor) LIMIT 0, 1000       1 row(s) returned                                                 0.000 sec / 0.000 sec
83   08:48:21  select distinct d.customer_name from depositor d,bankAccount b, loan l where d.accno=b.accno and  b.branch_name = l.branch_name and  branch_n...   Error Code: 1052. Column 'branch_name' in IN/ALL/ANY subquery is ambiguous   0.000 sec
84   08:48:40  select branch_name from branch where assests > all(select assests from branch where branch_city='bangalore') LIMIT 0, 1000  0 row(s) returned                                                 0.000 sec / 0.000 sec
```

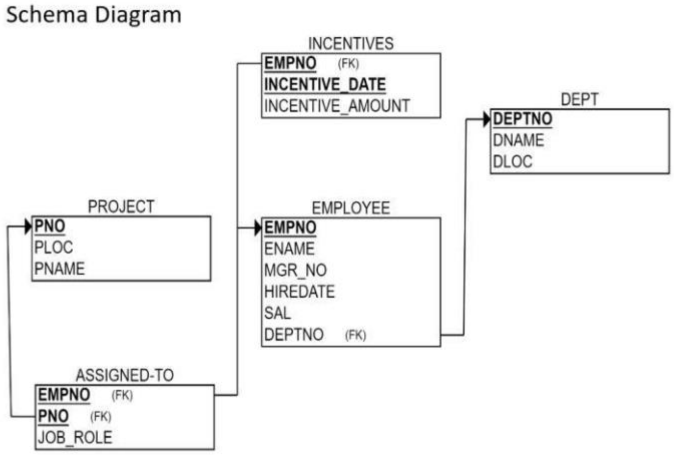
Experiment 5: Employee DATABASE

The employee database must record each employee's identifying number, name, manager reference, hire date, salary, and department affiliation while also tracking departmental details, project assignments (including the role an employee plays on a project), and any incentive payments given to employees. Every employee is represented by a unique employee number and has a hire date and salary that must be valid; the manager field is a self-referencing link that must, if present, point to an existing employee and must never create a circular management chain or reference the employee themself. Departments are identified by a unique department number and include a department name and location; every department referenced by an employee or by other structures must exist in the department table, and departments may contain zero or many employees. Projects are recorded with a unique project number, project name and project location; employees may be assigned to multiple projects and each project may have many employees, with each assignment carrying the employee's job role for that project — duplicate assignments of the same employee to the same project are disallowed. Incentive payments are recorded with the employee reference, the incentive date and the incentive amount; an incentive entry must reference an existing employee and incentive amounts must be non-negative and dated on or after the employee's hire date. Referential integrity must be enforced so that employee records cannot reference non-existent departments, projects, or managers, and assignment and incentive records cannot exist without corresponding employee, project, or department records as appropriate. Salary, incentive amounts, and any monetary fields must be constrained to valid numeric ranges and hire/ incentive dates must be valid calendar dates (and typically not future-dated unless business rules permit). Deletion and update policies must preserve historical consistency: deleting an employee who appears as a manager, as a project assignee, or in incentive records should be prevented or should be handled via controlled archival, reassignment, or soft-delete flags rather than hard deletion to preserve audit trails; similarly, changing a department or project identifier must either be disallowed if it would orphan historical records or handled by introducing immutable surrogate keys. Business rules include preventing circular manager chains, ensuring an employee's manager (if specified) cannot be the employee themself, disallowing duplicate project-assignments, requiring that incentive dates fall within the employee's employment window, and optionally requiring at least one project assignment or at least one incentive record depending on policy for reporting. Implementation should use primary-key and foreign-key constraints for identity and linkage, unique constraints to prevent duplicate assignments, check constraints for monetary and date ranges, and application logic or triggers for complex temporal or graph constraints (like cycle detection in management relationships and enforcing non-overlap or other schedule-related rules if assignments gain temporal attributes later). The system must therefore reliably support queries such as employee reporting lines, department staffing lists, project rosters

with job roles, incentive payment histories, salary analyses, and audit reports while maintaining data integrity, preventing inconsistent deletions, and preserving a complete historical record for HR and compliance needs.

.

Schema Diagram



Create Database

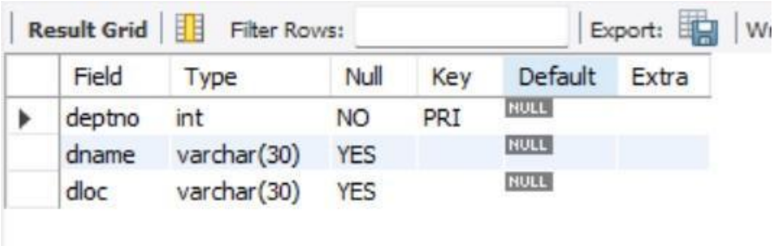
```
create database Employee_project; show
databases;
use employee_project;
```

Create table

```
CREATE TABLE DEPT (
DEPTNO INT PRIMARY KEY,
DNAME VARCHAR(30) NOT NULL,
DLOC VARCHAR(30) NOT NULL
);
desc dept;

CREATE TABLE EMPLOYEE (
   EMPNO INT PRIMARY KEY,
   ENAME VARCHAR(30) NOT NULL,
   MGR_NO INT,
   HIREDATE DATE,
   SAL DECIMAL(10,2),
   DEPTNO INT,
   FOREIGN KEY (DEPTNO) REFERENCES DEPT(DEPTNO)
);
desc employee;
```

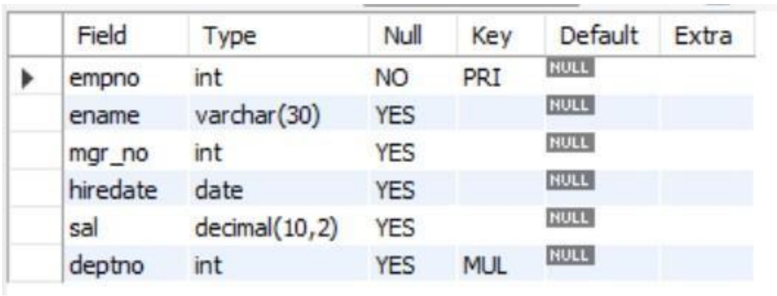
```
CREATE TABLE PROJECT (
   PNO INT PRIMARY KEY,
   PLOC VARCHAR(30) NOT NULL,
   PNAME VARCHAR(30) NOT NULL
);
desc project;

CREATE TABLE ASSIGNED_TO (
   EMPNO INT,
   PNO INT,
   JOB_ROLE VARCHAR(30),
   PRIMARY KEY (EMPNO, PNO),
   FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE(EMPNO),
   FOREIGN KEY (PNO) REFERENCES PROJECT(PNO)
);
desc assigned_to;

CREATE TABLE INCENTIVES (
   EMPNO INT,
   INCENTIVE_DATE DATE,
   INCENTIVE_AMOUNT DECIMAL(10,2),
   PRIMARY KEY (EMPNO, INCENTIVE_DATE),
   FOREIGN KEY (EMPNO) REFERENCES EMPLOYEE(EMPNO) );
```

Structure of the table desc dept;



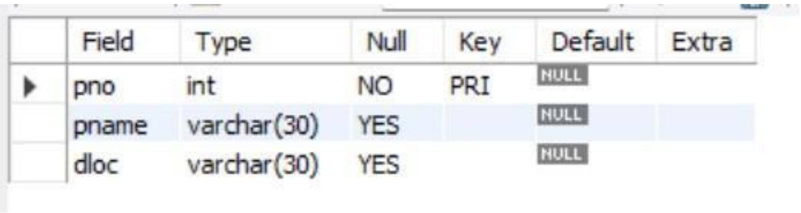
	Field	Type	Null	Key	Default	Extra
▶	deptno	int	NO	PRI	NULL	
	dname	varchar(30)	YES		NULL	
	dloc	varchar(30)	YES		NULL	

desc employeee;



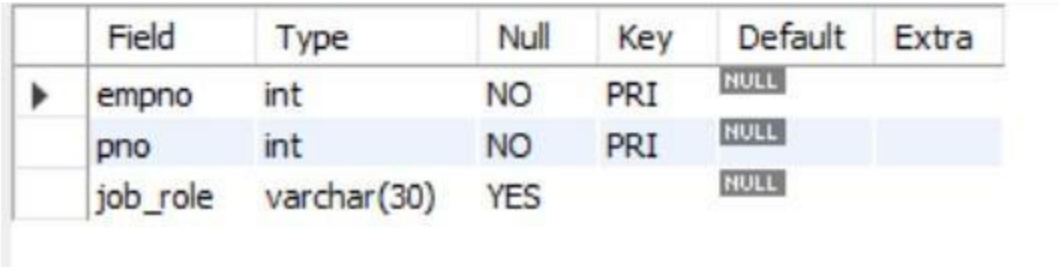
	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	ename	varchar(30)	YES		NULL	
	mgr_no	int	YES		NULL	
	hiredate	date	YES		NULL	
	sal	decimal(10,2)	YES		NULL	
	deptno	int	YES	MUL	NULL	

desc project;



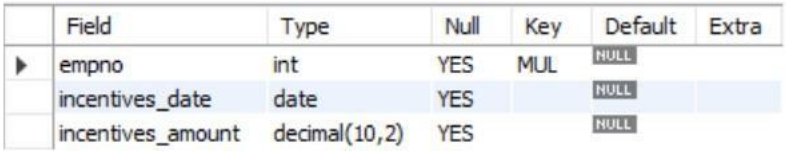
	Field	Type	Null	Key	Default	Extra
▶	pno	int	NO	PRI	NULL	
	pname	varchar(30)	YES		NULL	
	dloc	varchar(30)	YES		NULL	

desc assigned;



	Field	Type	Null	Key	Default	Extra
▶	empno	int	NO	PRI	NULL	
	pno	int	NO	PRI	NULL	
	job_role	varchar(30)	YES		NULL	

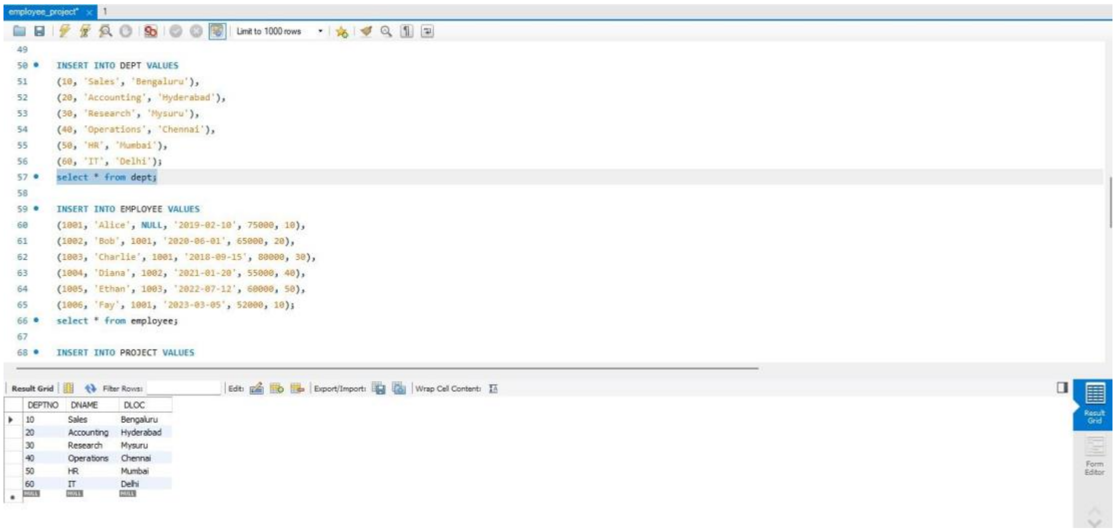
desc incentives;



	Field	Type	Null	Key	Default	Extra
▶	empno	int	YES	MUL	NULL	
	incentives_date	date	YES		NULL	
	incentives_amount	decimal(10,2)	YES		NULL	

Insertion of values into Table

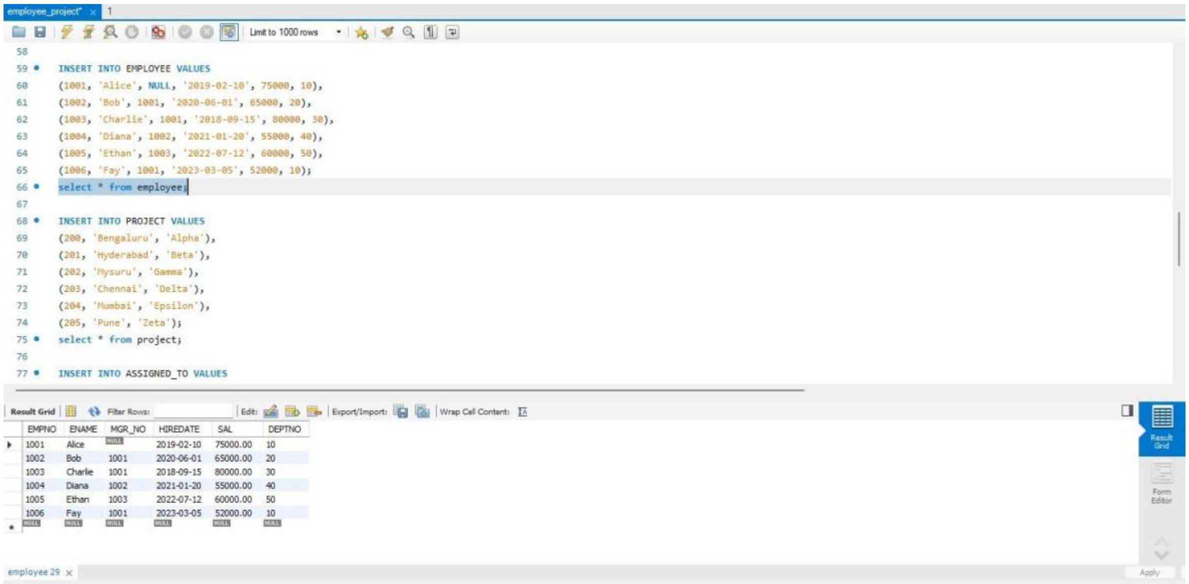
```
INSERT INTO DEPT VALUES
(10, 'Sales', 'Bengaluru'),
(20, 'Accounting', 'Hyderabad'),
(30, 'Research', 'Mysuru'),
(40, 'Operations', 'Chennai'),
(50, 'HR', 'Mumbai'),
(60, 'IT', 'Delhi');
select * from dept;
```



```
INSERT INTO DEPT VALUES
(10, 'Sales', 'Bengaluru'),
(20, 'Accounting', 'Hyderabad'),
(30, 'Research', 'Mysuru'),
(40, 'Operations', 'Chennai'),
(50, 'HR', 'Mumbai'),
(60, 'IT', 'Delhi');
select * from dept;

INSERT INTO EMPLOYEE VALUES
(1001, 'Alice', NULL, '2019-02-10', 75000, 10),
(1002, 'Bob', 1001, '2020-06-01', 65000, 20),
(1003, 'Charlie', 1001, '2018-09-15', 80000, 30),
(1004, 'Diana', 1002, '2021-01-20', 55000, 40),
(1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
(1006, 'Fay', 1001, '2023-03-05', 52000, 10);
select * from employee;

INSERT INTO PROJECT VALUES
```

DEPTNO	DNAME	DLOC
10	Sales	Bengaluru
20	Accounting	Hyderabad
30	Research	Mysuru
40	Operations	Chennai
50	HR	Mumbai
60	IT	Delhi
NULL	NULL	NULL

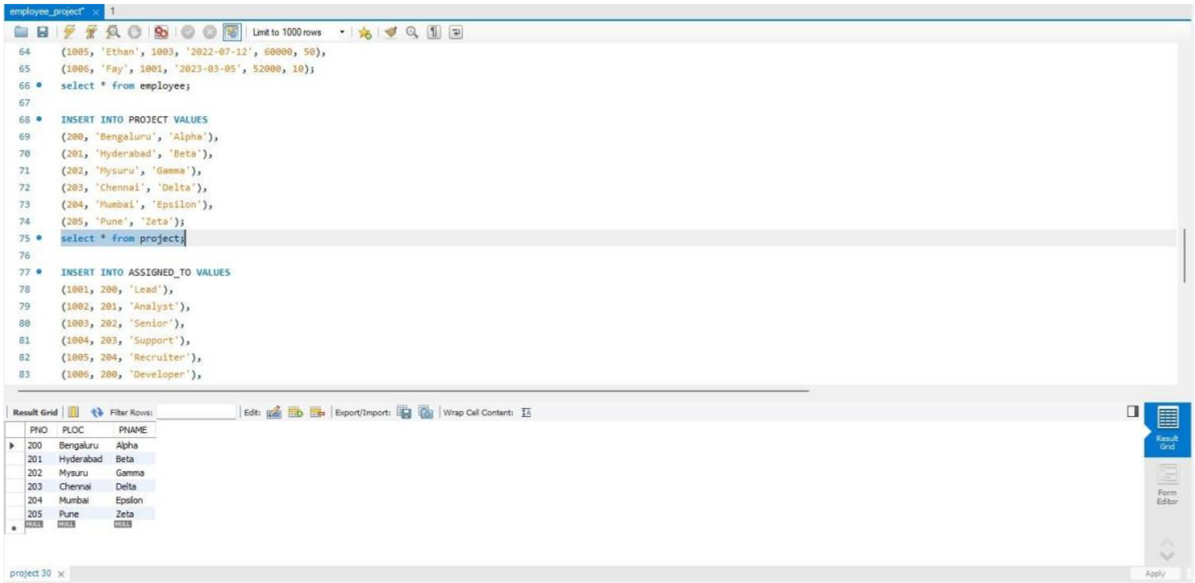
INSERT INTO EMPLOYEE VALUES
(1001, 'Alice', NULL, '2019-02-10', 75000, 10),
(1002, 'Bob', 1001, '2020-06-01', 65000, 20),
(1003, 'Charlie', 1001, '2018-09-15', 80000, 30),
(1004, 'Diana', 1002, '2021-01-20', 55000, 40),
(1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
(1006, 'Fay', 1001, '2023-03-05', 52000, 10);
select * from employee;



```
INSERT INTO EMPLOYEE VALUES
(1001, 'Alice', NULL, '2019-02-10', 75000, 10),
(1002, 'Bob', 1001, '2020-06-01', 65000, 20),
(1003, 'Charlie', 1001, '2018-09-15', 80000, 30),
(1004, 'Diana', 1002, '2021-01-20', 55000, 40),
(1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
(1006, 'Fay', 1001, '2023-03-05', 52000, 10);
select * from employee;

INSERT INTO PROJECT VALUES
(200, 'Bengaluru', 'Alpha'),
(201, 'Hyderabad', 'Beta'),
(202, 'Mysuru', 'Gamma'),
(203, 'Chennai', 'Delta'),
(204, 'Mumbai', 'Epsilon'),
(205, 'Pune', 'Zeta');
select * from project;

INSERT INTO ASSIGNED_TO VALUES
```

EMPNO	ENAME	MGR_NO	HIREDATE	SAL	DEPTNO
1001	Alice	NULL	2019-02-10	75000.00	10
1002	Bob	1001	2020-06-01	65000.00	20
1003	Charlie	1001	2018-09-15	80000.00	30
1004	Diana	1002	2021-01-20	55000.00	40
1005	Ethan	1003	2022-07-12	60000.00	50
1006	Fay	1001	2023-03-05	52000.00	10
NULL	NULL	NULL	NULL	NULL	NULL

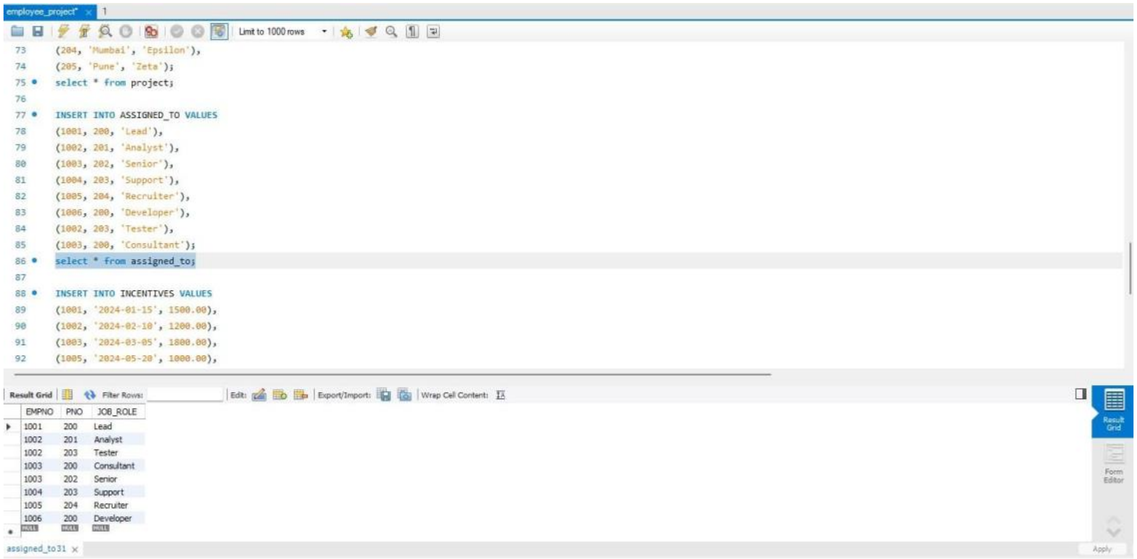
```
INSERT INTO PROJECT VALUES
(200, 'Bengaluru', 'Alpha'),
(201, 'Hyderabad', 'Beta'),
(202, 'Mysuru', 'Gamma'),
(203, 'Chennai', 'Delta'),
(204, 'Mumbai',
'Epsilon'), (205, 'Pune',
'Zeta'); select * from
project;
```



```
(1005, 'Ethan', 1003, '2022-07-12', 60000, 50),
(1006, 'Fay', 1001, '2023-03-05', 52000, 10);
select * from employee;

INSERT INTO PROJECT VALUES
(200, 'Bengaluru', 'Alpha'),
(201, 'Hyderabad', 'Beta'),
(202, 'Mysuru', 'Gamma'),
(203, 'Chennai', 'Delta'),
(204, 'Mumbai', 'Epsilon'),
(205, 'Pune', 'Zeta');
select * from project;

INSERT INTO ASSIGNED_TO VALUES
(1001, 200, 'Lead'),
(1002, 201, 'Analyst'),
(1003, 202, 'Senior'),
(1004, 203, 'Support'),
(1005, 204, 'Recruiter'),
(1006, 200, 'Developer'),
```

PNO	PLOC	PNAME
200	Bengaluru	Alpha
201	Hyderabad	Beta
202	Mysuru	Gamma
203	Chennai	Delta
204	Mumbai	Epsilon
205	Pune	Zeta
NULL	NULL	NULL

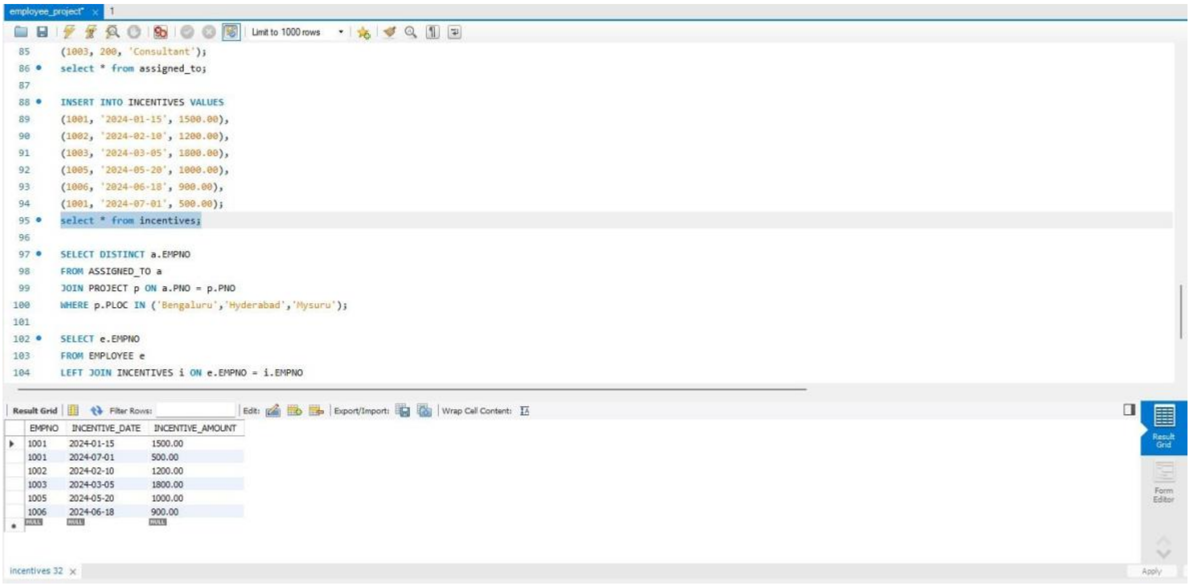
```
INSERT INTO ASSIGNED_TO VALUES
(1001, 200, 'Lead'),
(1002, 201, 'Analyst'),
(1003, 202, 'Senior'),
(1004, 203, 'Support'),
(1005, 204, 'Recruiter'),
(1006, 200, 'Developer'),
(1002, 203, 'Tester'),
(1003, 200, 'Consultant');
select * from assigned_to;
```



```
(204, 'Mumbai', 'Epsilon'),
(205, 'Pune', 'Zeta');
select * from project;

INSERT INTO ASSIGNED_TO VALUES
(1001, 200, 'Lead'),
(1002, 201, 'Analyst'),
(1003, 202, 'Senior'),
(1004, 203, 'Support'),
(1005, 204, 'Recruiter'),
(1006, 200, 'Developer'),
(1002, 203, 'Tester'),
(1003, 200, 'Consultant');
select * from assigned_to;

INSERT INTO INCENTIVES VALUES
(1001, '2024-01-15', 1500.00),
(1002, '2024-02-10', 1200.00),
(1003, '2024-03-05', 1800.00),
(1005, '2024-05-20', 1000.00),
```

EMPNO	PNO	JOB_ROLE
1001	200	Lead
1002	201	Analyst
1002	203	Tester
1003	200	Consultant
1003	202	Senior
1004	203	Support
1005	204	Recruiter
1006	200	Developer
NULL	NULL	NULL

INSERT INTO INCENTIVES VALUES
(1001, '2024-01-15', 1500.00),
(1002, '2024-02-10', 1200.00),
(1003, '2024-03-05', 1800.00),
(1005, '2024-05-20', 1000.00),
(1006, '2024-06-18', 900.00),
(1001, '2024-07-01', 500.00);
select * from incentives;

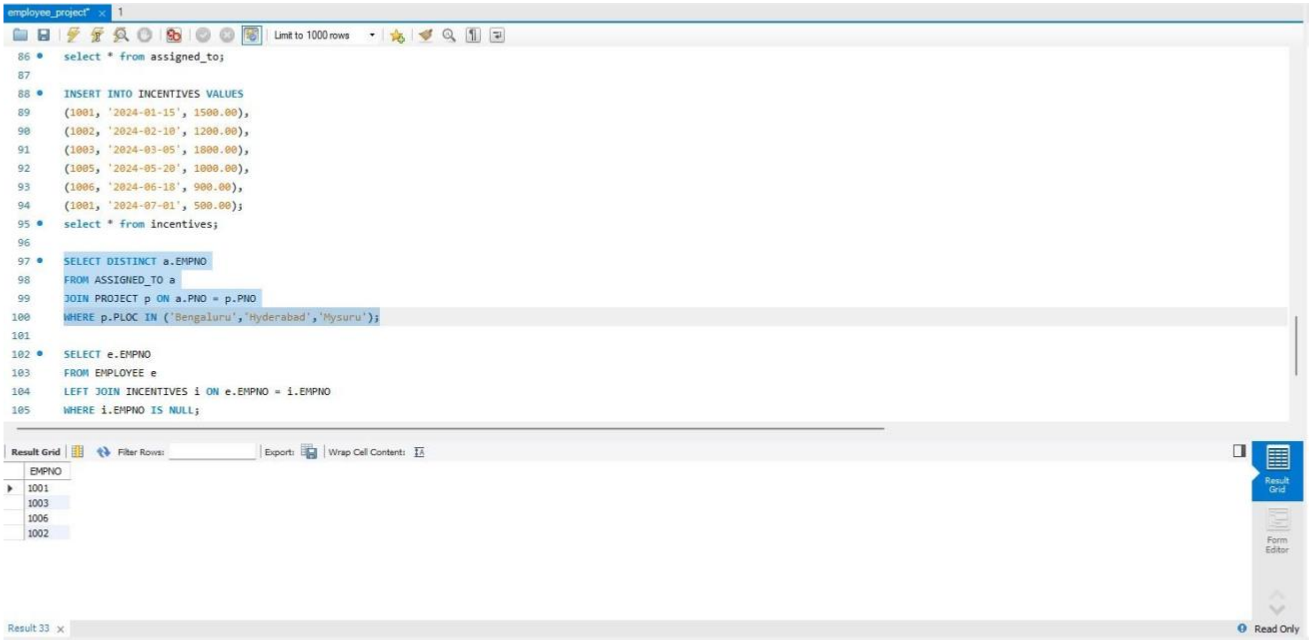


```
(1003, 200, 'Consultant');
select * from assigned_to;

INSERT INTO INCENTIVES VALUES
(1001, '2024-01-15', 1500.00),
(1002, '2024-02-10', 1200.00),
(1003, '2024-03-05', 1800.00),
(1005, '2024-05-20', 1000.00),
(1006, '2024-06-18', 900.00),
(1001, '2024-07-01', 500.00);
select * from incentives;

SELECT DISTINCT a.EMPNO
FROM ASSIGNED_TO a
JOIN PROJECT p ON a.PNO = p.PNO
WHERE p.PLOC IN ('Bengaluru','Hyderabad','Mysuru');

SELECT e.EMPNO
FROM EMPLOYEE e
LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO
```

EMPNO	INCENTIVE_DATE	INCENTIVE_AMOUNT
1001	2024-01-15	1500.00
1001	2024-07-01	500.00
1002	2024-02-10	1200.00
1003	2024-03-05	1800.00
1005	2024-05-20	1000.00
1006	2024-06-18	900.00
NULL	NULL	NULL

Queries

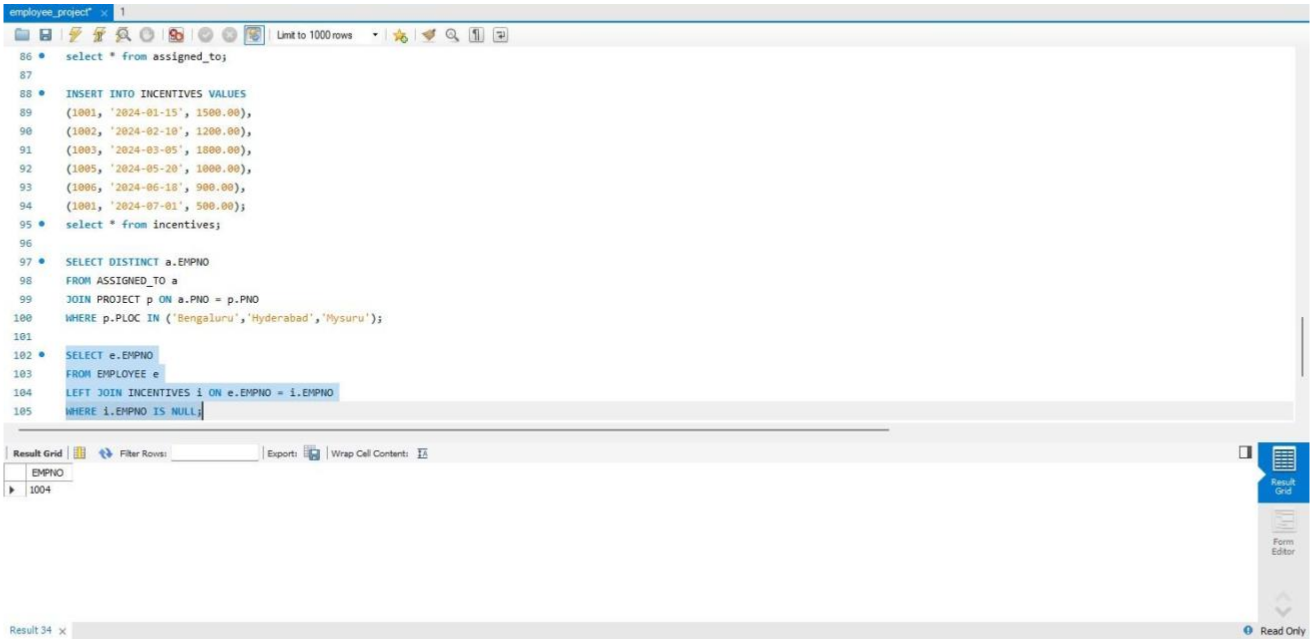
Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru.



```sql
select * from assigned_to;

INSERT INTO INCENTIVES VALUES
(1001, '2024-01-15', 1500.00),
(1002, '2024-02-10', 1200.00),
(1003, '2024-03-05', 1800.00),
(1005, '2024-05-20', 1000.00),
(1006, '2024-06-18', 900.00),
(1001, '2024-07-01', 500.00);
select * from incentives;

SELECT DISTINCT a.EMPNO
FROM ASSIGNED_TO a
JOIN PROJECT p ON a.PNO = p.PNO
WHERE p.PLOC IN ('Bengaluru','Hyderabad','Mysuru');

SELECT e.EMPNO
FROM EMPLOYEE e
LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO
WHERE i.EMPNO IS NULL;
```

```
EMPNO
1001
1003
1006
1002
```

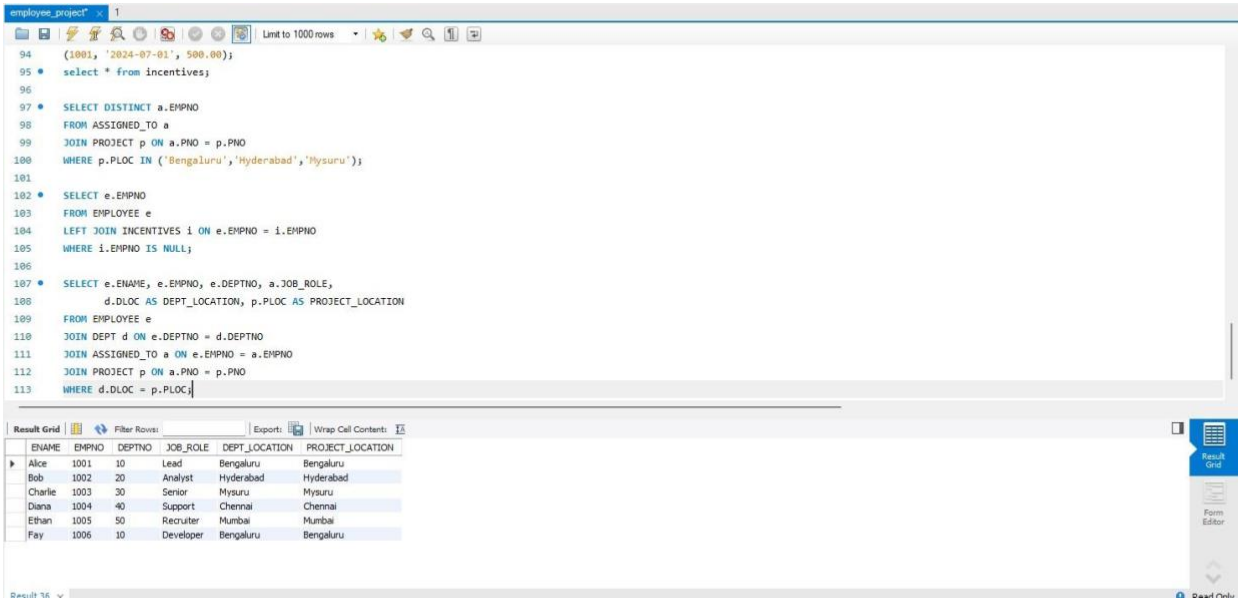
Get Employee ID's of those employees who didn't receive incentives



```sql
select * from assigned_to;

INSERT INTO INCENTIVES VALUES
(1001, '2024-01-15', 1500.00),
(1002, '2024-02-10', 1200.00),
(1003, '2024-03-05', 1800.00),
(1005, '2024-05-20', 1000.00),
(1006, '2024-06-18', 900.00),
(1001, '2024-07-01', 500.00);
select * from incentives;

SELECT DISTINCT a.EMPNO
FROM ASSIGNED_TO a
JOIN PROJECT p ON a.PNO = p.PNO
WHERE p.PLOC IN ('Bengaluru','Hyderabad','Mysuru');

SELECT e.EMPNO
FROM EMPLOYEE e
LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO
WHERE i.EMPNO IS NULL;
```

```
EMPNO
1004
```

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.

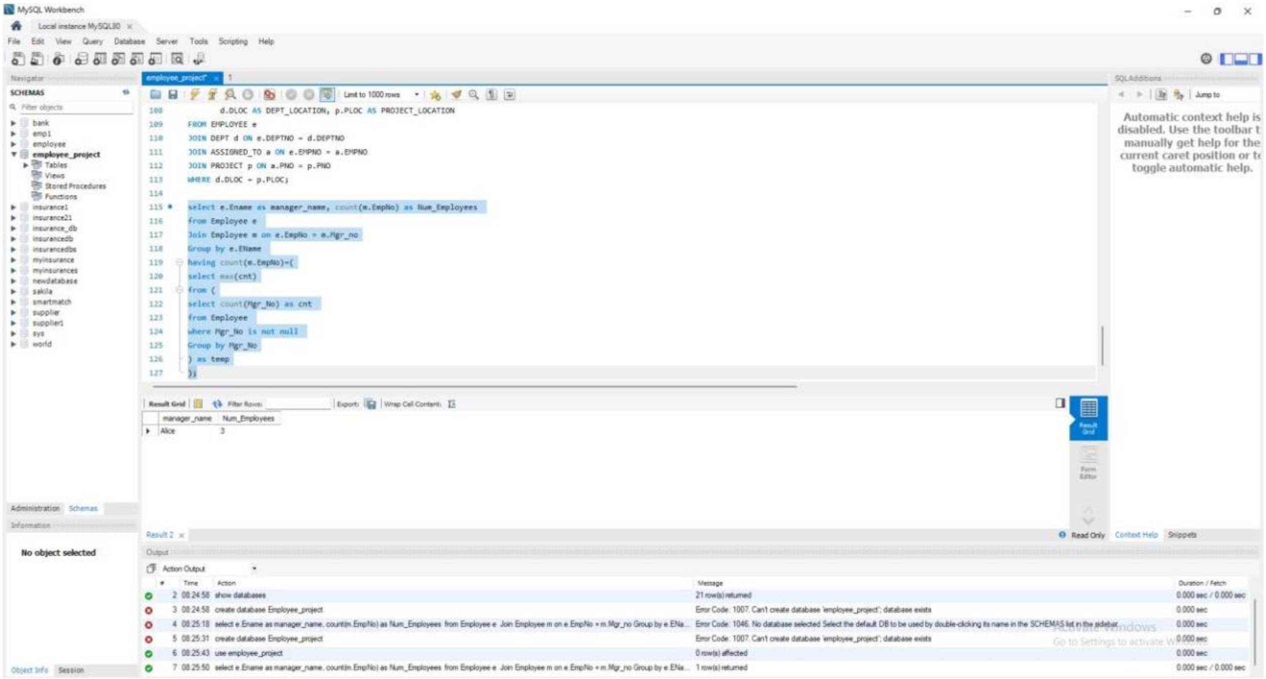


```
(1001, '2024-07-01', 500.00);
select * from incentives;

SELECT DISTINCT a.EMPNO
FROM ASSIGNED_TO a
JOIN PROJECT p ON a.PNO = p.PNO
WHERE p.PLOC IN ('Bengaluru','Hyderabad','Mysuru');

SELECT e.EMPNO
FROM EMPLOYEE e
LEFT JOIN INCENTIVES i ON e.EMPNO = i.EMPNO
WHERE i.EMPNO IS NULL;

SELECT e.ENAME, e.EMPNO, e.DEPTNO, a.JOB_ROLE,
       d.DLOC AS DEPT_LOCATION, p.PLOC AS PROJECT_LOCATION
FROM EMPLOYEE e
JOIN DEPT d ON e.DEPTNO = d.DEPTNO
JOIN ASSIGNED_TO a ON e.EMPNO = a.EMPNO
JOIN PROJECT p ON a.PNO = p.PNO
WHERE d.DLOC = p.PLOC;
```

ENAME	EMPNO	DEPTNO	JOB_ROLE	DEPT_LOCATION	PROJECT_LOCATION
Alice	1001	10	Lead	Bengaluru	Bengaluru
Bob	1002	20	Analyst	Hyderabad	Hyderabad
Charlie	1003	30	Senior	Mysuru	Mysuru
Diana	1004	40	Support	Chennai	Chennai
Ethan	1005	50	Recruiter	Mumbai	Mumbai
Fay	1006	10	Developer	Bengaluru	Bengaluru

Experiment 6: More Queries on Employee Database

List the name of the managers with the maximum employees.

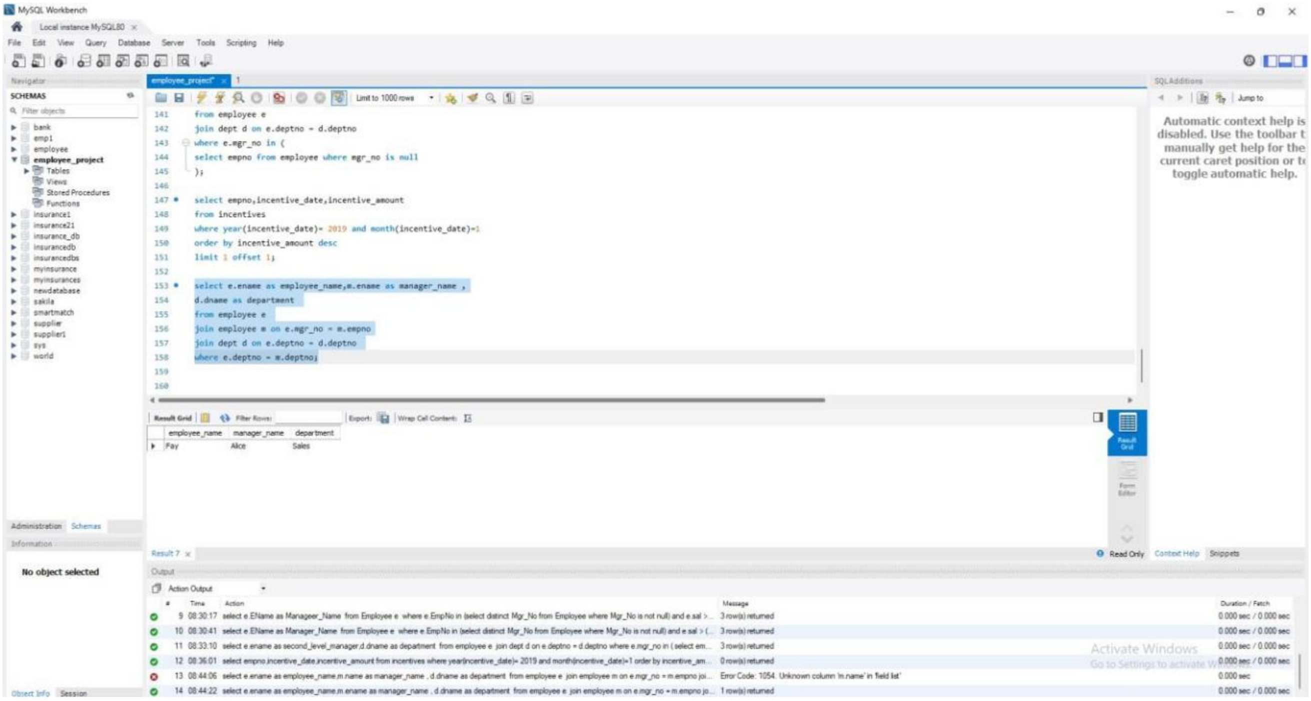


```
        d.DLOC AS DEPT_LOCATION, p.PLOC AS PROJECT_LOCATION
FROM EMPLOYEE e
JOIN DEPT d ON e.DEPTNO = d.DEPTNO
JOIN ASSIGNED_TO a ON e.EMPNO = a.EMPNO
JOIN PROJECT p ON a.PNO = p.PNO
WHERE d.DLOC = p.PLOC;

select e.Ename as manager_name, count(m.EmpNo) as Num_Employees
from Employee e
Join Employee m on e.EmpNo = m.Mgr_no
Group by e.EName
having count(m.EmpNo)=(
select max(cnt)
from (
select count(Mgr_No) as cnt
from Employee
where Mgr_No is not null
Group by Mgr_No
) as temp
);
```

```
manager_name   Num_Employees
Alice          3
```

```
#  Time      Action                                                                                                                     Message                                                                                                        Duration / Fetch
2  08:24:58  show databases                                                                                                             21 row(s) returned                                                                                             0.000 sec / 0.000 sec
3  08:24:58  create database Employee_project                                                                                           Error Code: 1007. Can't create database 'employee_project'; database exists                                    0.000 sec
4  08:25:18  select e.Ename as manager_name, count(m.EmpNo) as Num_Employees from Employee e Join Employee m on e.EmpNo = m.Mgr_no Group by e.ENa...   Error Code: 1046. No database selected Select the default DB to be used by double-clicking its name in the SCHEMAS list in the sidebar.   0.000 sec
5  08:25:31  create database Employee_project                                                                                           Error Code: 1007. Can't create database 'employee_project'; database exists                                    0.000 sec
6  08:25:43  use employee_project                                                                                                       0 row(s) affected                                                                                              0.000 sec
7  08:25:50  select e.Ename as manager_name, count(m.EmpNo) as Num_Employees from Employee e Join Employee m on e.EmpNo = m.Mgr_no Group by e.ENa...   1 row(s) returned                                                                                              0.000 sec / 0.000 sec
```

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

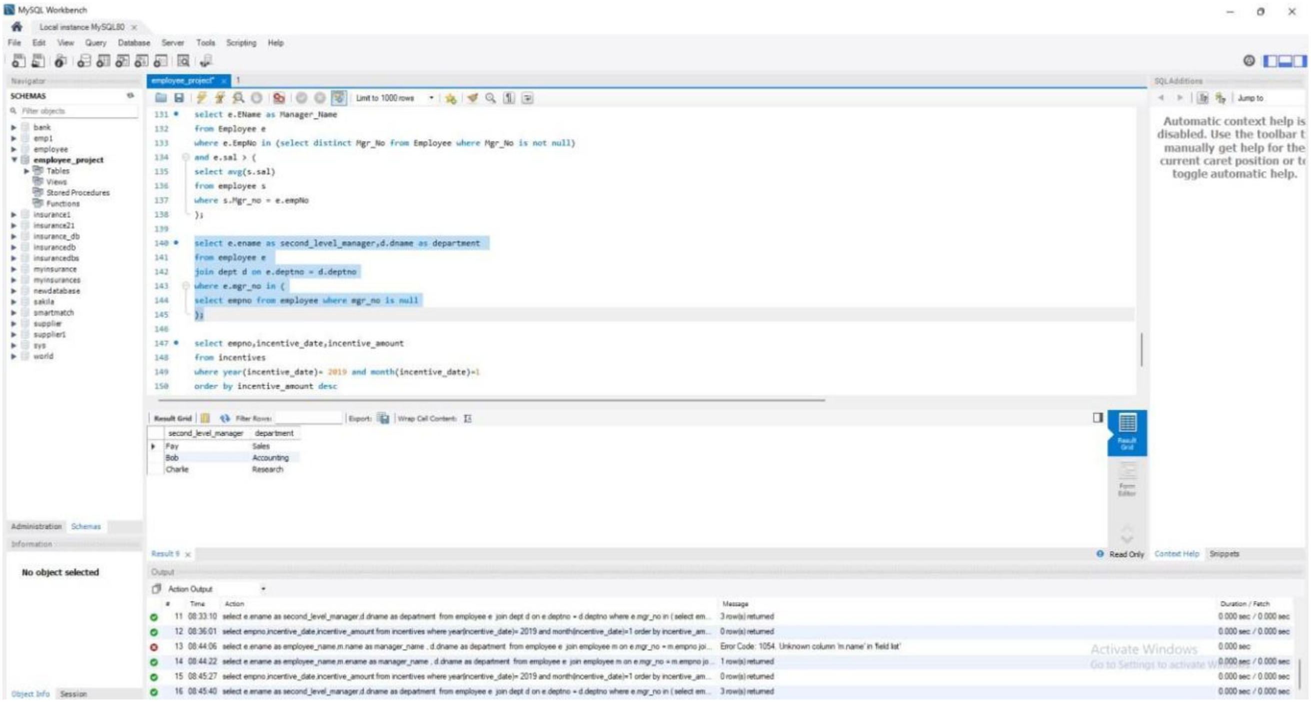


```
from employee e
join dept d on e.deptno = d.deptno
where e.mgr_no in (
select empno from employee where mgr_no is null
);

select empno,incentive_date,incentive_amount
from incentives
where year(incentive_date)= 2019 and month(incentive_date)=1
order by incentive_amount desc
limit 1 offset 1;

select e.ename as employee_name,m.ename as manager_name ,
d.dname as department
from employee e
join employee m on e.mgr_no = m.empno
join dept d on e.deptno = d.deptno
where e.deptno = m.deptno;
```

```
employee_name   manager_name   department
Fay             Alice          Sales
```

```
#   Time      Action                                                                                                                      Message                                                      Duration / Fetch
9   08:30:17  select e.EName as Manageer_Name from Employee e where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null) and e.sal > ...   3 row(s) returned                      0.000 sec / 0.000 sec
10  08:30:41  select e.EName as Manager_Name from Employee e where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null) and e.sal > (...   3 row(s) returned                      0.000 sec / 0.000 sec
11  08:33:10  select e.ename as second_level_manager,d.dname as department from employee e join dept d on e.deptno = d.deptno where e.mgr_no in ( select em...   3 row(s) returned                      0.000 sec / 0.000 sec
12  08:36:01  select empno,incentive_date,incentive_amount from incentives where year(incentive_date)= 2019 and month(incentive_date)=1 order by incentive_am...   0 row(s) returned                      0.000 sec / 0.000 sec
13  08:44:06  select e.ename as employee_name,m.name as manager_name , d.dname as department from employee e join employee m on e.mgr_no = m.empno joi...   Error Code: 1054. Unknown column 'm.name' in 'field list'   0.000 sec
14  08:44:22  select e.ename as employee_name,m.ename as manager_name , d.dname as department from employee e join employee m on e.mgr_no = m.empno jo...   1 row(s) returned                      0.000 sec / 0.000 sec
```

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

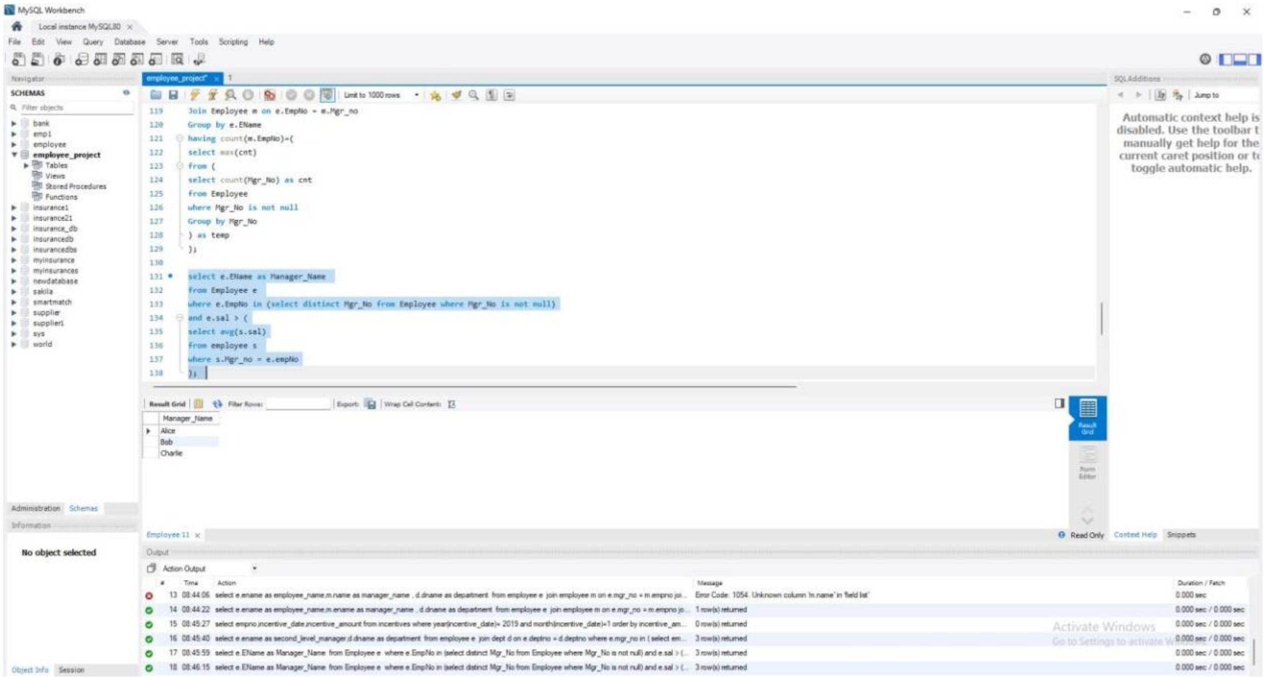


```sql
select e.EName as Manager_Name
from Employee e
where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null)
and e.sal > (
select avg(s.sal)
from employee s
where s.Mgr_no = e.empNo
);

select e.ename as second_level_manager,d.dname as department
from employee e
join dept d on e.deptno = d.deptno
where e.mgr_no in (
select empno from employee where mgr_no is null
);

select empno,incentive_date,incentive_amount
from incentives
where year(incentive_date)= 2019 and month(incentive_date)=1
order by incentive_amount desc
```

second_level_manager	department
Fay	Sales
Bob	Accounting
Charlie	Research

```
#   Time      Action                                                                                                                                          Message                                                  Duration / Fetch
11  08:33:10  select e.ename as second_level_manager,d.dname as department  from employee e  join dept d on e.deptno = d.deptno where e.mgr_no in ( select em...   3 row(s) returned                                        0.000 sec / 0.000 sec
12  08:36:01  select empno,incentive_date,incentive_amount from incentives where year(incentive_date)= 2019 and month(incentive_date)=1 order by incentive_am...   0 row(s) returned                                        0.000 sec / 0.000 sec
13  08:44:06  select e.ename as employee_name,m.name as manager_name , d.dname as department  from employee e  join employee m on e.mgr_no = m.empno joi...     Error Code: 1054. Unknown column 'm.name' in 'field list'   0.000 sec
14  08:44:22  select e.ename as employee_name,m.ename as manager_name , d.dname as department  from employee e  join employee m on e.mgr_no = m.empno jo...    1 row(s) returned                                        0.000 sec / 0.000 sec
15  08:45:27  select empno,incentive_date,incentive_amount from incentives where year(incentive_date)= 2019 and month(incentive_date)=1 order by incentive_am...   0 row(s) returned                                        0.000 sec / 0.000 sec
16  08:45:40  select e.ename as second_level_manager,d.dname as department  from employee e  join dept d on e.deptno = d.deptno where e.mgr_no in ( select em...   3 row(s) returned                                        0.000 sec / 0.000 sec
```

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



```sql
Join Employee m on e.EmpNo = m.Mgr_no
Group by e.EName
having count(m.EmpNo)=(
select max(cnt)
from (
select count(Mgr_No) as cnt
from Employee
where Mgr_No is not null
Group by Mgr_No
) as temp
);

select e.EName as Manager_Name
from Employee e
where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null)
and e.sal > (
select avg(s.sal)
from employee s
where s.Mgr_no = e.empNo
);
```

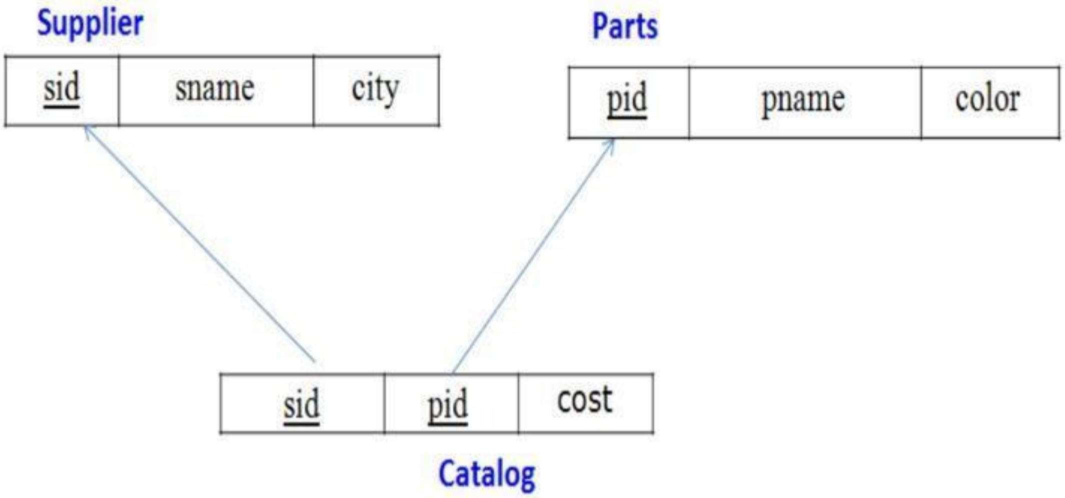
Manager_Name
Alice
Bob
Charlie

```
#   Time      Action                                                                                                                                          Message                                                  Duration / Fetch
13  08:44:06  select e.ename as employee_name,m.name as manager_name , d.dname as department  from employee e  join employee m on e.mgr_no = m.empno joi...     Error Code: 1054. Unknown column 'm.name' in 'field list'   0.000 sec
14  08:44:22  select e.ename as employee_name,m.ename as manager_name , d.dname as department  from employee e  join employee m on e.mgr_no = m.empno jo...    1 row(s) returned                                        0.000 sec / 0.000 sec
15  08:45:27  select empno,incentive_date,incentive_amount from incentives where year(incentive_date)= 2019 and month(incentive_date)=1 order by incentive_am...   0 row(s) returned                                        0.000 sec / 0.000 sec
16  08:45:40  select e.ename as second_level_manager,d.dname as department  from employee e  join dept d on e.deptno = d.deptno where e.mgr_no in ( select em...   3 row(s) returned                                        0.000 sec / 0.000 sec
17  08:45:59  select e.EName as Manager_Name  from Employee e  where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null) and e.sal > (...  3 row(s) returned                                        0.000 sec / 0.000 sec
18  08:46:15  select e.EName as Manager_Name  from Employee e  where e.EmpNo in (select distinct Mgr_No from Employee where Mgr_No is not null) and e.sal > (...  3 row(s) returned                                        0.000 sec / 0.000 sec
```

Experiment no7 Supplier Database

The supplier database must store information about suppliers, the parts they provide, and the prices at which each part is offered so that purchasing, analysis, and reporting can be done accurately. Each supplier is uniquely identified by a supplier ID and is recorded with a name and the city in which the supplier is located; each part is uniquely identified by a part ID and includes a part name and a color. The system must maintain a catalog that links suppliers to the parts they supply and records the cost at which a given supplier sells a given part. Every catalog entry must reference an existing supplier and an existing part, and there must be no duplicate entries for the same combination of supplier and part, so that at most one current price record exists per supplier–part pair. Costs must be valid numeric values and strictly non-negative, and business rules may specify upper limits or currency formats that must be enforced consistently. The data model must support the possibility that a supplier can provide many different parts, that a part can be supplied by many different suppliers, and that some suppliers or parts may temporarily have no catalog entries if they are inactive or not currently traded. Referential integrity must be enforced so that a supplier or part cannot be deleted while still referenced in the catalog unless such deletion is handled by controlled archival or cascade rules that preserve historical price information; in general, historical catalog data should not be lost, as it may be required for audits or trend analysis. The system should allow queries such as "find all suppliers for a given part," "list all parts provided by a given supplier," "retrieve the cheapest supplier for each part," and "analyze supplier coverage by city," and must therefore guarantee that identifiers are unique, relationships between suppliers, parts, and catalog entries are consistent, and price information is accurate and reliably maintained over time.

Schema Diagram



Create database

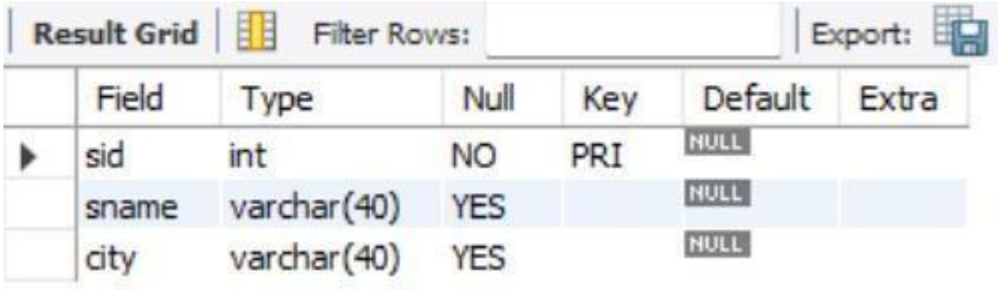
```
create database if not exists Supplierr ; use
supplierr;
```

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 INT,
    PRIMARY KEY (sid, pid),
    FOREIGN KEY (sid) REFERENCES Supplier(sid),
    FOREIGN KEY (pid) REFERENCES Parts(pid)
);
```

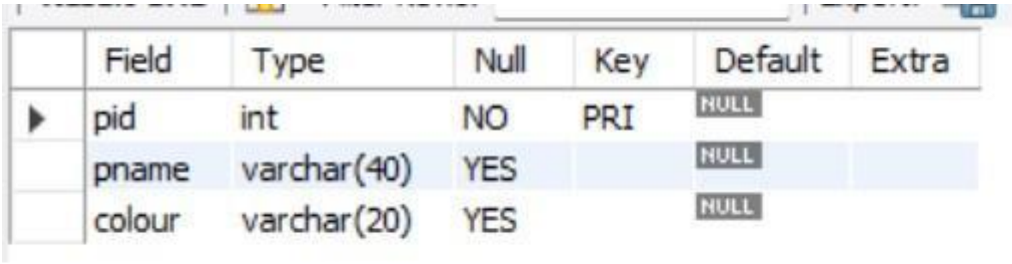
Structure of the table

desc supplier;



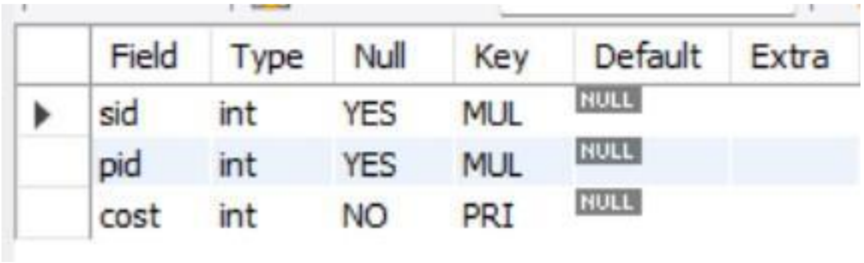
Field	Type	Null	Key	Default	Extra
sid	int	NO	PRI	NULL	
sname	varchar(40)	YES		NULL	
city	varchar(40)	YES		NULL	

desc parts;



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

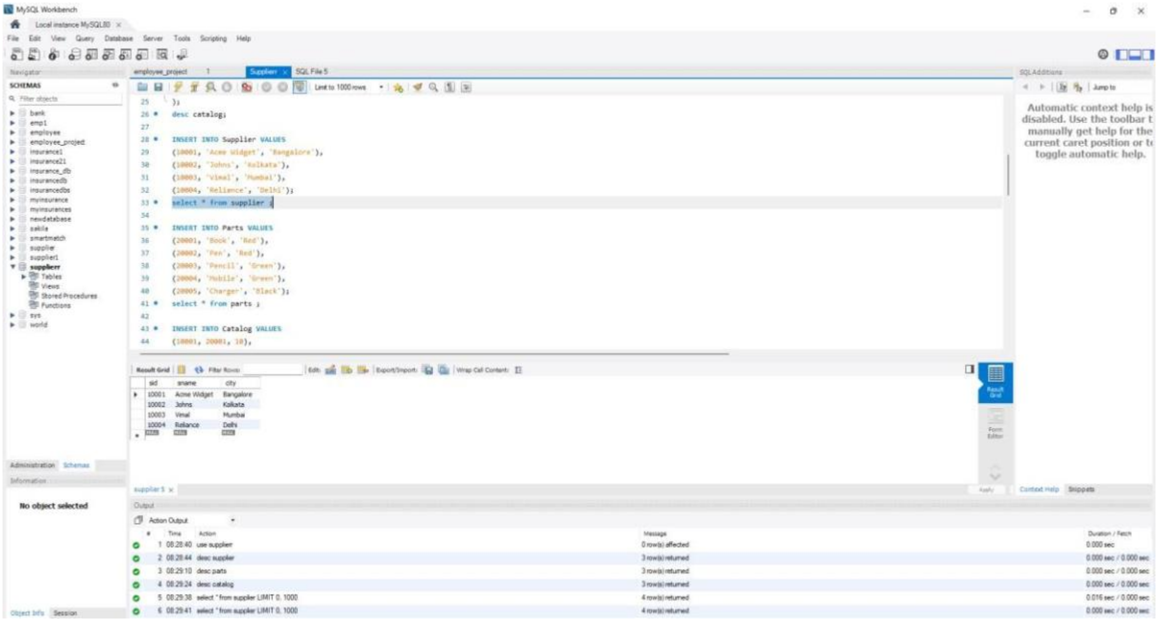
desc catalog;



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

Insertion of values into the table

INSERT INTO Supplier VALUES
(10001, 'Acme Widget', 'Bangalore'),
(10002, 'Johns', 'Kolkata'),
(10003, 'Vimal', 'Mumbai'),
(10004, 'Reliance', 'Delhi');



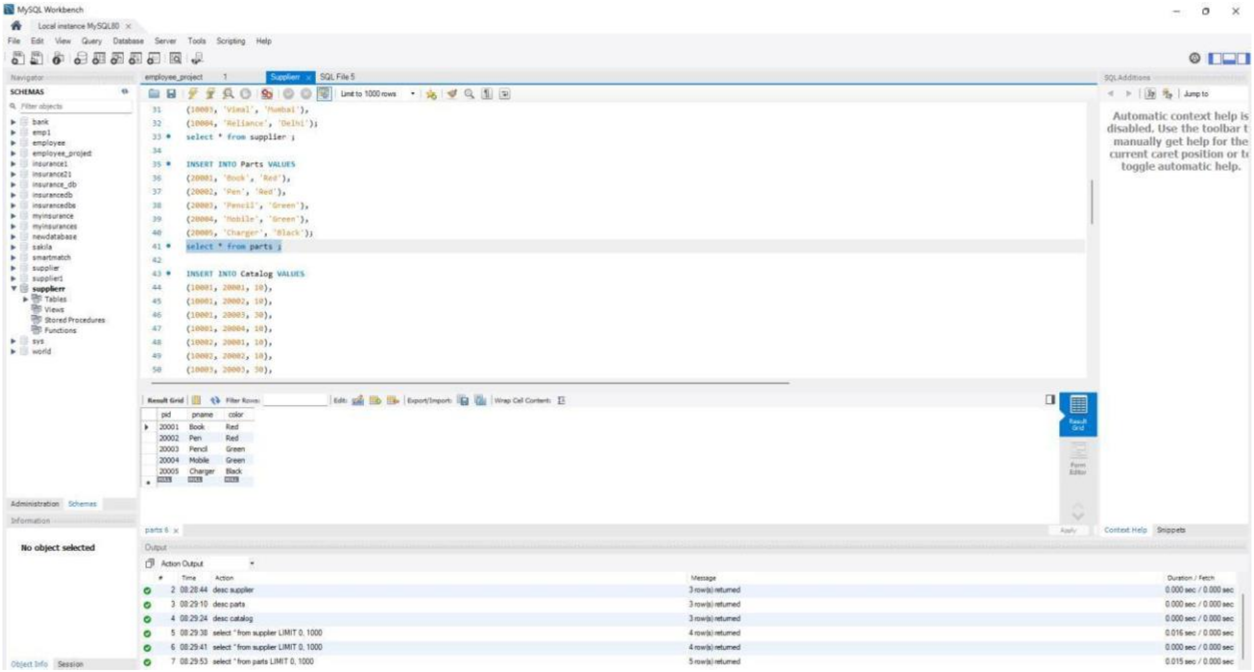
```
);
desc catalog;

INSERT INTO Supplier VALUES
(10001, 'Acme Widget', 'Bangalore'),
(10002, 'Johns', 'Kolkata'),
(10003, 'Vimal', 'Mumbai'),
(10004, 'Reliance', 'Delhi');
select * from supplier ;

INSERT INTO Parts VALUES
(20001, 'Book', 'Red'),
(20002, 'Pen', 'Red'),
(20003, 'Pencil', 'Green'),
(20004, 'Mobile', 'Green'),
(20005, 'Charger', 'Black');
select * from parts ;

INSERT INTO Catalog VALUES
(10001, 20001, 10),
```

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

```
#  Time      Action                                  Message              Duration / Fetch
1  08:28:40  use supplier                            0 row(s) affected    0.000 sec
2  08:28:44  desc supplier                           3 row(s) returned    0.000 sec / 0.000 sec
3  08:29:10  desc parts                              3 row(s) returned    0.000 sec / 0.000 sec
4  08:29:24  desc catalog                            3 row(s) returned    0.000 sec / 0.000 sec
5  08:29:38  select * from supplier LIMIT 0, 1000    4 row(s) returned    0.016 sec / 0.000 sec
6  08:29:41  select * from supplier LIMIT 0, 1000    4 row(s) returned    0.000 sec / 0.000 sec
```

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

INSERT INTO Parts VALUES
(20001, 'Book', 'Red'),
(20002, 'Pen', 'Red'),
(20003, 'Pencil', 'Green'),
(20004, 'Mobile', 'Green'),
(20005, 'Charger', 'Black');



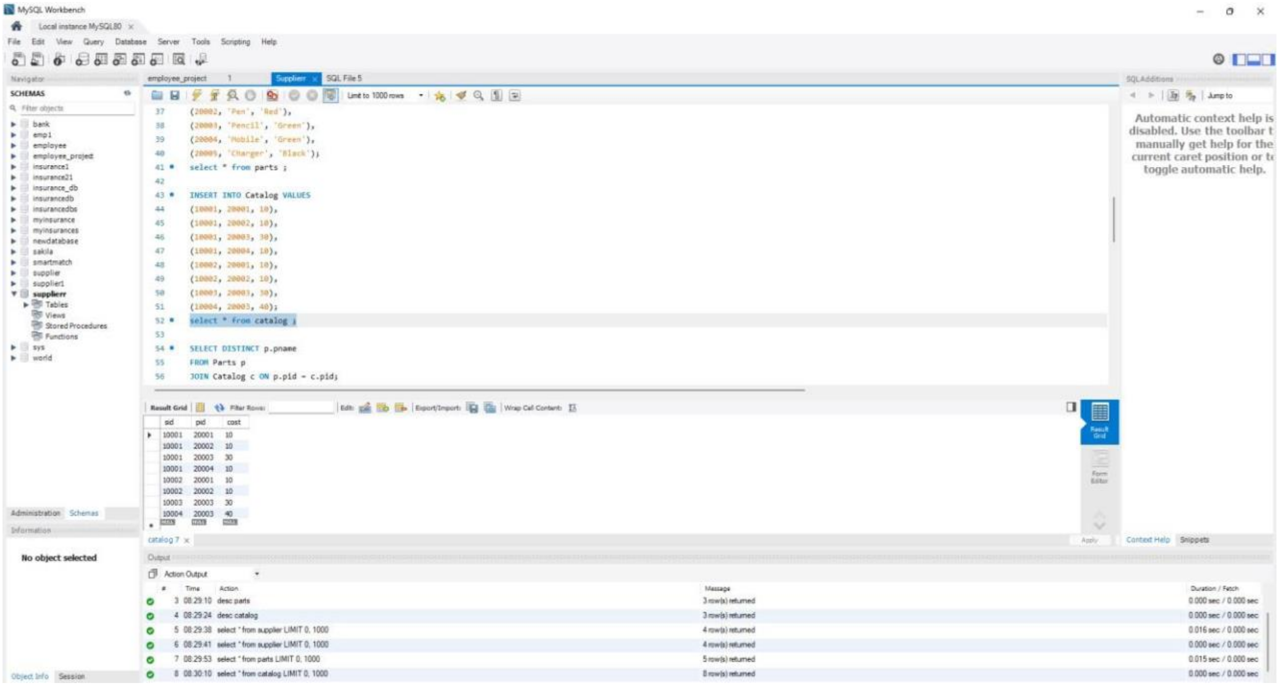
```
(10003, 'Vimal', 'Mumbai'),
(10004, 'Reliance', 'Delhi');
select * from supplier ;

INSERT INTO Parts VALUES
(20001, 'Book', 'Red'),
(20002, 'Pen', 'Red'),
(20003, 'Pencil', 'Green'),
(20004, 'Mobile', 'Green'),
(20005, 'Charger', 'Black');
select * from parts ;

INSERT INTO Catalog VALUES
(10001, 20001, 10),
(10001, 20002, 10),
(10001, 20003, 30),
(10001, 20004, 10),
(10002, 20001, 10),
(10002, 20002, 10),
(10003, 20003, 30),
```

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

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

```
#  Time      Action                                 Message            Duration / Fetch
2  08:28:44  desc supplier                          3 row(s) returned  0.000 sec / 0.000 sec
3  08:29:10  desc parts                             3 row(s) returned  0.000 sec / 0.000 sec
4  08:29:24  desc catalog                           3 row(s) returned  0.000 sec / 0.000 sec
5  08:29:38  select * from supplier LIMIT 0, 1000   4 row(s) returned  0.016 sec / 0.000 sec
6  08:29:41  select * from supplier LIMIT 0, 1000   4 row(s) returned  0.000 sec / 0.000 sec
7  08:29:53  select * from parts LIMIT 0, 1000      5 row(s) returned  0.015 sec / 0.000 sec
```

```
INSERT INTO Catalog VALUES
(10001, 20001, 10),
(10001, 20002, 10),
(10001, 20003, 30),
(10001, 20004, 10),
(10002, 20001, 10),
(10002, 20002, 10),
(10003, 20003, 30),
(10004, 20003, 40);
```



```
(20002, 'Pen', 'Red'),
(20003, 'Pencil', 'Green'),
(20004, 'Mobile', 'Green'),
(20005, 'Charger', 'Black');
select * from parts ;

INSERT INTO Catalog VALUES
(10001, 20001, 10),
(10001, 20002, 10),
(10001, 20003, 30),
(10001, 20004, 10),
(10002, 20001, 10),
(10002, 20002, 10),
(10003, 20003, 30),
(10004, 20003, 40);
select * from catalog ;

SELECT DISTINCT p.pname
FROM Parts p
JOIN Catalog c ON p.pid = c.pid;
```

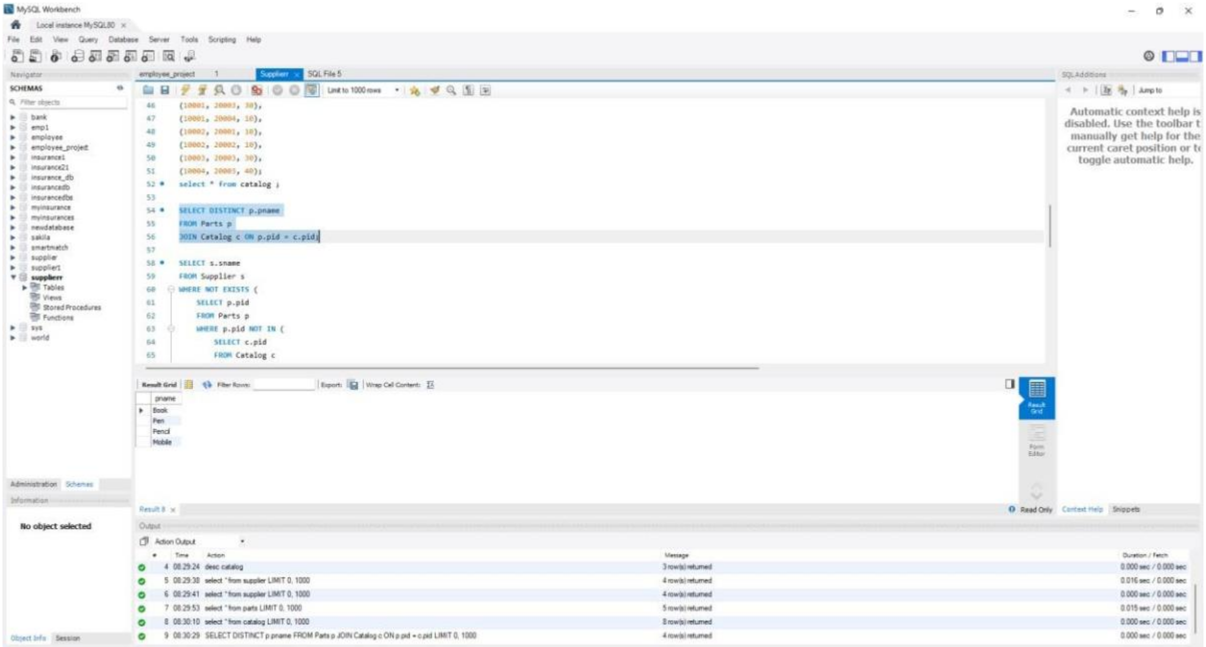
sid	pid	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10002	20001	10
10002	20002	10
10003	20003	30
10004	20003	40
NULL	NULL	NULL

Automatic context help is disabled. Use the toolbar t manually get help for the current caret position or t toggle automatic help.

```
#  Time      Action                                  Message           Duration / Fetch
3  08:29:10  desc parts                              3 row(s) returned  0.000 sec / 0.000 sec
4  08:29:24  desc catalog                            3 row(s) returned  0.000 sec / 0.000 sec
5  08:29:38  select * from supplier LIMIT 0, 1000    4 row(s) returned  0.016 sec / 0.000 sec
6  08:29:41  select * from supplier LIMIT 0, 1000    4 row(s) returned  0.000 sec / 0.000 sec
7  08:29:53  select * from parts LIMIT 0, 1000       5 row(s) returned  0.015 sec / 0.000 sec
8  08:30:10  select * from catalog LIMIT 0, 1000     8 row(s) returned  0.000 sec / 0.000 sec
```

Queries

Find the pnames of parts for which there is some supplier

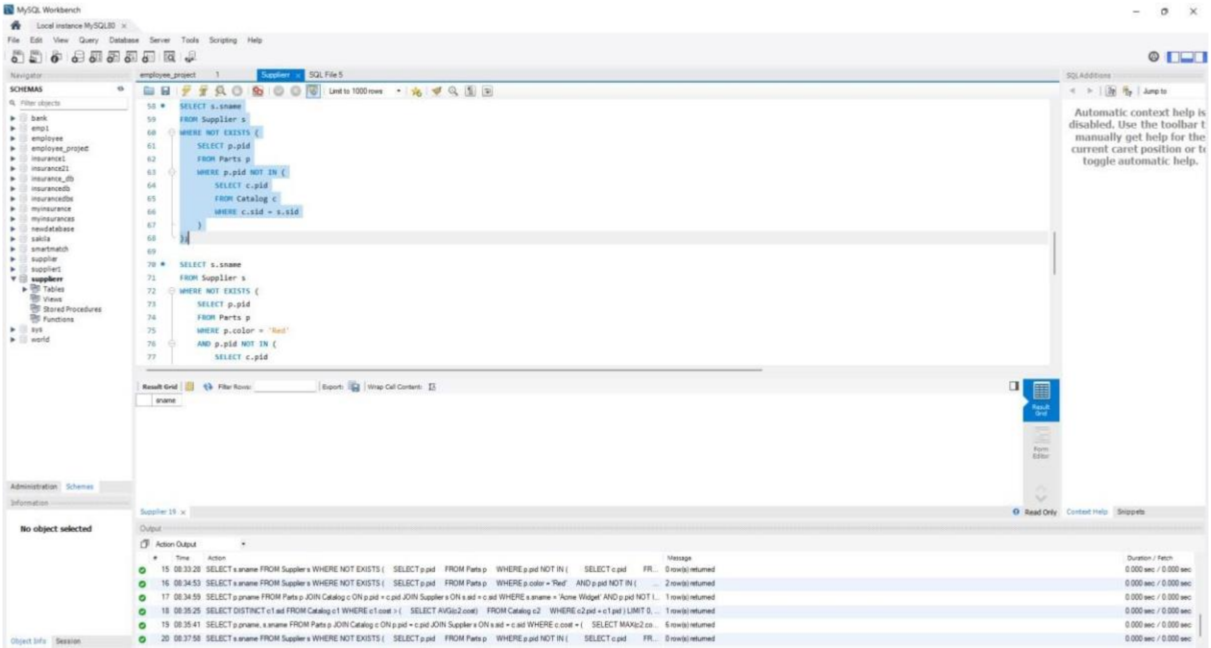


```
(10001, 20003, 30),
(10001, 20004, 10),
(10002, 20001, 10),
(10002, 20002, 10),
(10003, 20003, 30),
(10004, 20005, 40);
select * from catalog ;

SELECT DISTINCT p.pname
FROM Parts p
JOIN Catalog c ON p.pid = c.pid;

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
```

pname
Book
Pen
Pencil
Mobile

```
 #  Time      Action                                                                      Message            Duration / Fetch
 4  08:29:24  desc catalog                                                                3 row(s) returned  0.000 sec / 0.000 sec
 5  08:29:38  select * from supplier LIMIT 0, 1000                                        4 row(s) returned  0.016 sec / 0.000 sec
 6  08:29:41  select * from supplier LIMIT 0, 1000                                        4 row(s) returned  0.000 sec / 0.000 sec
 7  08:29:53  select * from parts LIMIT 0, 1000                                           5 row(s) returned  0.015 sec / 0.000 sec
 8  08:30:10  select * from catalog LIMIT 0, 1000                                         8 row(s) returned  0.000 sec / 0.000 sec
 9  08:30:29  SELECT DISTINCT p.pname FROM Parts p JOIN Catalog c ON p.pid = c.pid LIMIT 0, 1000  4 row(s) returned  0.000 sec / 0.000 sec
```

Find the snames of suppliers who supply every part

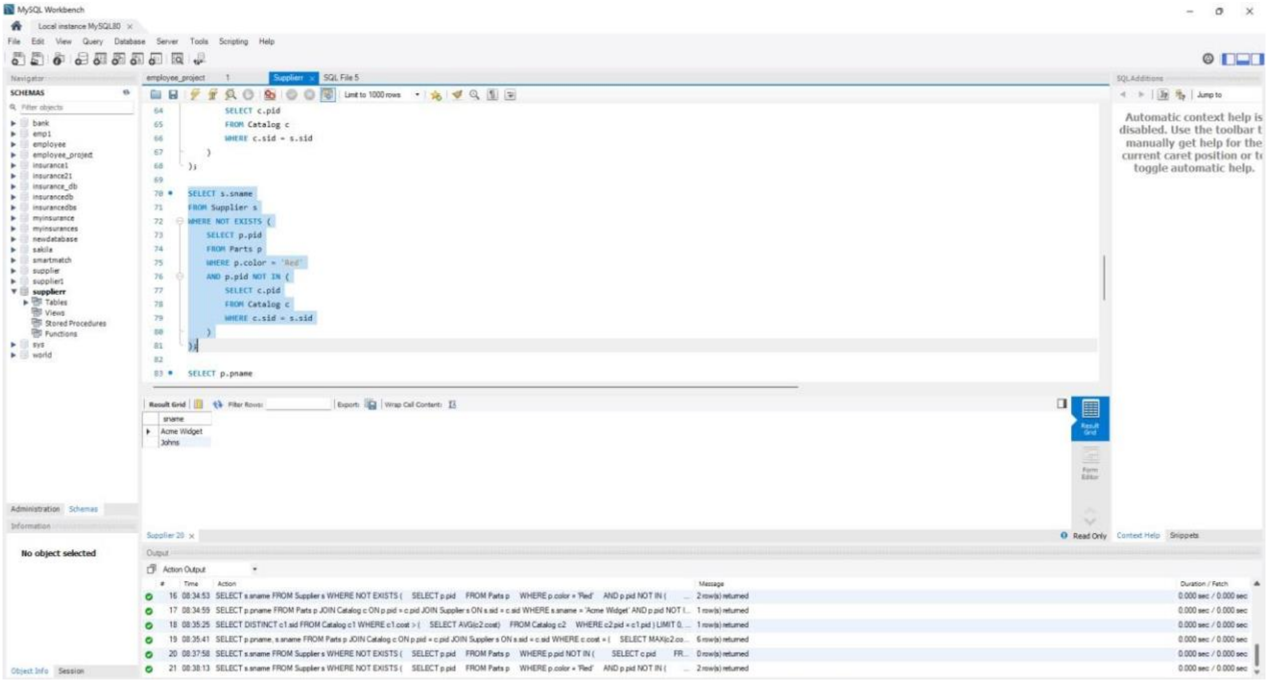


```
SELECT s.sname
FROM Supplier s
WHERE NOT EXISTS (
    SELECT p.pid
    FROM Parts p
    WHERE p.pid NOT IN (
        SELECT c.pid
        FROM Catalog c
        WHERE c.sid = s.sid
    )
);

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
```

sname

```
 #   Time      Action                                                                                                              Message            Duration / Fetch
 15  08:33:20  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (    SELECT p.pid    FROM Parts p    WHERE p.pid NOT IN (      SELECT c.pid      FR...   0 row(s) returned  0.000 sec / 0.000 sec
 16  08:34:53  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (    SELECT p.pid    FROM Parts p    WHERE p.color = 'Red'    AND p.pid NOT IN (      ...   2 row(s) returned  0.000 sec / 0.000 sec
 17  08:34:59  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' AND p.pid NOT I...   1 row(s) returned  0.000 sec / 0.000 sec
 18  08:35:25  SELECT DISTINCT c1.sid FROM Catalog c1 WHERE c1.cost > (    SELECT AVG(c2.cost)    FROM Catalog c2    WHERE c2.pid = c1.pid ) LIMIT 0, ...   1 row(s) returned  0.000 sec / 0.000 sec
 19  08:35:41  SELECT p.pname, s.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.sid WHERE c.cost = (    SELECT MAX(c2.co...   6 row(s) returned  0.000 sec / 0.000 sec
 20  08:37:58  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (    SELECT p.pid    FROM Parts p    WHERE p.pid NOT IN (      SELECT c.pid      FR...   0 row(s) returned  0.000 sec / 0.000 sec
```

Find the snames of suppliers who supply every red part.

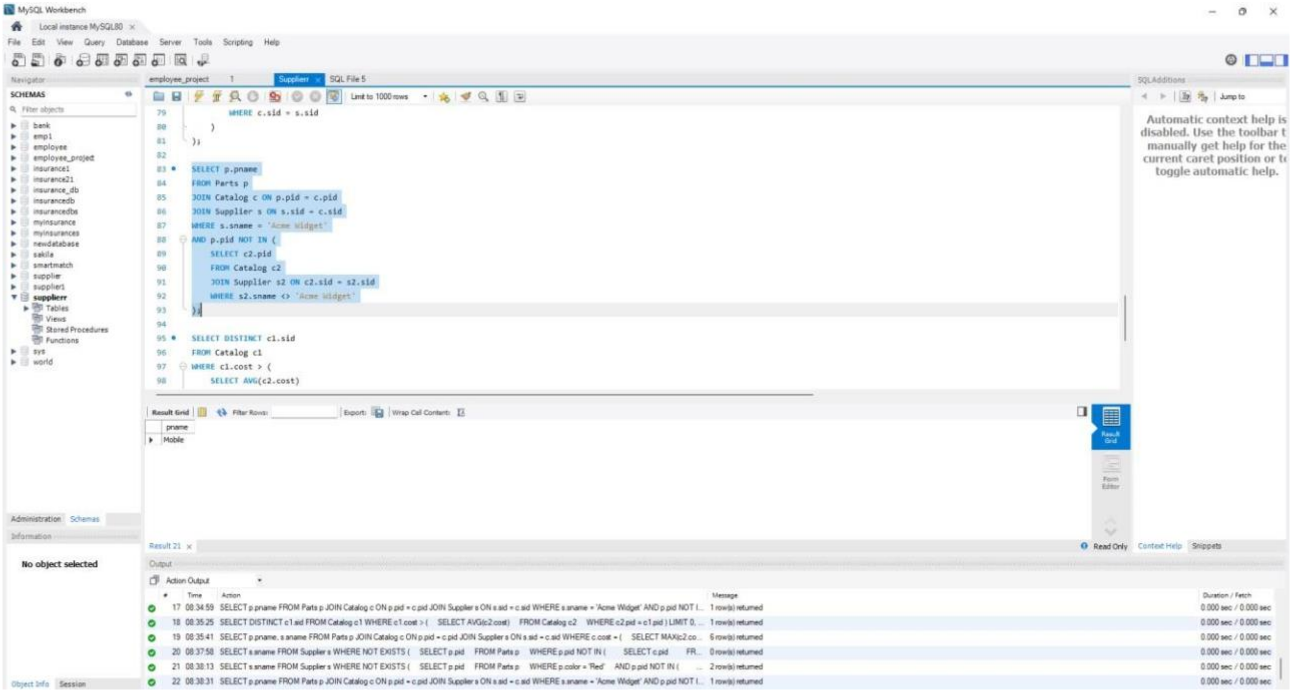


```
        SELECT c.pid
        FROM Catalog c
        WHERE c.sid = s.sid
    )
);

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
    )
);

SELECT p.pname
```

sname
Acme Widget
Johns

```
Action Output
#   Time      Action                                                                                                                    Message             Duration / Fetch
16  08:34:53  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.color = 'Red'   AND p.pid NOT IN (   ...   2 row(s) returned   0.000 sec / 0.000 sec
17  08:34:59  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' AND p.pid NOT I...   1 row(s) returned   0.000 sec / 0.000 sec
18  08:35:25  SELECT DISTINCT c1.sid FROM Catalog c1 WHERE c1.cost > (   SELECT AVG(c2.cost)   FROM Catalog c2   WHERE c2.pid = c1.pid ) LIMIT 0, ...   1 row(s) returned   0.000 sec / 0.000 sec
19  08:35:41  SELECT p.pname, s.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.sid WHERE c.cost = (   SELECT MAX(c2.co...   6 row(s) returned   0.000 sec / 0.000 sec
20  08:37:58  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.pid NOT IN (   SELECT c.pid   FR...   0 row(s) returned   0.000 sec / 0.000 sec
21  08:38:13  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.color = 'Red'   AND p.pid NOT IN (   ...   2 row(s) returned   0.000 sec / 0.000 sec
```

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

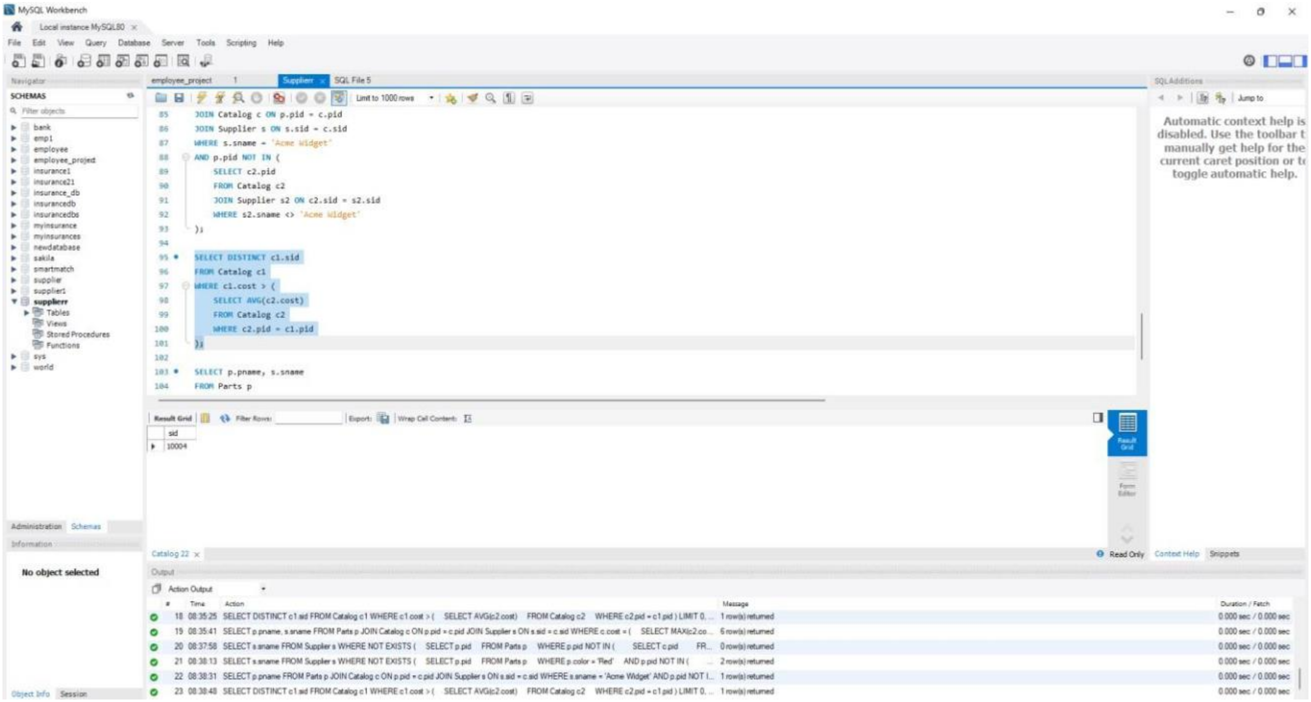


```
            WHERE c.sid = s.sid
    )
);

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'
AND p.pid NOT IN (
    SELECT c2.pid
    FROM Catalog c2
    JOIN Supplier s2 ON c2.sid = s2.sid
    WHERE s2.sname <> 'Acme Widget'
);

SELECT DISTINCT c1.sid
FROM Catalog c1
WHERE c1.cost > (
    SELECT AVG(c2.cost)
```

pname
Mobile

```
Action Output
#   Time      Action                                                                                                                    Message             Duration / Fetch
17  08:34:59  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' AND p.pid NOT I...   1 row(s) returned   0.000 sec / 0.000 sec
18  08:35:25  SELECT DISTINCT c1.sid FROM Catalog c1 WHERE c1.cost > (   SELECT AVG(c2.cost)   FROM Catalog c2   WHERE c2.pid = c1.pid ) LIMIT 0, ...   1 row(s) returned   0.000 sec / 0.000 sec
19  08:35:41  SELECT p.pname, s.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.sid WHERE c.cost = (   SELECT MAX(c2.co...   6 row(s) returned   0.000 sec / 0.000 sec
20  08:37:58  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.pid NOT IN (   SELECT c.pid   FR...   0 row(s) returned   0.000 sec / 0.000 sec
21  08:38:13  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.color = 'Red'   AND p.pid NOT IN (   ...   2 row(s) returned   0.000 sec / 0.000 sec
22  08:38:31  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' AND p.pid NOT I...   1 row(s) returned   0.000 sec / 0.000 sec
```

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

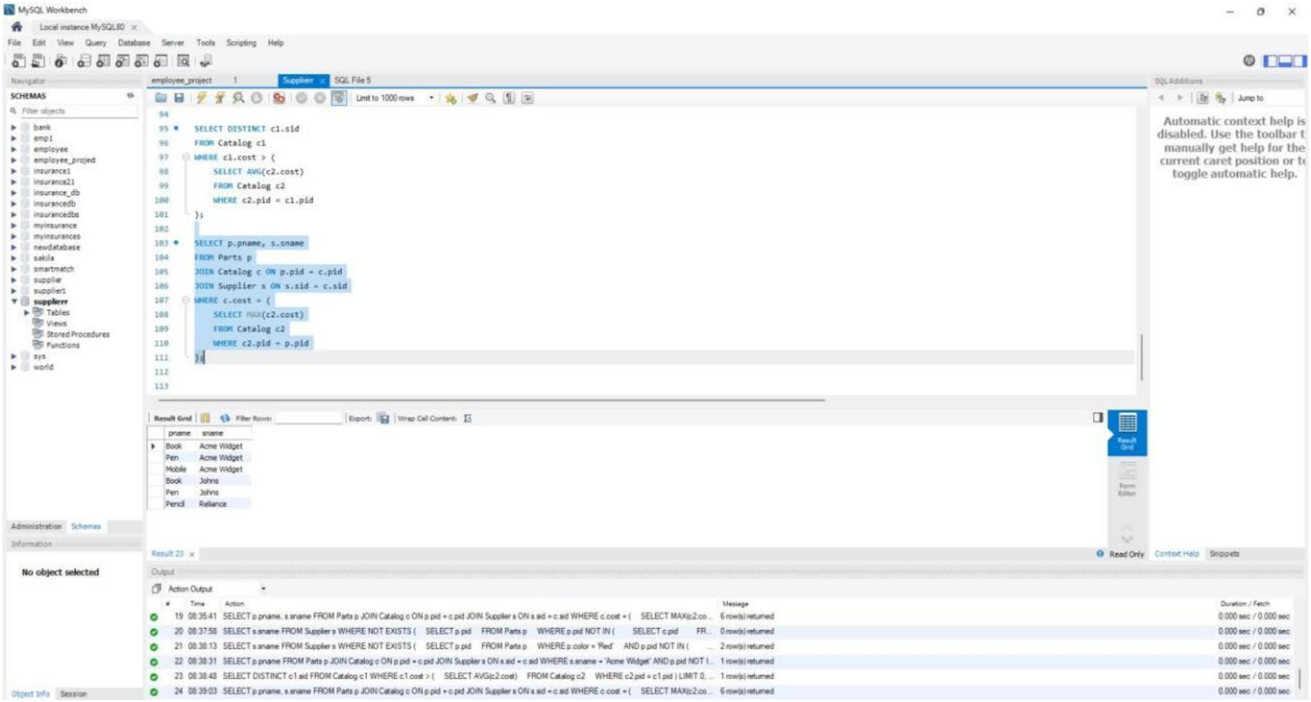


```sql
JOIN Catalog c ON p.pid = c.pid
JOIN Supplier s ON s.sid = c.sid
WHERE s.sname = 'Acme Widget'
AND p.pid NOT IN (
    SELECT c2.pid
    FROM Catalog c2
    JOIN Supplier s2 ON c2.sid = s2.sid
    WHERE s2.sname <> 'Acme Widget'
);

SELECT DISTINCT c1.sid
FROM Catalog c1
WHERE c1.cost > (
    SELECT AVG(c2.cost)
    FROM Catalog c2
    WHERE c2.pid = c1.pid
);

SELECT p.pname, s.sname
FROM Parts p
```

sid
10004

```
#   Time      Action                                                                                                              Message            Duration / Fetch
18  08:35:25  SELECT DISTINCT c1.sid FROM Catalog c1 WHERE c1.cost > (   SELECT AVG(c2.cost)   FROM Catalog c2   WHERE c2.pid = c1.pid ) LIMIT 0, ...   1 row(s) returned   0.000 sec / 0.000 sec
19  08:35:41  SELECT p.pname, s.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.sid WHERE c.cost = (   SELECT MAX(c2.co...   6 row(s) returned   0.000 sec / 0.000 sec
20  08:37:58  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.pid NOT IN (   SELECT c.pid   FR...   0 row(s) returned   0.000 sec / 0.000 sec
21  08:38:13  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.color = 'Red'   AND p.pid NOT IN (   ...   2 row(s) returned   0.000 sec / 0.000 sec
22  08:38:31  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' AND p.pid NOT I...   1 row(s) returned   0.000 sec / 0.000 sec
23  08:38:48  SELECT DISTINCT c1.sid FROM Catalog c1 WHERE c1.cost > (   SELECT AVG(c2.cost)   FROM Catalog c2   WHERE c2.pid = c1.pid ) LIMIT 0, ...   1 row(s) returned   0.000 sec / 0.000 sec
```

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



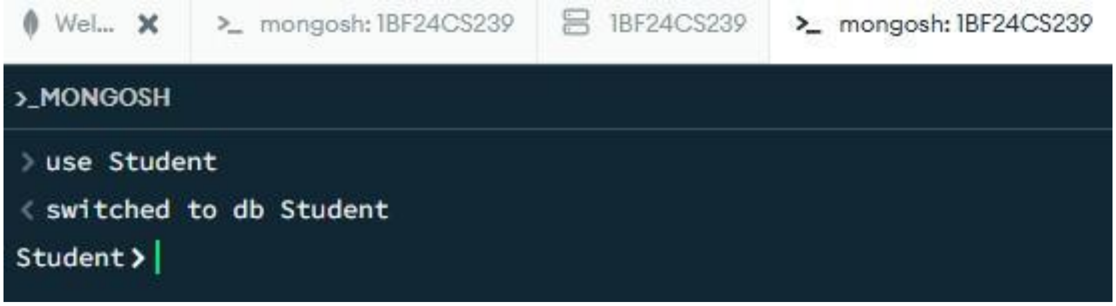
```sql
SELECT DISTINCT c1.sid
FROM Catalog c1
WHERE c1.cost > (
    SELECT AVG(c2.cost)
    FROM Catalog c2
    WHERE c2.pid = c1.pid
);

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

pname	sname
Book	Acme Widget
Pen	Acme Widget
Mobile	Acme Widget
Book	Johns
Pen	Johns
Pencil	Reliance

```
#   Time      Action                                                                                                              Message            Duration / Fetch
19  08:35:41  SELECT p.pname, s.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.sid WHERE c.cost = (   SELECT MAX(c2.co...   6 row(s) returned   0.000 sec / 0.000 sec
20  08:37:58  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.pid NOT IN (   SELECT c.pid   FR...   0 row(s) returned   0.000 sec / 0.000 sec
21  08:38:13  SELECT s.sname FROM Supplier s WHERE NOT EXISTS (   SELECT p.pid   FROM Parts p   WHERE p.color = 'Red'   AND p.pid NOT IN (   ...   2 row(s) returned   0.000 sec / 0.000 sec
22  08:38:31  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' AND p.pid NOT I...   1 row(s) returned   0.000 sec / 0.000 sec
23  08:38:48  SELECT DISTINCT c1.sid FROM Catalog c1 WHERE c1.cost > (   SELECT AVG(c2.cost)   FROM Catalog c2   WHERE c2.pid = c1.pid ) LIMIT 0, ...   1 row(s) returned   0.000 sec / 0.000 sec
24  08:39:03  SELECT p.pname, s.sname FROM Parts p JOIN Catalog c ON p.pid = c.pid JOIN Supplier s ON s.sid = c.sid WHERE c.cost = (   SELECT MAX(c2.co...   6 row(s) returned   0.000 sec / 0.000 sec
```

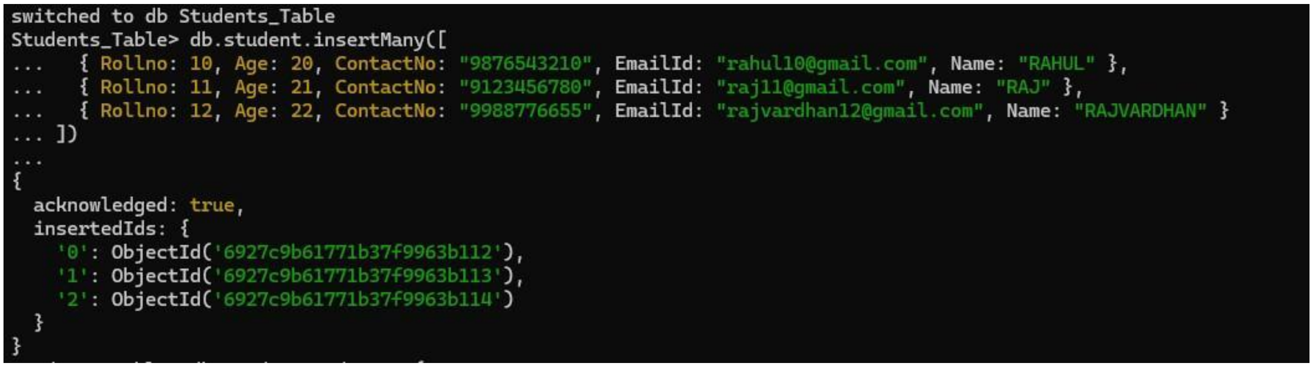
Experiment no8 : Nosql lab1(Student)

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

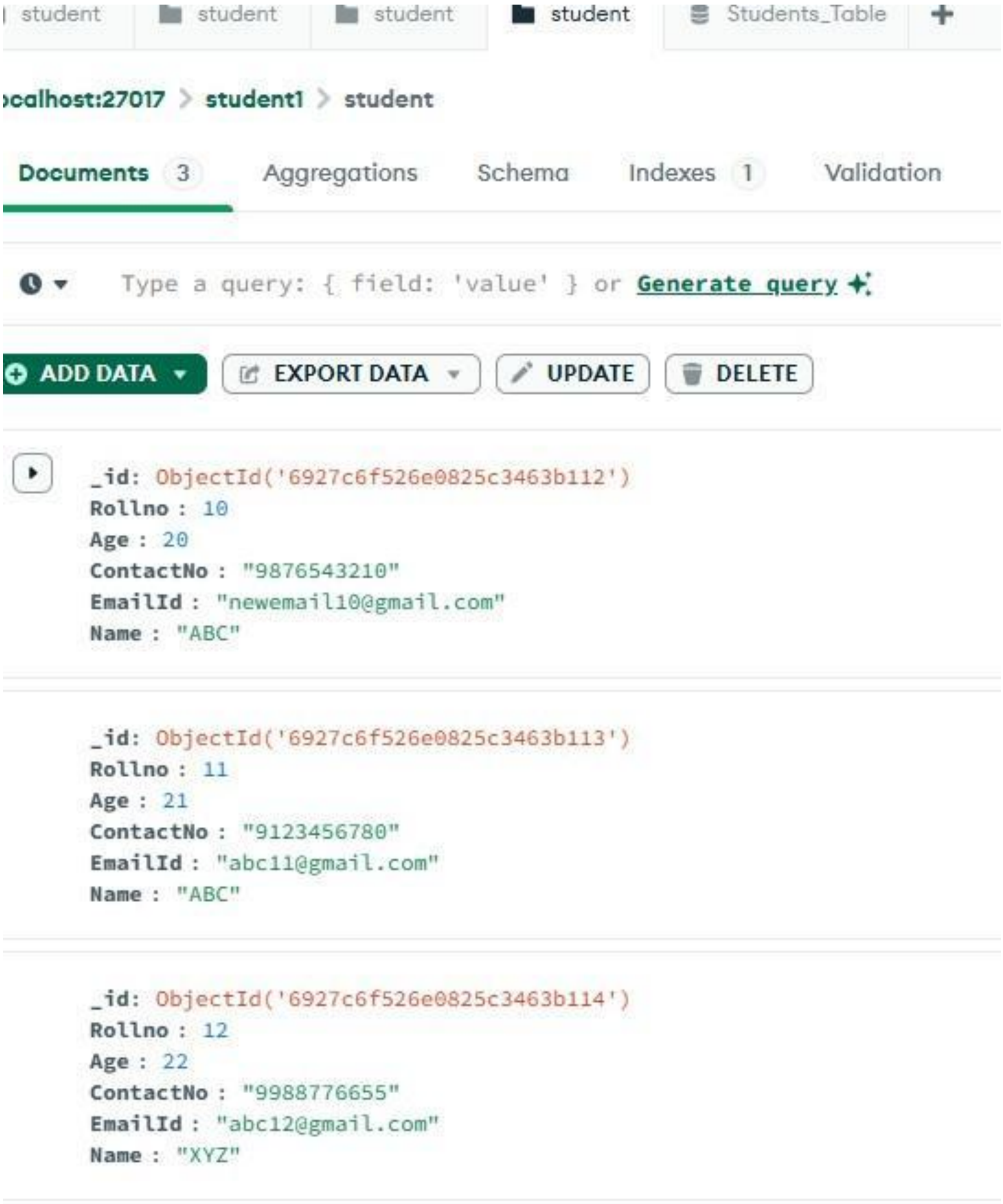


```
> use Student
< switched to db Student
Student >
```

Insert appropriate values.

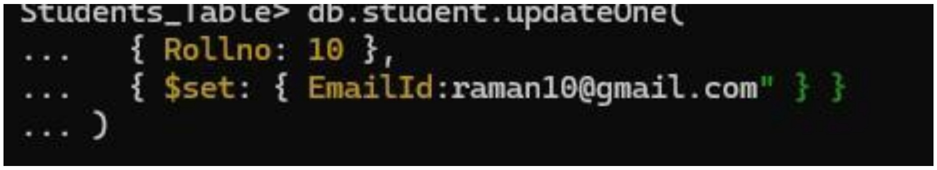


```
switched to db Students_Table
Students_Table> db.student.insertMany([
...   { Rollno: 10, Age: 20, ContactNo: "9876543210", EmailId: "rahul10@gmail.com", Name: "RAHUL" },
...   { Rollno: 11, Age: 21, ContactNo: "9123456780", EmailId: "raj11@gmail.com", Name: "RAJ" },
...   { Rollno: 12, Age: 22, ContactNo: "9988776655", EmailId: "rajvardhan12@gmail.com", Name: "RAJVARDHAN" }
... ])
...
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6927c9b61771b37f9963b112'),
    '1': ObjectId('6927c9b61771b37f9963b113'),
    '2': ObjectId('6927c9b61771b37f9963b114')
  }
}
```



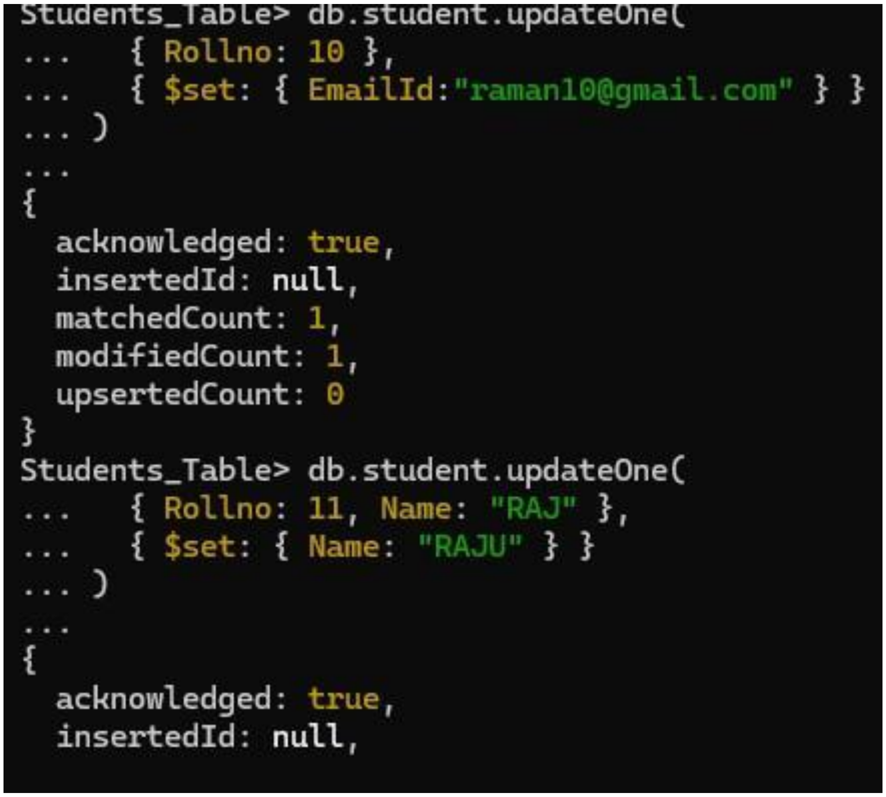
```
_id: ObjectId('6927c6f526e0825c3463b112')
Rollno : 10
Age : 20
ContactNo : "9876543210"
EmailId : "newemail10@gmail.com"
Name : "ABC"

_id: ObjectId('6927c6f526e0825c3463b113')
Rollno : 11
Age : 21
ContactNo : "9123456780"
EmailId : "abc11@gmail.com"
Name : "ABC"

_id: ObjectId('6927c6f526e0825c3463b114')
Rollno : 12
Age : 22
ContactNo : "9988776655"
EmailId : "abc12@gmail.com"
Name : "XYZ"
```

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



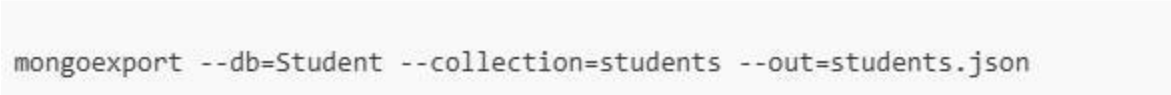
```
Students_Table> db.student.updateOne(
...    { Rollno: 10 },
...    { $set: { EmailId:raman10@gmail.com" } }
... )
```

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



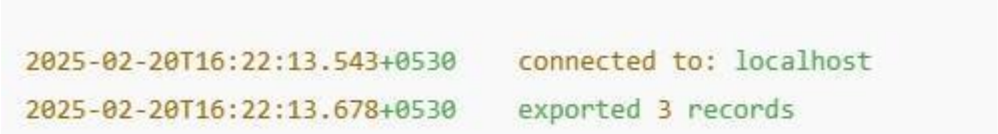
```
Students_Table> db.student.updateOne(
...    { Rollno: 10 },
...    { $set: { EmailId:"raman10@gmail.com" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Students_Table> db.student.updateOne(
...    { Rollno: 11, Name: "RAJ" },
...    { $set: { Name: "RAJU" } }
... )
...
{
  acknowledged: true,
  insertedId: null,
```

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



```
mongoexport --db=Student --collection=students --out=students.json
```

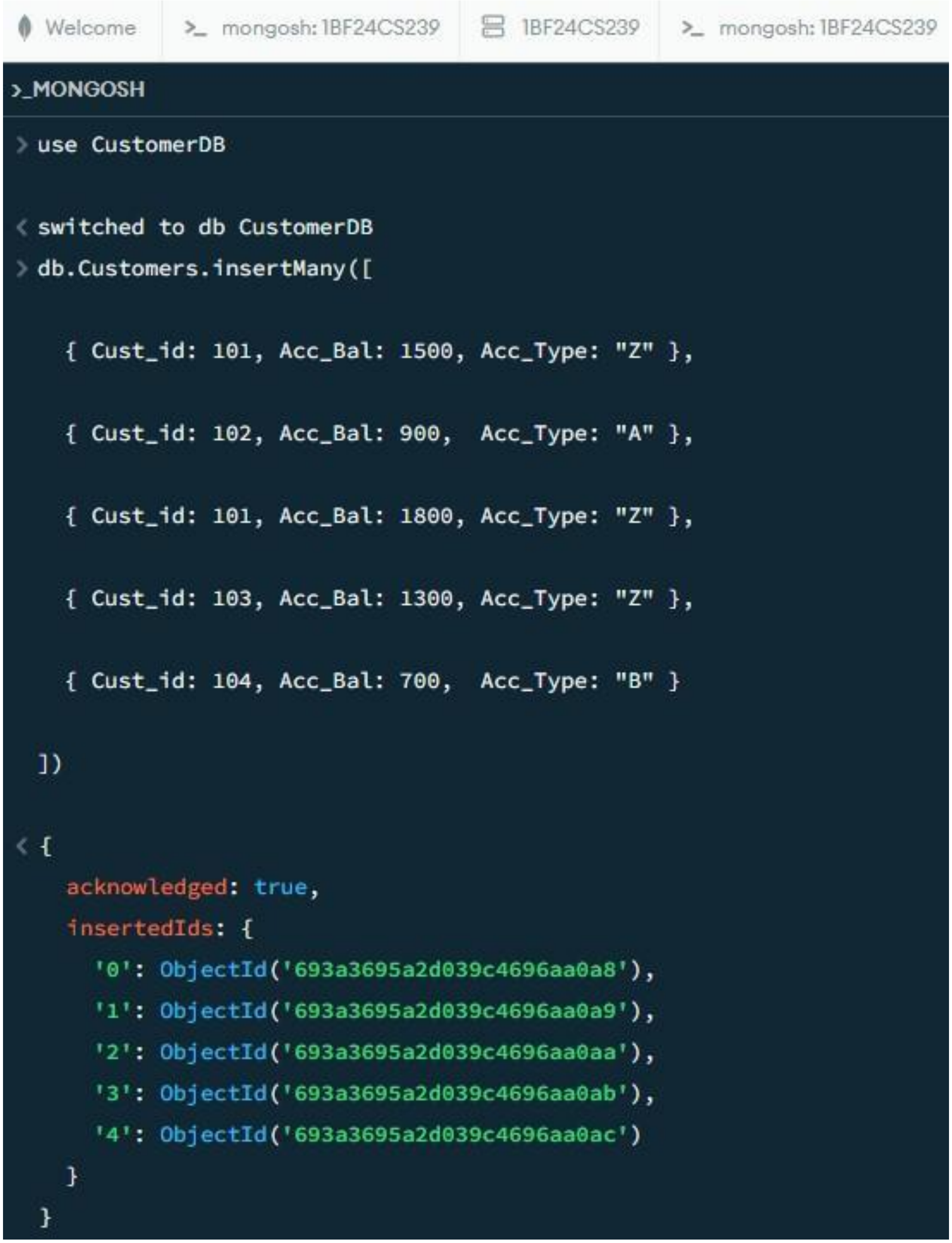
Export the created table into local file system.



```
2025-02-20T16:22:13.543+0530    connected to: localhost
2025-02-20T16:22:13.678+0530    exported 3 records
```

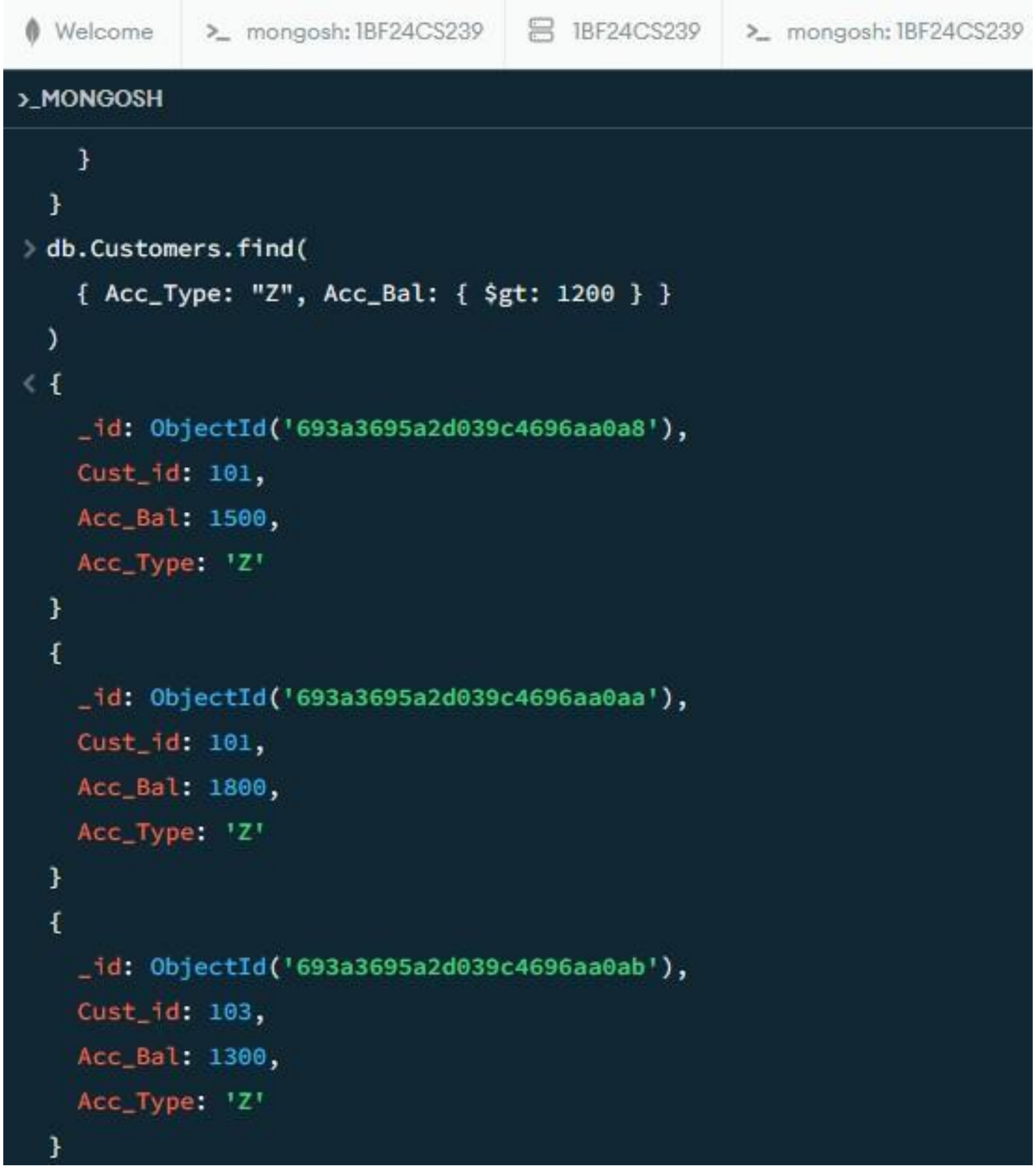
Experiment no9 : Nosql lab1(Customer)

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



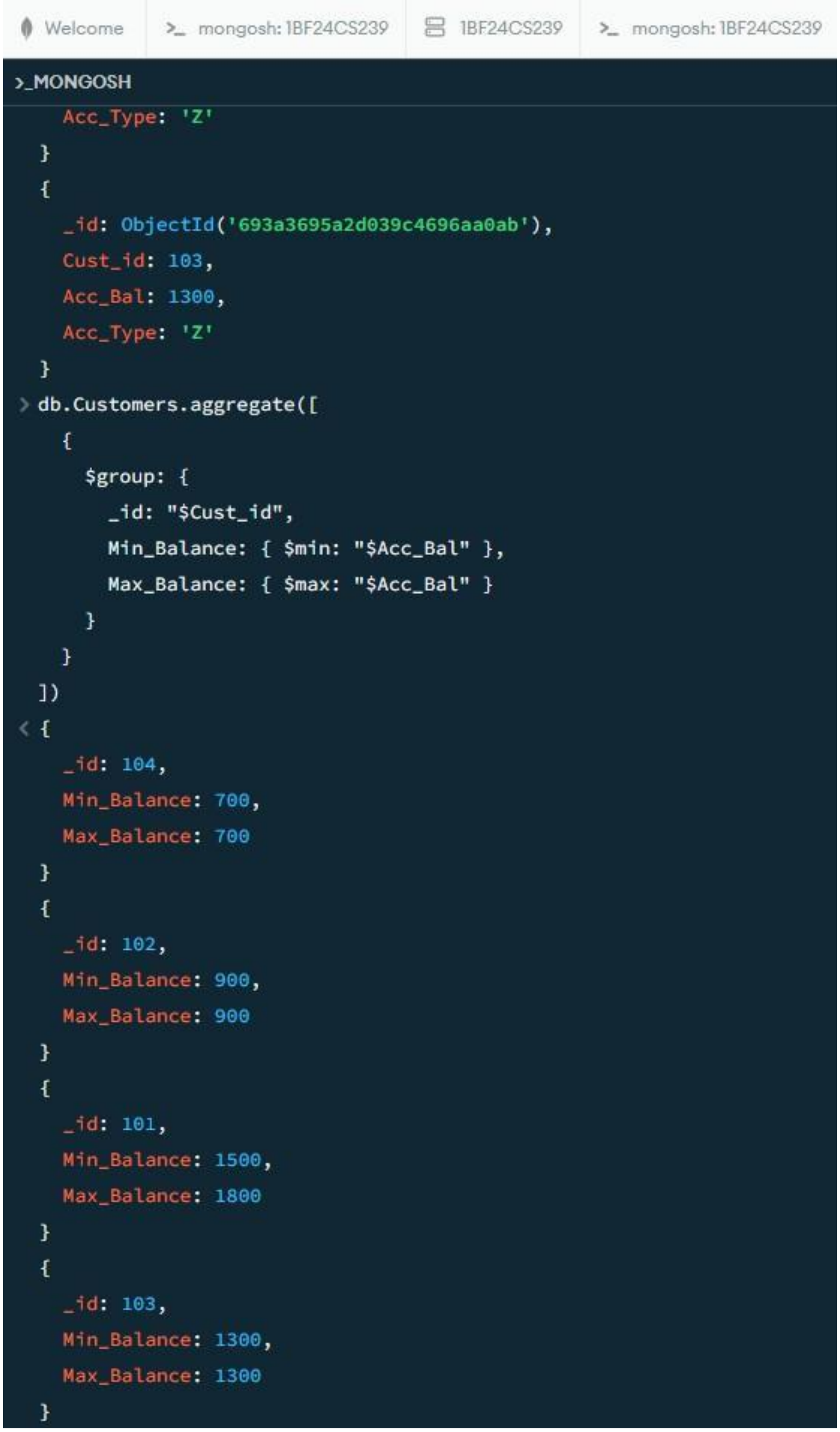
```
> use CustomerDB

< switched to db CustomerDB
> db.Customers.insertMany([

    { Cust_id: 101, Acc_Bal: 1500, Acc_Type: "Z" },

    { Cust_id: 102, Acc_Bal: 900,  Acc_Type: "A" },

    { Cust_id: 101, Acc_Bal: 1800, Acc_Type: "Z" },

    { Cust_id: 103, Acc_Bal: 1300, Acc_Type: "Z" },

    { Cust_id: 104, Acc_Bal: 700,  Acc_Type: "B" }

  ])

< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('693a3695a2d039c4696aa0a8'),
      '1': ObjectId('693a3695a2d039c4696aa0a9'),
      '2': ObjectId('693a3695a2d039c4696aa0aa'),
      '3': ObjectId('693a3695a2d039c4696aa0ab'),
      '4': ObjectId('693a3695a2d039c4696aa0ac')
    }
  }
```

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



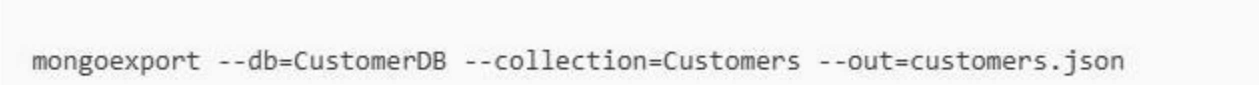
```
    }
  }
> db.Customers.find(
    { Acc_Type: "Z", Acc_Bal: { $gt: 1200 } }
  )
< {
    _id: ObjectId('693a3695a2d039c4696aa0a8'),
    Cust_id: 101,
    Acc_Bal: 1500,
    Acc_Type: 'Z'
  }
  {
    _id: ObjectId('693a3695a2d039c4696aa0aa'),
    Cust_id: 101,
    Acc_Bal: 1800,
    Acc_Type: 'Z'
  }
  {
    _id: ObjectId('693a3695a2d039c4696aa0ab'),
    Cust_id: 103,
    Acc_Bal: 1300,
    Acc_Type: 'Z'
  }
```

Determine Minimum and Maximum account balance for each customer_id.



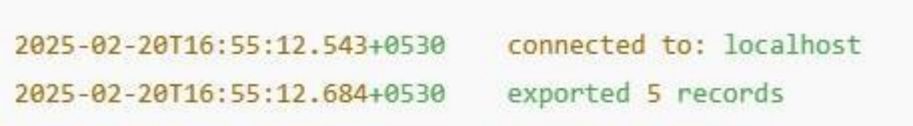
```
    Acc_Type: 'Z'
  }
  {
    _id: ObjectId('693a3695a2d039c4696aa0ab'),
    Cust_id: 103,
    Acc_Bal: 1300,
    Acc_Type: 'Z'
  }
> db.Customers.aggregate([
    {
      $group: {
        _id: "$Cust_id",
        Min_Balance: { $min: "$Acc_Bal" },
        Max_Balance: { $max: "$Acc_Bal" }
      }
    }
  ])
< {
    _id: 104,
    Min_Balance: 700,
    Max_Balance: 700
  }
  {
    _id: 102,
    Min_Balance: 900,
    Max_Balance: 900
  }
  {
    _id: 101,
    Min_Balance: 1500,
    Max_Balance: 1800
  }
  {
    _id: 103,
    Min_Balance: 1300,
    Max_Balance: 1300
  }
```

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



```
mongoexport --db=CustomerDB --collection=Customers --out=customers.json
```

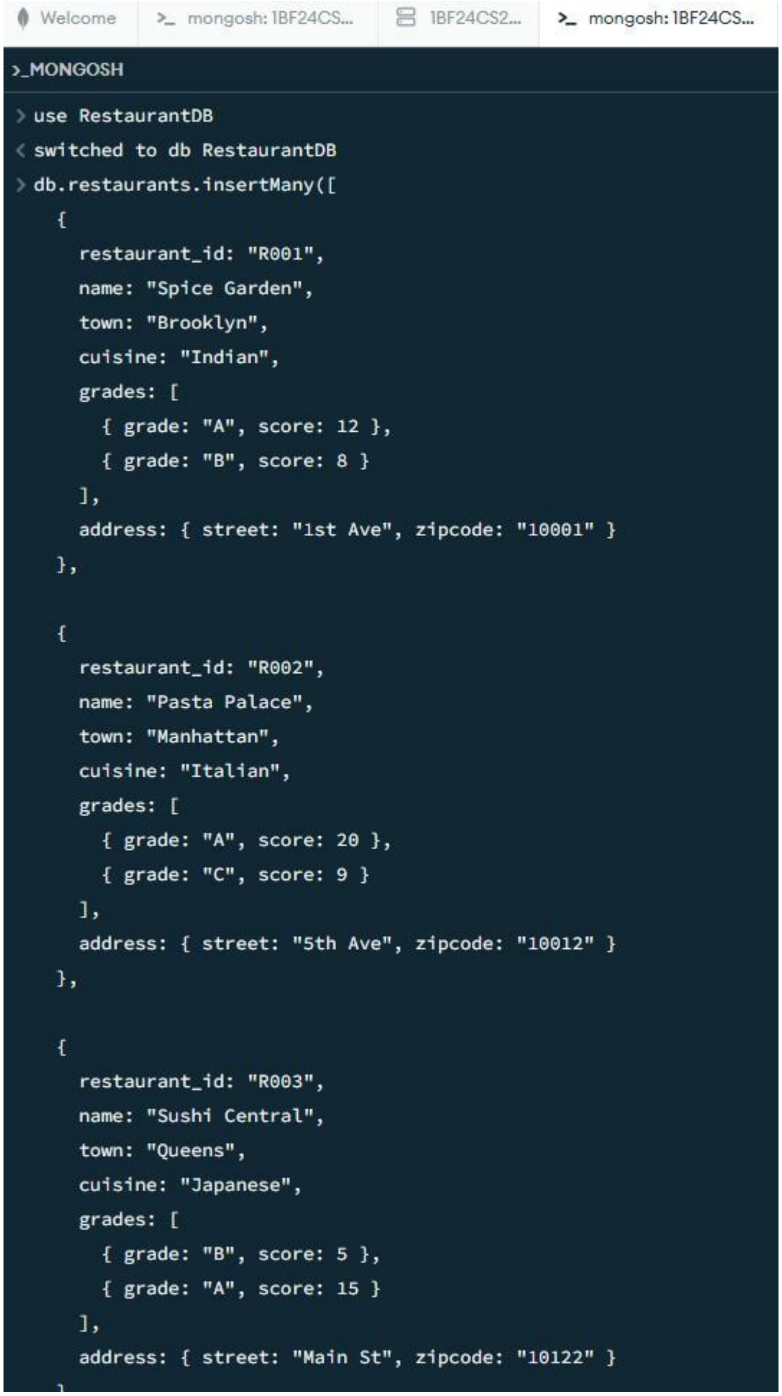
Export the created collection into local file system.



```
2025-02-20T16:55:12.543+0530    connected to: localhost
2025-02-20T16:55:12.684+0530    exported 5 records
```

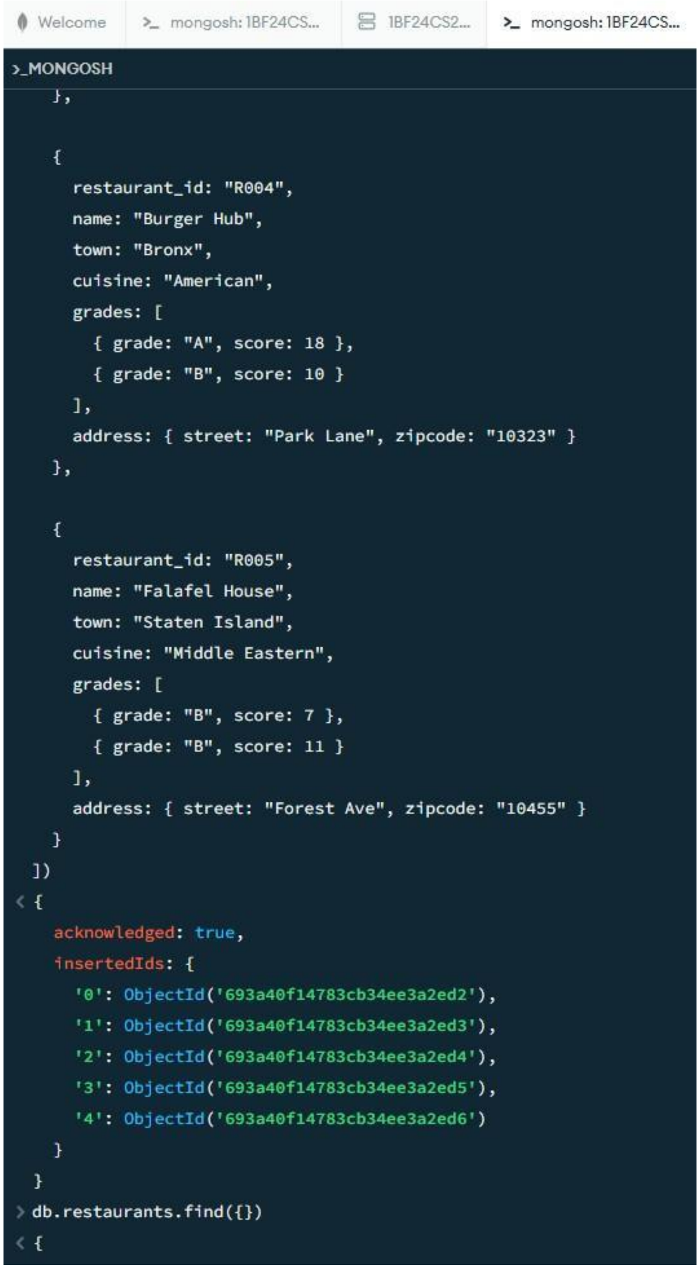
Experimentno 10: NoSQL Restaurant Database

Write NoSQL Queries on “Restaurant” collection.



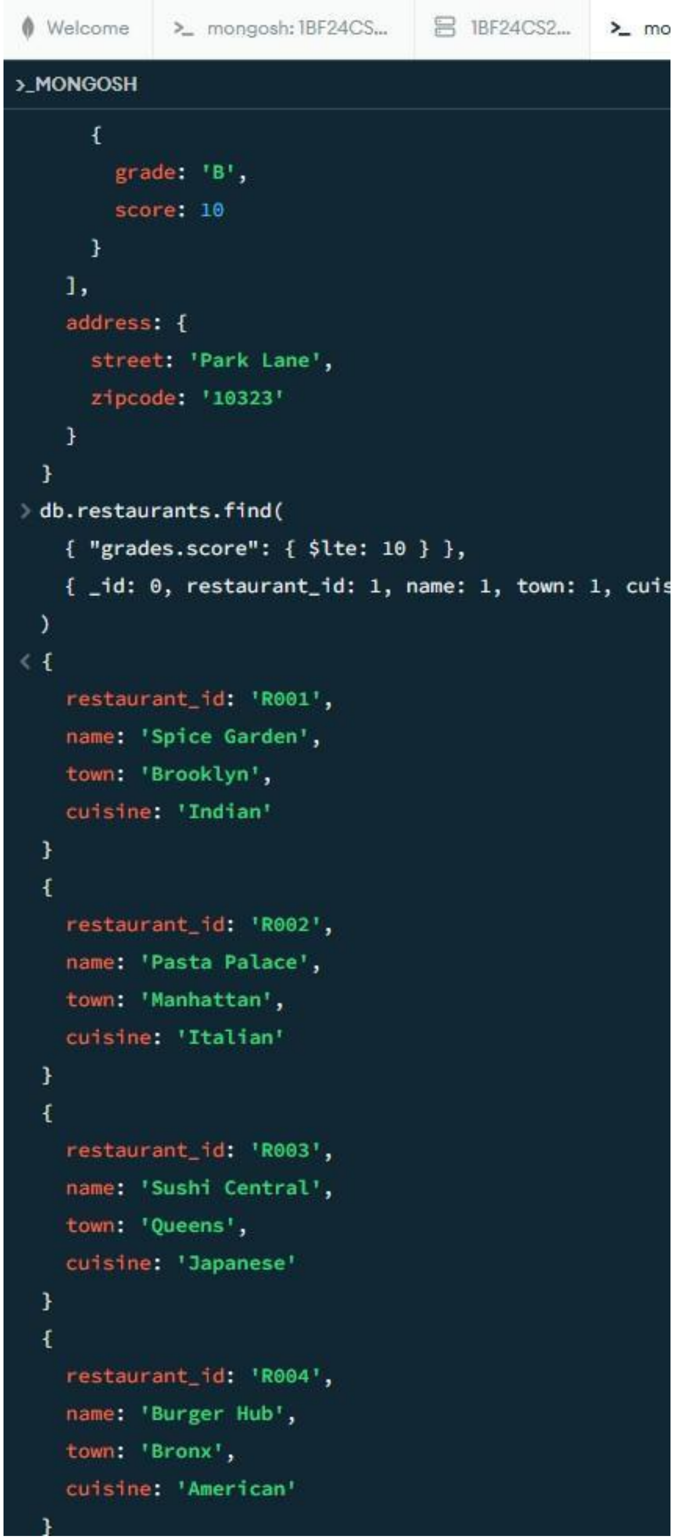
```
> use RestaurantDB
< switched to db RestaurantDB
> db.restaurants.insertMany([
    {
      restaurant_id: "R001",
      name: "Spice Garden",
      town: "Brooklyn",
      cuisine: "Indian",
      grades: [
        { grade: "A", score: 12 },
        { grade: "B", score: 8 }
      ],
      address: { street: "1st Ave", zipcode: "10001" }
    },

    {
      restaurant_id: "R002",
      name: "Pasta Palace",
      town: "Manhattan",
      cuisine: "Italian",
      grades: [
        { grade: "A", score: 20 },
        { grade: "C", score: 9 }
      ],
      address: { street: "5th Ave", zipcode: "10012" }
    },

    {
      restaurant_id: "R003",
      name: "Sushi Central",
      town: "Queens",
      cuisine: "Japanese",
      grades: [
        { grade: "B", score: 5 },
        { grade: "A", score: 15 }
      ],
      address: { street: "Main St", zipcode: "10122" }
```

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



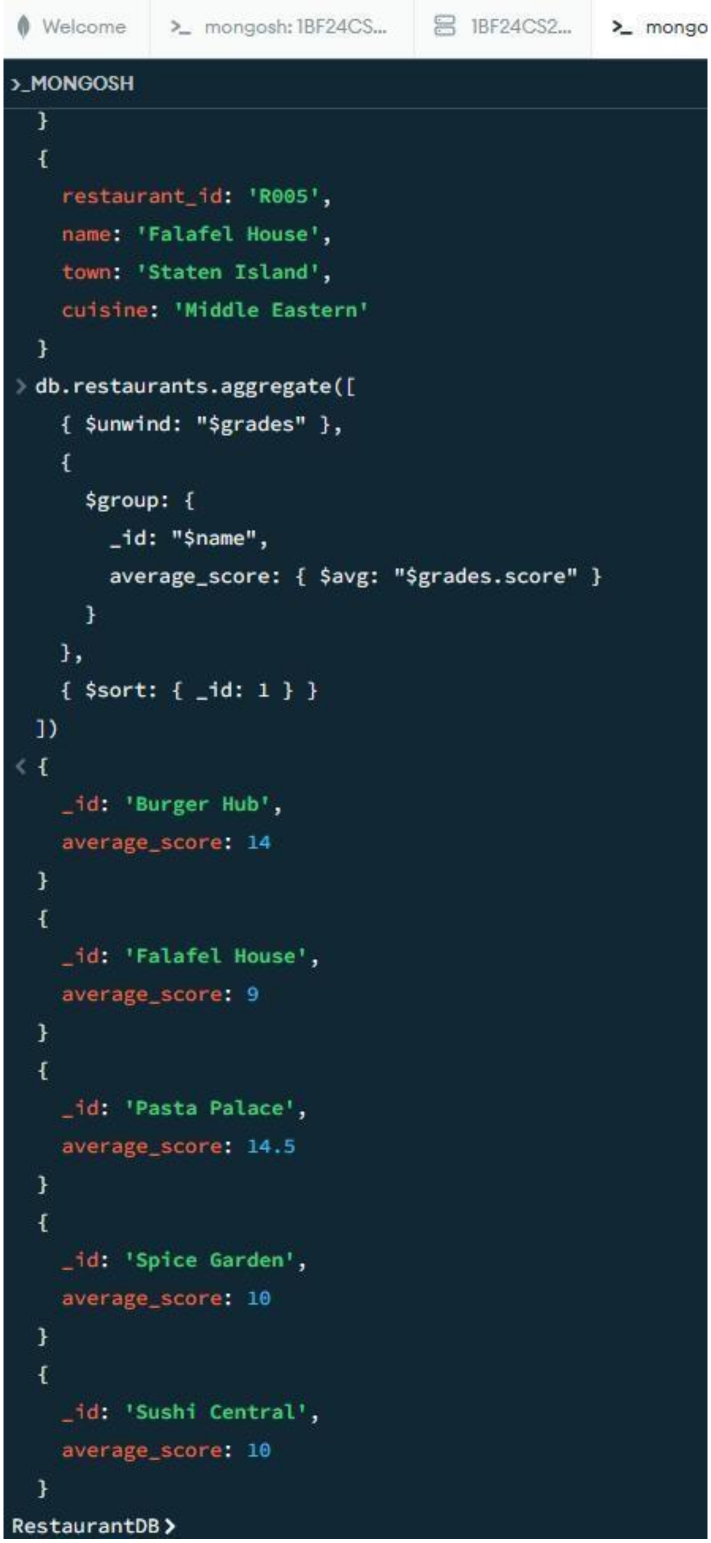
```
    },

    {
      restaurant_id: "R004",
      name: "Burger Hub",
      town: "Bronx",
      cuisine: "American",
      grades: [
        { grade: "A", score: 18 },
        { grade: "B", score: 10 }
      ],
      address: { street: "Park Lane", zipcode: "10323" }
    },

    {
      restaurant_id: "R005",
      name: "Falafel House",
      town: "Staten Island",
      cuisine: "Middle Eastern",
      grades: [
        { grade: "B", score: 7 },
        { grade: "B", score: 11 }
      ],
      address: { street: "Forest Ave", zipcode: "10455" }
    }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('693a40f14783cb34ee3a2ed2'),
      '1': ObjectId('693a40f14783cb34ee3a2ed3'),
      '2': ObjectId('693a40f14783cb34ee3a2ed4'),
      '3': ObjectId('693a40f14783cb34ee3a2ed5'),
      '4': ObjectId('693a40f14783cb34ee3a2ed6')
    }
  }
> db.restaurants.find({})
< {
```

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.



```
      {
        grade: 'B',
        score: 10
      }
    ],
    address: {
      street: 'Park Lane',
      zipcode: '10323'
    }
  }
> db.restaurants.find(
    { "grades.score": { $lte: 10 } },
    { _id: 0, restaurant_id: 1, name: 1, town: 1, cuis
  )
< {
    restaurant_id: 'R001',
    name: 'Spice Garden',
    town: 'Brooklyn',
    cuisine: 'Indian'
  }
  {
    restaurant_id: 'R002',
    name: 'Pasta Palace',
    town: 'Manhattan',
    cuisine: 'Italian'
  }
  {
    restaurant_id: 'R003',
    name: 'Sushi Central',
    town: 'Queens',
    cuisine: 'Japanese'
  }
  {
    restaurant_id: 'R004',
    name: 'Burger Hub',
    town: 'Bronx',
    cuisine: 'American'
  }
```

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



```
  }
  {
    restaurant_id: 'R005',
    name: 'Falafel House',
    town: 'Staten Island',
    cuisine: 'Middle Eastern'
  }
> db.restaurants.aggregate([
    { $unwind: "$grades" },
    {
      $group: {
        _id: "$name",
        average_score: { $avg: "$grades.score" }
      }
    },
    { $sort: { _id: 1 } }
  ])
< {
    _id: 'Burger Hub',
    average_score: 14
  }
  {
    _id: 'Falafel House',
    average_score: 9
  }
  {
    _id: 'Pasta Palace',
    average_score: 14.5
  }
  {
    _id: 'Spice Garden',
    average_score: 10
  }
  {
    _id: 'Sushi Central',
    average_score: 10
  }
RestaurantDB >
```