VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT

on

Database Management Systems

(23CS3PCDBM)

Submitted by

Prakriti Jain (1BM23CS239)

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

B.M.S. COLLEGE OF ENGINEERING
(Autonomous Institution under VTU)
BENGALURU-560019

Sep-2024 to Jan-2025

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

CERTIFICATE

This is to certify that the Lab work entitled "Database Management Systems (23CS3PCDBM)" carried out by **Prakriti Jain (1BM23CS239),** who is a bonafide student of **B.M.S. College of Engineering.** It is in partial fulfilment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum. The lab report has been approved as it satisfies the academic requirements for a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

Dr. Amrutha B Associate Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

Insurance Database

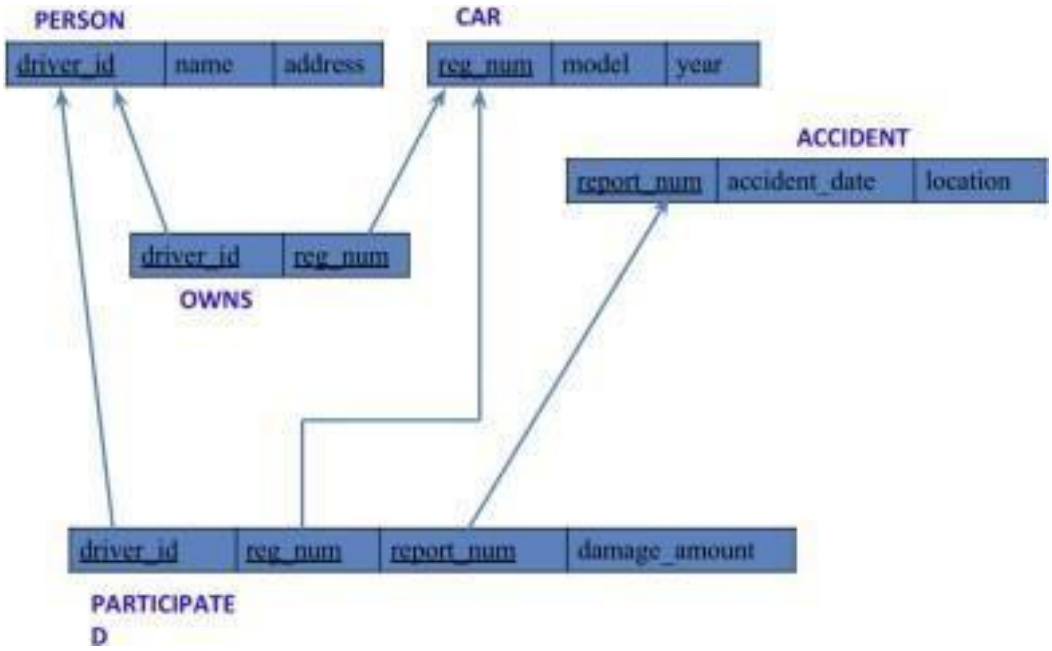
Questio

n (Week

1)

- PERSON (driver_id: String, name: String, address: String)
- CAR (reg_num: String, model: String, year: int)
- ACCIDENT (report_num: int, accident_date: date, location: String)
- OWNS (driver_id: String, reg_num: String)
- PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)
- Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation
- Display Accident date and location
- Update the damage amount to 25000 for the car with a specific reg_num (example 'K A053408') for which the accident report number was 12.
- Add a new accident to the database.
- To Do
- Display Accident date and location
- Display driver id who did accident with damage amount greater than or equal to Rs.25000

Schema Diagram



Create database

create database insurance;

use insurance;

Create table

create table person (driver_id varchar(10),

name varchar(20), address varchar(30), primary key(driver_id));

create table car(reg_num varchar(10),model varchar(10),year int, primary key(reg_num));

create table accident(report_num int, accident_date date, location varchar(20),primary key(report_num));

create table 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 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(10)	NO	PRI	NULL	
	name	varchar(20)	YES		NULL	
	address	varchar(30)	YES		NULL	

desc car;

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

desc accident;

	Field	Type	Null	Key	Default	Extra
▶	report_num	int	NO	PRI	NULL	
	accident_date	date	YES		NULL	
	location	varchar(20)	YES		NULL	

desc owns;

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

desc participated;

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

Inserting Values to the table

insert into person values("A01", "Richard", "Srinivasanagar");

insert into person values("A02", "Pradeep", "Rajajinagar");

insert into person values("A03", "Smith", "Ashoknagar");

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

insert into person values("A05", "John",

"Hanumanthanagar"); select * from person;

	driver_id	name	address
▶	A01	Richard	Srinivasanagar
	A02	Pradeep	Rajajinagar
	A03	Smith	Ashoknagar
	A04	Venu	N R Colony
	A05	John	Hanumanthanagar
*	NULL	NULL	NULL

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;

driver_id	name	address
A01	Richard	Srinivasanagar
A02	Pradeep	Rajajinagar
A03	Smith	Ashoknagar
A04	Venu	N R Colony
A05	John	Hanumanthanagar
NULL	NULL	NULL

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
A03	KA031181
A05	KA041702
A01	KA052250
A02	KA053408
A04	KA095477
NULL	NULL

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
16	2008-03-08	Domlur
NULL	NULL	NULL

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
NULL	NULL	NULL	NULL

Queries

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 where report_num=12;

driver_id	reg_num	report_num	damage_amount
A02	KA053408	12	25000
NULL	NULL	NULL	NULL

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

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

	count(distinct driver_id)
▶	1

Add a new accident to the database.

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

	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
	16	2008-03-08	Domlur

accident;

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
	2004-03-05	Kanakpura Road
	2008-03-08	Domlur

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

select driver_id from participated where damage_amount>=25000;

	driver_id
▶	A02
	A03

More Queries on Insurance Database

Questio

n (Week

2)

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

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

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

- OWNS (driver_id: String, reg_num: String)

- PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)

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

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

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

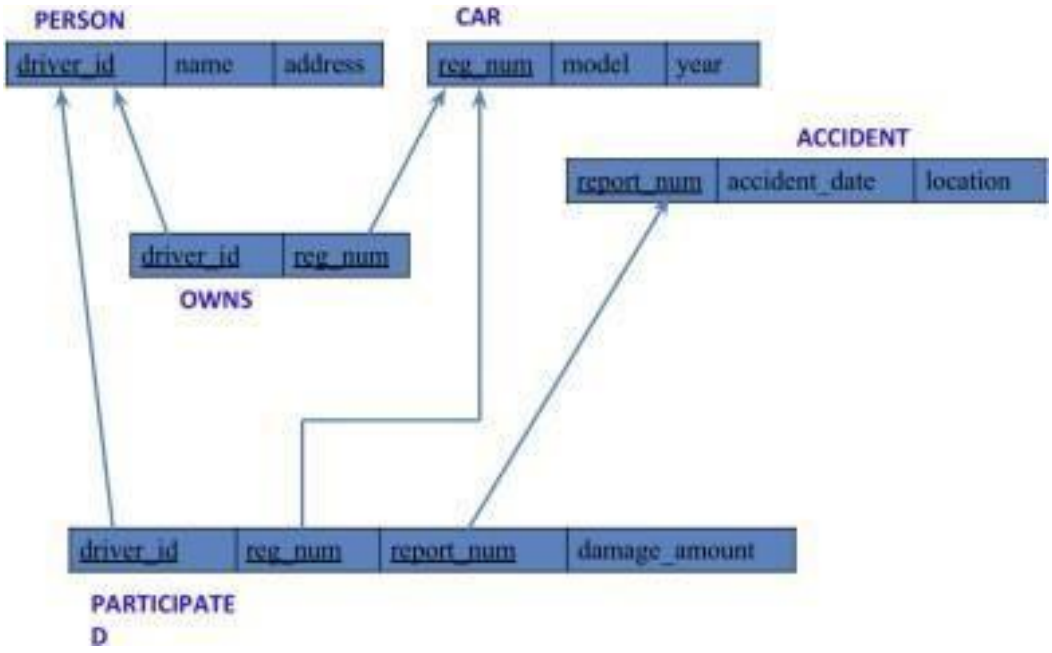
- List the entire participated relation in the descending order of damage amount.

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

- Delete the tuple whose damage amount is below the average damage amount

- Find maximum damage amount.

Schema Diagram



Queries

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

```
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.

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

count(report_num)
1

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

```
select count(distinct driver_id)
  from participated a, accident b
  where a.report_num=b.report_num and b.accident_date like "_08%";
```

count(distinct driver_id)
1

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

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

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

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

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

name
Pradeep
Smith

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

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

select * from participated;

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

Find maximum damage amount.

select max(damage_amount) from participated;

max(damage_amount)
25000

Bank Database

Questio

n (Week

3)

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

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

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

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

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

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

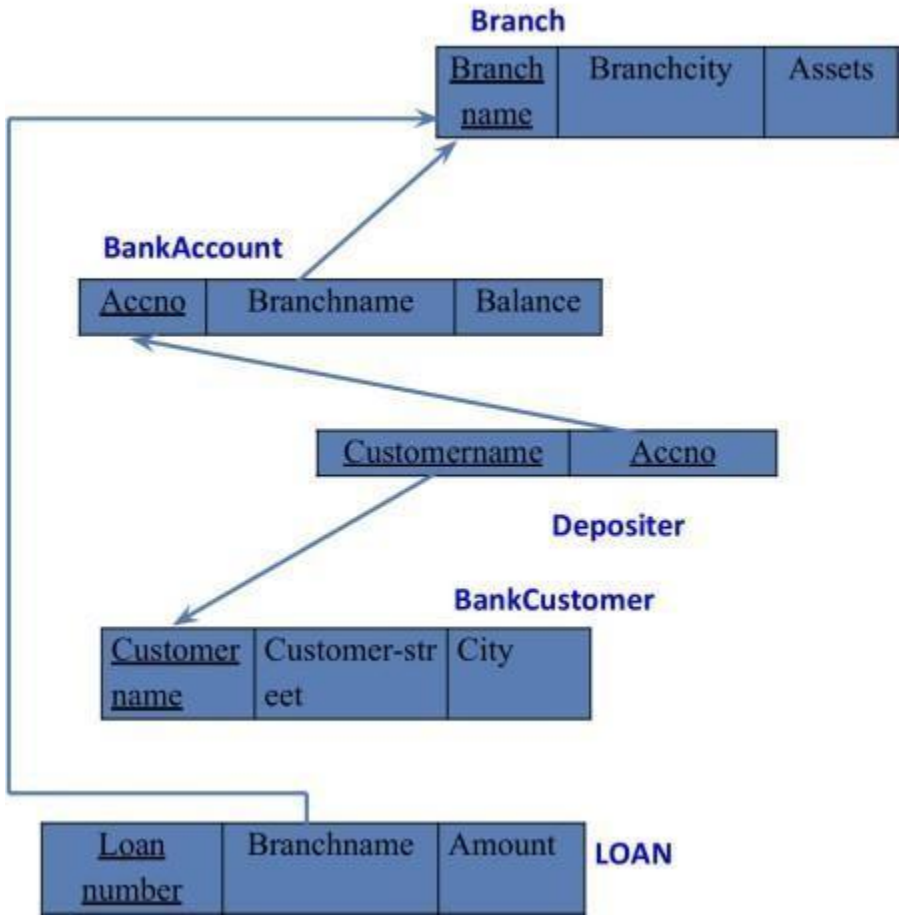
- Enter at least five tuples for each relation.

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

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

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

Schema Diagram



Create Database:

```
create database
bankdb;
Use bankdb;
```

Create Table:

```
create table Branch(
BranchName varchar(30) primary key,
BranchCity varchar(20),
Assets int);


create table BankAccount(
AccNo int primary key,
BranchName varchar(30),
Balance int,
foreign key(BranchName) references Branch(BranchName));
create table BankCustomer(
```

```
CustomerName varchar(20) primary key,
CustomerStreet varchar(30),
CustomerCity varchar(20));

create table Depositer(
CustomerName varchar(20),
AccNo int,
foreign key(CustomerName) references BankCustomer(CustomerName),
foreign key(AccNo) references BankAccount(AccNo));

create table Loan(
LoanNumber int primary key,
BranchName varchar(30),
Amount int,
foreign key(BranchName) references Branch(BranchName));
```

Structure of the Table:

desc branch;

Field	Type	Null	Key	Default	Extra
BranchName	varchar(30)	NO	PRI	NULL	
BranchCity	varchar(20)	YES		NULL	
Assets	int	YES		NULL	

desc BankAccount;

Field	Type	Null	Key	Default	Extra
AccNo	int	NO	PRI	NULL	
BranchName	varchar(30)	YES	MUL	NULL	
Balance	int	YES		NULL	

desc Depositer;

Field	Type	Null	Key	Default	Extra
CustomerName	varchar(20)	YES	MUL	NULL	
AccNo	int	YES	MUL	NULL	

desc BankCustomer;

Field	Type	Null	Key	Default	Extra
CustomerName	varchar(20)	NO	PRI	NULL	
CustomerStreet	varchar(30)	YES		NULL	
CustomerCity	varchar(20)	YES		NULL	

desc Loan;

	Field	Type	Null	Key	Default	Extra
▶	LoanNumber	int	NO	PRI	NULL	
	BranchName	varchar(30)	YES	MUL	NULL	
	Amount	int	YES		NULL	

Inserting Values to the tables:

```
insert into Branch values
("SBI_Chamrajpet", "Bangalore",
50000),
("SBI_ResidencyRoad", "Bangalore", 10000),
("SBI_ShivajiRoad", "Bombay", 20000),
("SBI_ParliamentRoad", "Delhi", 10000),
("SBI_Jantarmantar", "Delhi", 20000);
select * from Branch;
```

	BranchName	BranchCity	Assets
▶	SBI_Chamrajpet	Bangalore	50000
	SBI_Jantarmantar	Delhi	20000
	SBI_ParliamentRoad	Delhi	10000
	SBI_ResidencyRoad	Bangalore	10000
	SBI_ShivajiRoad	Bombay	20000
*	NULL	NULL	NULL

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

	AccNo	BranchName	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
	8	SBI_ResidencyRoad	4000
	9	SBI_ParliamentRoad	3000
	10	SBI_ResidencyRoad	5000
	11	SBI_Jantarmantar	2000
*	NULL	NULL	NULL

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

	CustomerName	CustomerStreet	CustomerCity
▶	Avinash	Bull_Temple_Road	Bangalore
	Dinesh	Bannergatta_Road	Bangalore
	Mohan	NationalCollege_Road	Bangalore
	Nikil	Akbar_Road	Delhi
	Ravi	PrithviRaj	Delhi
*	NULL	NULL	NULL

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

	CustomerName	AccNo
▶	Avinash	1
	Dinesh	2
	Nikil	4
	Ravi	5
	Avinash	8
	Nikil	9
	Dinesh	10
	Nikil	11

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

	LoanNumber	BranchName	Amount
▶	1	SBI_Chamrajpet	1000
	2	SBI_ResidencyRoad	2000
	3	SBI_ShivajiRoad	3000
	4	SBI_ParliamentRoad	4000
	5	SBI_Jantarmantar	5000
*	NULL	NULL	NULL

Queries:

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

```
select BranchName, Assets / 100000 as "Assets in Lakhs" from Branch;
```

BranchName	Assets in Lakhs
SBI_Chamrajpet	0.5000
SBI_Jantarmantar	0.2000
SBI_ParliamentRoad	0.1000
SBI_ResidencyRoad	0.1000
SBI_ShivajiRoad	0.2000

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

```
select CustomerName from Depositer where AccNo in
(select AccNo from BankAccount where BranchName = "SBI_ResidencyRoad")
group by CustomerName having count(AccNo) > 1;
```

CustomerName
Dinesh

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

```
create view NetLoan as
select BranchName, sum(Amount) as "Net Loan Amount" from Loan group by BranchName;
select * from NetLoan;
```

BranchName	Net Loan Amount
SBI_Chamrajpet	1000
SBI_Jantarmantar	5000
SBI_ParliamentRoad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

More Queries on Bank Database

Questio

n (Week

4)

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

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

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

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

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

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

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

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

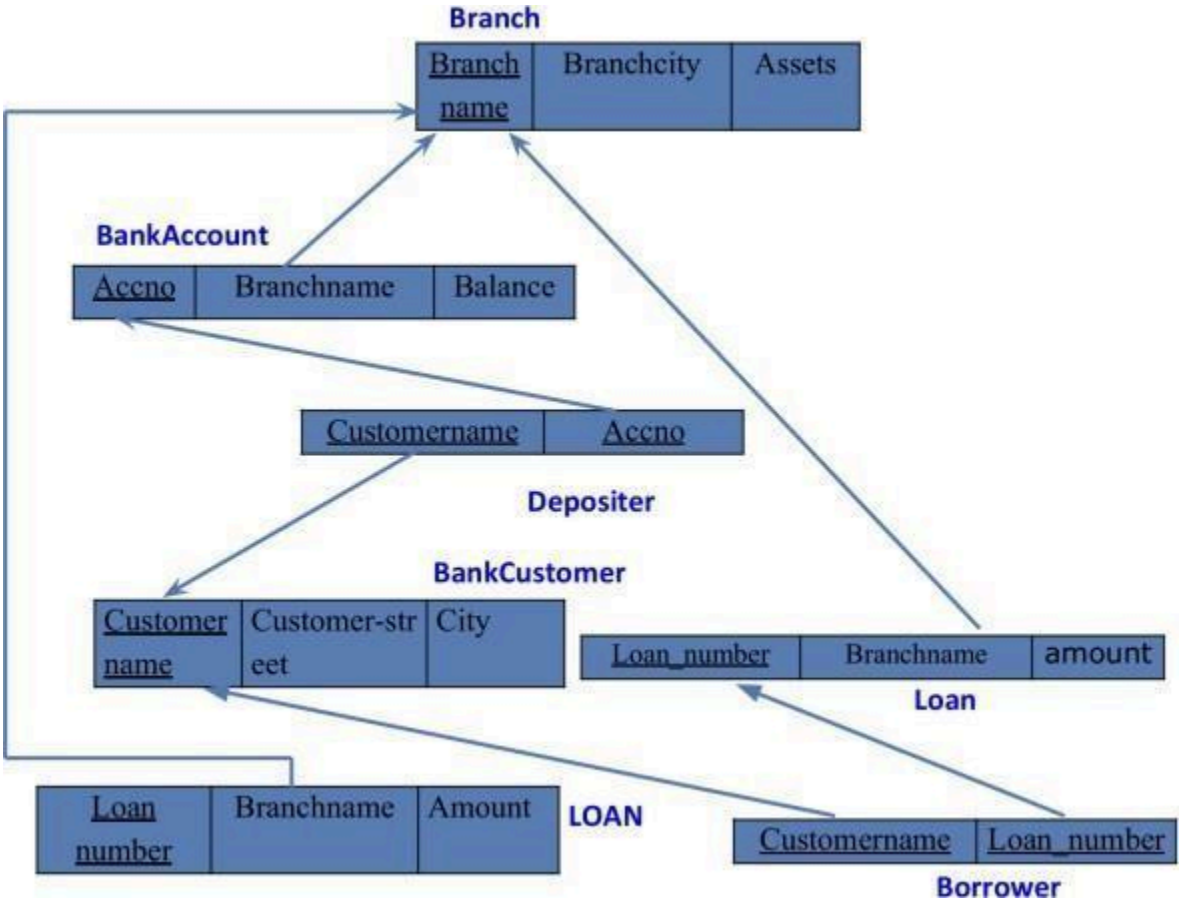
- Find all customers who have both an account and a loan at the Bangalore branch

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

- Update the Balance of all accounts by 5%

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

Schema Diagram:



Create Table:

```
create table Borrower(
CustomerName varchar(30),
LoanNumber int,
foreign key(CustomerName) references BankCustomer(CustomerName),
foreign key(LoanNumber) references Loan(LoanNumber));
```

Structure of the Table:

desc Borrower;

	Field	Type	Null	Key	Default	Extra
▶	CustomerName	varchar(30)	YES	MUL	NULL	
	LoanNumber	int	YES	MUL	NULL	

Inserting Values to the tables:

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

	CustomerName	LoanNumber
▶	Avinash	1
	Dinesh	2
	Mohan	3
	Nikil	4
	Ravi	5

Queries:

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

```
select distinct CustomerName, CustomerCity from Branch b, BankCustomer bc
where b.BranchCity=bc.CustomerCity and bc.CustomerCity="Delhi";
```

	CustomerName	CustomerCity
▶	Nikil	Delhi
	Ravi	Delhi

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

```
select distinct bc.CustomerName, l.BranchName, l.LoanNumber
from BankCustomer bc, Loan l, Borrower b
where bc.CustomerName= b.CustomerName and
l.LoanNumber=b.LoanNumber
and bc.CustomerName NOT IN (
   select d.CustomerName
   from Depositer d);
```

	CustomerName	BranchName	LoanNumber
▶	Mohan	SBI_ShivajiRoad	3

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

```
select distinct bc.CustomerCity, b.CustomerName, l.LoanNumber, br.BranchName
fromBranch br, Borrower b, Loan l, BankCustomer bc
where
br.BranchCity = bc.CustomerCity AND
br.BranchCity = 'Bangalore' AND
l.LoanNumber = b.LoanNumber AND
bc.CustomerName = b.CustomerName
and br.BranchName=l.BranchName;
```

CustomerCity	CustomerName	LoanNumber	BranchName
Bangalore	Avinash	1	SBI_Chamrajpet
Bangalore	Dinesh	2	SBI_ResidencyRoad

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

```
select distinct br.BranchName, br.BranchCity, br.Assets
from Branch br where
br.Assets> all(select max(br.Assets) where br.BranchCity="Bangalore");
select max(Assets), BranchName from Branch
group by BranchName;
```

BranchName	BranchCity	Assets

Update the Balance of all accounts by 5%

```
update BankAccount set Balance= Balance+ 0.05*Balance;
```

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

```
delete from BankAccount ba
where ba.BranchName in(
select br.BranchName from Branch br
where br.BranchCity="Bombay");
select * from BankAccount;
```

	AccNo	BranchName	Balance
▶	1	SBI_Chamrajpet	2100
	2	SBI_ResidencyRoad	5250
	4	SBI_ParliamentRoad	9450
	5	SBI_Jantarmantar	8400
	8	SBI_ResidencyRoad	4200
	9	SBI_ParliamentRoad	3150
	10	SBI_ResidencyRoad	5250
	11	SBI_Jantarmantar	2100
*	NULL	NULL	NULL

Employee Database

Questio

n (Week

5)

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

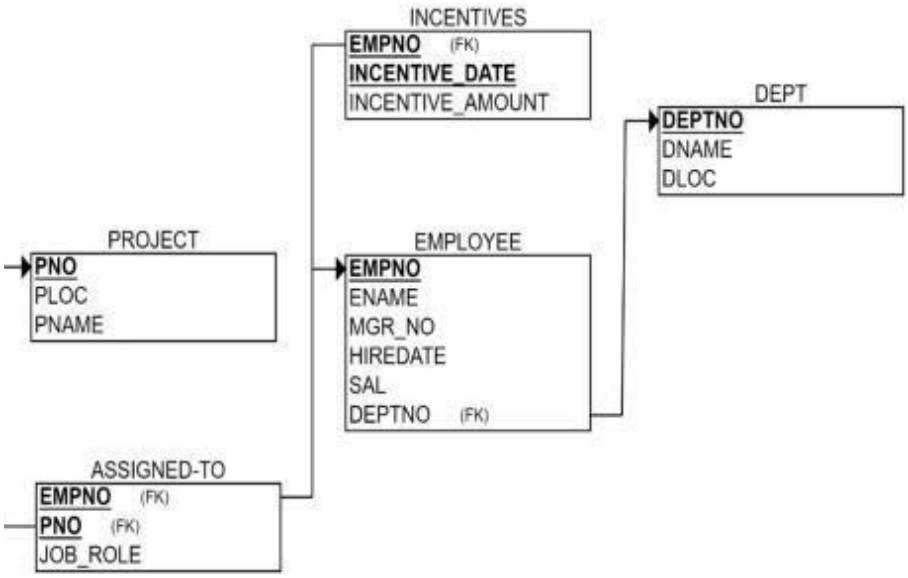
- Enter greater than five tuples for each table.

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

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

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

Schema Diagram:



Create Database:

```
create database empl;
use empl;
```

Create Table:

```
create table Department(
Dept_No int,
Dept_Name
varchar(30), Dept_Loc
varchar(30), primary
key(Dept_No));

create table
Employee( Emp_No
int,
Ename varchar(30),
MGR_No int,
Hire_Date date,
Salary float,
Dept_No int,
primary key(Emp_No),
foreign key (Dept_No) references
Department(Dept_No) on delete cascade );

create table Incentives(
Emp_No int,
Incentive_Date date,
Incentive_Amount float,
primary key(Emp_No, Incentive_Date),
foreign key (Emp_No) references Employee(Emp_No));

create table Project(
Pro_Loc varchar(30),
Pro_No int,
Pro_Name varchar(30),
primary key(Pro_No));

create table Assigned_To(
Emp_No int,
Pro_No int,
Job_Role varchar(30),
foreign key (Emp_No) references Employee(Emp_No),
foreign key (Pro_No) references Project(Pro_No));
```

Structure of the Table:

desc department;

Field	Type	Null	Key	Default	Extra
Dept_No	int	NO	PRI	NULL	
Dept_Name	varchar(30)	YES		NULL	
Dept_Loc	varchar(30)	YES		NULL	

desc Employee;

Field	Type	Null	Key	Default	Extra
Emp_No	int	NO	PRI	NULL	
Ename	varchar(30)	YES		NULL	
MGR_No	int	YES		NULL	
Hire_Date	date	YES		NULL	
Salary	float	YES		NULL	
Dept_No	int	YES	MUL	NULL	

desc Incentives;

Field	Type	Null	Key	Default	Extra
Emp_No	int	NO	PRI	NULL	
Incentive_Date	date	NO	PRI	NULL	
Incentive_Amount	float	YES		NULL	

desc Project;

Field	Type	Null	Key	Default	Extra
Pro_Loc	varchar(30)	YES		NULL	
Pro_No	int	NO	PRI	NULL	
Pro_Name	varchar(30)	YES		NULL	

desc Assigned_To;

Field	Type	Null	Key	Default	Extra
Emp_No	int	YES	MUL	NULL	
Pro_No	int	YES	MUL	NULL	
Job_Role	varchar(30)	YES		NULL	

Inserting Values to the tables:

```
insert into Department values(1, 'IT', 'Bengaluru');
insert into Department values(2, 'Finance',
'Bengaluru');
insert into Department values(3, 'Fund_Raising', 'Mysuru');
insert into Department values(4, 'Testing_and_Debugging',
'Bengaluru'); insert into Department values(5, 'App_Developer',
'Mysuru');
select * from Department;
```

Dept_No	Dept_Name	Dept_Loc
1	IT	Bengaluru
2	Finance	Bengaluru
3	Fund_Raising	Mysuru
4	Testing_and_Debugging	Bengaluru
5	App_Developer	Mysuru
NULL	NULL	NULL

```
insert into Employee values(1, 'Avinash', 34, '2015-05-17', 250000, 1);
insert into Employee values(2, 'Balaji', 20, '2018-06-20', 200000, 2);
insert into Employee values(3, 'Chandan', 45, '2017-05-09', 180000, 3);
insert into Employee values(4, 'Dinesh', 2, '2023-04-23', 45000, 2);
insert into Employee values(5, 'Eshwar', 1, '2021-12-17', 55000, 1);
insert into Employee values(6, 'Fazal', 3, '2020-01-01', 75000, 3);
insert into Employee values(7, 'Gajendra', 1, '2021-10-17', 56000, 1);
insert into Employee values(8, 'Habeebullah', 3, '2024-05-17', 30000, 3);
insert into Employee values(9, 'Inaytullah', 1, '2022-09-09', 50000, 1);
select * from Employee;
```

Emp_No	Ename	MGR_No	Hire_Date	Salary	Dept_No
1	Avinash	34	2015-05-17	250000	1
2	Balaji	20	2018-06-20	200000	2
3	Chandan	45	2017-05-09	180000	3
4	Dinesh	2	2023-04-23	45000	2
5	Eshwar	1	2021-12-17	55000	1
6	Fazal	3	2020-01-01	75000	3
7	Gajendra	1	2021-10-17	56000	1
8	Habeebullah	3	2024-05-17	30000	3
9	Inaytullah	1	2022-09-09	50000	1
NULL	NULL	NULL	NULL	NULL	NULL

```
insert into Incentives values(1, '2019-01-14', 10000);
insert into Incentives values(2, '2019-01-16', 7500);
insert into Incentives values(3, '2019-01-05', 5000);
```

insert into Incentives values(4, '2024-05-14', null);
insert into Incentives values(5, '2023-12-13', 1500);
insert into Incentives values(6, '2021-12-28', 2000);
insert into Incentives values(7, '2023-10-13', 2500); insert into Incentives values(8, '2024-10-13', null); insert into Incentives values(9, '2024-09-07', 1000); select * from Incentives;

Emp_No	Incentive_Date	Incentive_Amount
1	2019-01-14	10000
2	2019-01-16	7500
3	2019-01-05	5000
4	2024-05-14	NULL
5	2023-12-13	1500
6	2021-12-28	2000
7	2023-10-13	2500
8	2024-10-13	1000
9	2024-09-07	1000
NULL	NULL	NULL

insert into Project values('Bengaluru',1,'ABC');
insert into Project values('Bengaluru',2,'XYZ');
insert into Project values('Mysuru',3,'PQR');
insert into Project values('Mysuru',4,'DEF');
insert into Project values('Bengaluru',5,'GHI');
select * from Project;

Pro_Loc	Pro_No	Pro_Name
Bengaluru	1	ABC
Bengaluru	2	XYZ
Mysuru	3	PQR
Mysuru	4	DEF
Bengaluru	5	GHI
NULL	NULL	NULL

insert into Assigned_To values(3,4,'Supervisor');
insert into Assigned_To values(6,1,'Manager');
insert into Assigned_To values(2,2,'Tester');
insert into Assigned_To values(7,5,'App_Developer');
insert into Assigned_To values(1,3,'Developer');
select * from Assigned_To;

	Emp_No	Pro_No	Job_Role
▶	3	4	Supervisor
	6	1	Manager
	2	2	Tester
	7	5	App_Developer
	1	3	Developer

Queries:

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

```
select Emp_No from Assigned_To a, Project p where
a.Pro_No=p.Pro_no and
p.Pro_Loc='Mysuru';
```

	Emp_No
▶	1
	3

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

```
select * from Incentives;
select i.Emp_No, e.Ename from Incentives i, Employee e where
i.Incentive_Amount is null and
e.Emp_No=i.Emp_No;
```

	Emp_No	Ename
▶	4	Dinesh
	8	Habeebullah

\

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

```
select e.Ename, e.Emp_No, d.Dept_Name, a.Job_Role,
d.Dept_Loc, p.Pro_Loc from Employee e, Assigned_To a, Department d, Project p where
e.Emp_No=a.Emp_No and
e.Dept_No=d.Dept_No and
p.Pro_No=a.Pro_no and
p.Pro_Loc=d.Dept_Loc;
```

	Ename	Emp_No	Dept_Name	Job_Role	Dept_Loc	Pro_Loc
▶	Chandan	3	Fund_Raising	Supervisor	Mysuru	Mysuru
	Balaji	2	Finance	Tester	Bengaluru	Bengaluru
	Gajendra	7	IT	App_Developer	Bengaluru	Bengaluru

More Queries on Employee Database

Questio

n (Week

6)

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

- Enter greater than five tuples for each table.

- List the name of the managers with the maximum employees

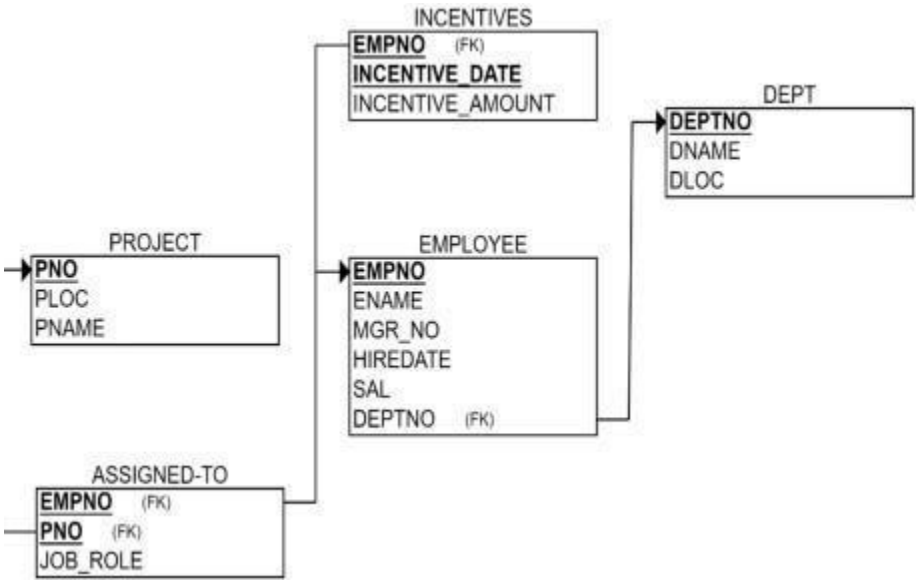
- Display those managers name whose salary is more than average salary of his employee

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

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

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

Schema Diagram:



create view Manager(Employee_Number, Employee_Name, Employee_Dept_No, Employee_Salary, Manager_Number, Manager_Name, Manager_Dept_No, Manager_Salary)
as select e.Emp_No, e.Dept_No, e.Ename, e.Salary, m.Emp_No, m.Ename, m.Dept_No, m.Salary
from Employee e, Employee m
where
e.MGR_No=m.Emp_No; select

Employee_Number	Employee_Name	Employee_Dept_No	Employee_Salary	Manager_Number	Manager_Name	Manager_Dept_No	Manager_Salary
4	2	Dinesh	45000	2	Balaji	2	200000
5	1	Eshwar	55000	1	Avinash	1	250000
6	3	Fazal	75000	3	Chandan	3	180000
7	1	Gajendra	56000	1	Avinash	1	250000
8	3	Habeebullah	30000	3	Chandan	3	180000
9	1	Inaytullah	50000	1	Avinash	1	250000

* from Manager;

List the name of the managers with the maximum employees

select Manager_Number, Manager_Name, COUNT(Employee_Number) AS Num_of_Employees
from Manager
group by Manager_Number
HAVING count(Employee_Number) = (select max(EmployeeCount) from (select count (Employee_Number) AS EmployeeCount
from Manager group by Manager_Number) as EmployeeCounts);

Manager_Number	Manager_Name	Num_of_Employees
1	Avinash	3

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

select distinct m.Manager_Number, m.Manager_name, m.Manager_Salary,
(select avg(Employee_Salary) from Manager e
where m.Manager_Number=e.Manager_Number) as Average_Employee_Salary
from Manager m where m.Manager_Salary>
(select avg(Employee_Salary) from Manager e
where m.Manager_Number=e.Manager_Number);

Manager_Number	Manager_Name	Manager_Salary	Average_Employee_Salary
2	Balaji	200000	45000
1	Avinash	250000	53666.666666666664
3	Chandan	180000	52500

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

```
select i.Emp_No, e.Ename, max(i.Incentive_Amount)
from Incentives i, Employee e
where e.Emp_No=i.Emp_No
and i.Incentive_date like
'2019-01-%'
group by i.Emp_No, e.Ename, i.Incentive_Date;
```

Emp_No	Ename	max(i.Incentive_Amount)
1	Avinash	10000

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

```
select e.Emp_No, e.Ename as Employee_Name, e.Dept_No, m.Ename AS
Manager_Name
from Employee
e join Employee
m
on e.MGR_No = m.Emp_No
where e.Dept_No =
```

Emp_No	Employee_Name	Dept_No	Manager_Name
4	Dinesh	2	Balaji
5	Eshwar	1	Avinash
6	Fazal	3	Chandan
7	Gajendra	1	Avinash
8	Habeebullah	3	Chandan
9	Inaytullah	1	Avinash

```
m.Dept_No;
```

Supplier Database

Questio

n (Week

7)

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

- Insert appropriate records in each table.

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

- Find the snames of suppliers who supply every part.

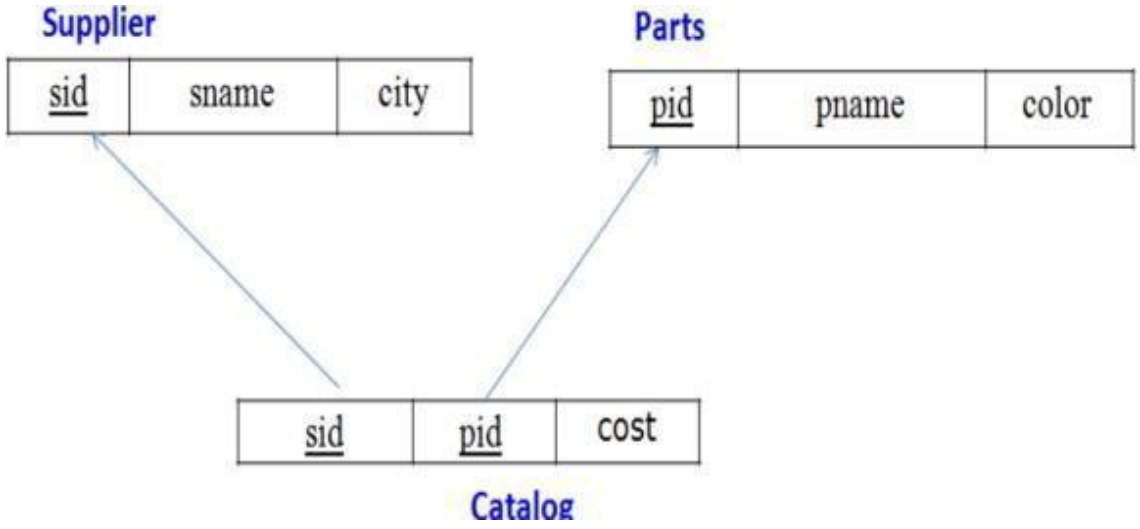
- Find the snames of suppliers who supply every red part.

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

- 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)

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

Schema Diagram:



Create Database:

```
create database supp;
use supp;
```

Create Tables:

```
 create table Supplier(
s_id int primary key,
s_name varchar(30),
city varchar(20));

create table Parts(
p_id int primary
key, p_name
varchar(30), color
varchar(30));

create table Catalog(
s_id int,
p_id int,
cost float,
foreign key(s_id) references Supplier(s_id),
foreign key(p_id) references Parts(p_id));
```

Structure of the Table:

desc Supplier;

	Field	Type	Null	Key	Default	Extra
▶	s_id	int	NO	PRI	NULL	
	s_name	varchar(30)	YES		NULL	
	city	varchar(20)	YES		NULL	

desc Parts;

	Field	Type	Null	Key	Default	Extra
▶	p_id	int	NO	PRI	NULL	
	p_name	varchar(30)	YES		NULL	
	color	varchar(30)	YES		NULL	

desc Catalog;

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

Inserting Values to the tables:

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

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

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

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

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

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

Queries:

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

```
select distinct p.p_name
from Supplier s, Catalog c, Parts p where
s.s_id = c.s_id and
p.p_id = c.p_id and
c.s_id is not null;
```

	p_name
▶	Book
	Pen
	Pencil
	Mobile
	Charger

Find the snames of suppliers who supply every part.

```
select distinct s_name
from Supplier s, Catalog c, Parts p
where s.s_id = c.s_id
group by s.s_id, s.s_name
having count(distinct c.p_id)=(select count(*) from Parts p);
```

	s_name
▶	Acme_Widget

Find the snames of suppliers who supply every red part.

```
select distinct s_name
from Supplier s, Catalog c, Parts p
where s.s_id = c.s_id and
c.p_id in (select p_id from Parts p where p.color = 'Red')
```

s_name
Johns
Acme_Widget

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

```
select distinct p_name from Supplier s, Parts p, Catalog c where
p.p_id in (select c.p_id from Catalog c, Supplier s where
s.s_id = c.s_id and s.s_name = 'Acme_Widget') and
p.p_id not in (select c.p_id from Catalog c, Supplier s where
s.s_id = c.s_id and s.s_name != 'Acme_Widget');
```

p_name
Mobile
Charger

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

```
create view Average(p_id, Average_Product_Cost) as
select c.p_id, avg(cost)
from Catalog c
group by
c.p_id;
 select c.s_id from Catalog c, Average a where
 c.p_id = a.p_id and
 c.cost>(a.Average_Product_Cost)
 group by c.p_id, c.s_id;
```

s_id
10002
10004

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

```
select distinct s.s_name, c.cost, c.p_id from Catalog c, Supplier s where
s.s_id = c.s_id and
c.cost in (select max(cost) from Catalog c group by c.p_id);
```

	s_name	cost	p_id
▶	Acme_Widget	10	20001
	Acme_Widget	10	20002
	Acme_Widget	10	20004
	Acme_Widget	10	20005
	Johns	10	20001
	Johns	20	20002
	Reliance	40	20003

No SQL Student Database

Questio

n (Week

8)

Perform the following DB operations using MongoDB.

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

- Insert appropriate values

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

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

Create Database:

db.createCollection("Student");

```
Atlas atlas-cci5oy-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-cci5oy-shard-0 [primary] test>
```

Inserting Values to the tables:

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe28cf2355f925cc449c9") }
}
```

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe295f2355f925cc449ca") }
}
```

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe29df2355f925cc449cb") }
}
```

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe2a5f2355f925cc449cc") }
}
```

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe2abf2355f925cc449cd") }
}
```

Queries:

db.Student.find()

```
Atlas atlas-cci5oy-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("6746b3bd3524069968624499"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3c73524069968624499a"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3d03524069968624499b"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3d83524069968624499c"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b3e13524069968624499d"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
```

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

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

```
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

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

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

```
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("675fe2cbf2355f925cc449ce") }
}
```

db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})

```
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

```
{
  _id: ObjectId("6746b41935240699686 2449e"),
  RollNo: 11,
  Age: 22,
  Name: 'FEM',
  Cont: 2276,
  email: 'rea.de9@gmail.com'
},
```

No SQL Customers Database

Questio

n (Week

9)

- Create a collection by name Customers with the following

  attributes. Cust_id, Acc_Bal, Acc_Type

- Insert at least 5 values into the table

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

- Determine Minimum and Maximum account balance for each customer_id.

- Export the created collection into local file system

- Drop the table

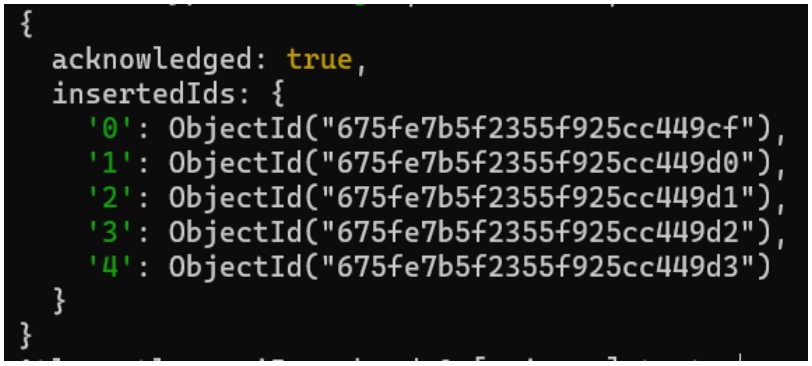
- Import a given csv dataset from local file system into mongodb collection.

Create Database:

db.createCollection("Customer");

Inserting Values to the tables:

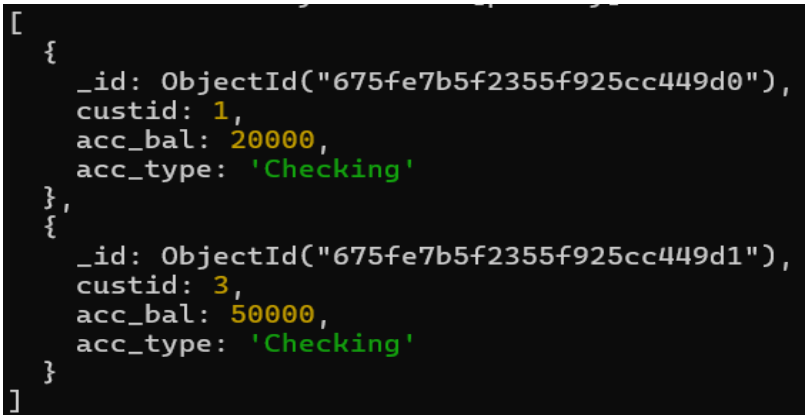
db.Customer.insertMany([{custid: 1, acc_bal:10000, acc_type:"Saving"}, {custid: 1, acc_bal:20000, acc_type: "Checking"}, {custid: 3, acc_bal:50000, acc_type: "Checking"}, {custid: 4, acc_bal:10000, acc_type: "Saving"}, {custid: 5, acc_bal:2000, acc_type: "Checking"}]);



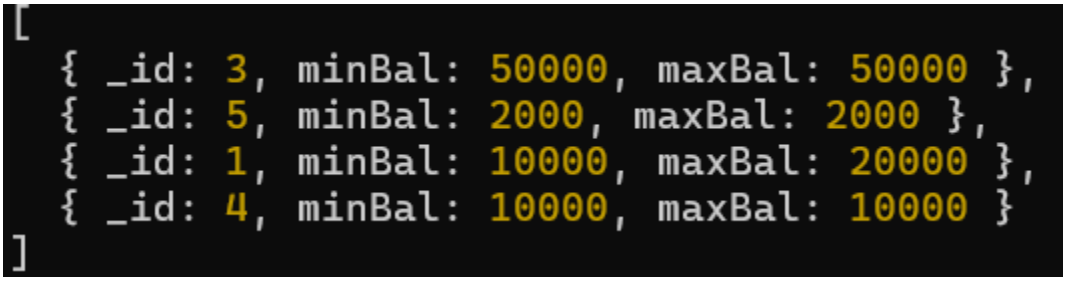
Queries:

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

db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});



Determine Minimum and Maximum account balance for each customer_id. db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal:{$max:"$acc_bal"}}}]);



db.Customers.drop()

```
true
```

mongoexport mongodb+srv://dbms:@cluster0.xmdk9.mongodb.net/test
--collection=Student --out C:\Users\BMSCECSE\Desktop\st.json

```
C:\Users\BMSCECSE\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoexport mongodb+srv://amithr028:Rangaram
2005@cluster0.o3wtn.mongodb.net/test --collection=Student --out C:\Users\ BMSCECSE  \Desktop\st.json
2024-12-16T14:30:01.812+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.o3wtn.mongodb.net/test
2024-12-16T14:30:01.876+0530    exported 5 records
```

mongoimport mongodb+srv://dbms:@cluster0.xmdk9.mongodb.net/test
--collection=New_Student --file C:\Users\BMSCECSE\Desktop\New_Student.json

```
C:\Users\BMSCECSE\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoimport mongodb+srv://amithr028:Rangaram
2005@cluster0.o3wtn.mongodb.net/test --collection=New_Student --file C:\Users\  BMSCECSE    \Desktop\New_Student.json
2024-12-16T14:33:27.107+0530    Failed: open C:\Users\amith\OneDrive\Desktop\New_Student.json: The system cannot find th
e file specified.
2024-12-16T14:33:27.109+0530    5 document(s) imported successfully. 0 document(s) failed to import.
```

No SQL Restaurants Database

Question

(Week 10)

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

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

- 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.

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

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

Create Database:

db.createCollection("restaurants");

```
{ ok: 1 }
```

Inserting Values to the tables:

db.restaurants.insertMany([{ name: "Meghna Foods", town: "Jayanagar", cuisine: "Indian", score: 8, address: { zipcode: "10001", street: "Jayanagar"}},{ name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode: "10100", street: "MG Road"}},{ name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar" }},{ name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" }},{ name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram"}}])

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("67600441f2355f925cc449d4"),
    '1': ObjectId("67600441f2355f925cc449d5"),
    '2': ObjectId("67600441f2355f925cc449d6"),
    '3': ObjectId("67600441f2355f925cc449d7"),
    '4': ObjectId("67600441f2355f925cc449d8")
  }
}
```

Queries:

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

db.restaurants.find({})

```
[
  {
    _id: ObjectId("67600441f2355f925cc449d4"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d5"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d6"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d7"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d8"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  }
]
```

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

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

```
[
  {
    _id: ObjectId("67600441f2355f925cc449d8"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Malleshwaram' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d4"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d7"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
    address: { zipcode: '10300', street: 'Majestic' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d5"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("67600441f2355f925cc449d6"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

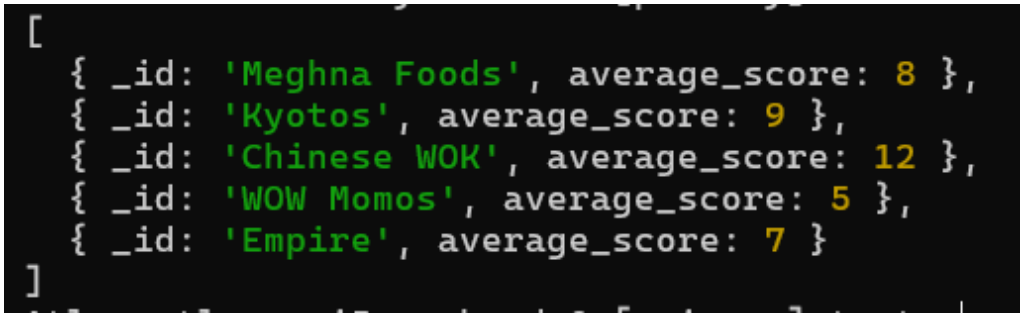
Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.

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

```
[
  {
    _id: ObjectId("67600441f2355f925cc449d4"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67600441f2355f925cc449d5"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("67600441f2355f925cc449d7"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId("67600441f2355f925cc449d8"),
    name: 'WOW Momos',
    town: 'Malleshwaram',
    cuisine: 'Indian'
  }
]
```

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

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



```
[
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Empire', average_score: 7 }
]
```

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

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

```
[
  { name: 'Meghna Foods', address: { street: 'Jayanagar' } },
  { name: 'Empire', address: { street: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
```