

VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.



LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

MAHESH GOPAL LAMANI (1BM24CS157)

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



B.M.S. COLLEGE OF ENGINEERING

(Autonomous Institution under VTU)

BENGALURU-560019

Sep-2024 to Jan-2025

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



CERTIFICATE

This is to certify that the Lab work entitled “Database Management Systems (23CS3PCDBM)” carried out by **MAHESH GOPAL LAMANI (1BM24CS157)**, 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 2022. 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.

Saritha A N Assistant Professor Department of CSE, BMSCE	Dr. Jyothi S Nayak Professor & HOD Department of CSE, BMSCE
--	---

Index

Sl. No.	Date	Experiment Title	Page No.
1	4-10-2024	Insurance Database	4 - 11
2	9-10-2024	More Queries on Insurance Database	12 - 13
3	16-10-2024	Bank Database	14 - 20
4	23-10-2024	More Queries on Bank Database	21 - 24
5	30-10-2024	Employee Database	25 - 30
6	13-11-2024	More Queries on Employee Database	31 - 32
7	20-11-2024	Supplier Database	33 - 37
8	27-11-2024	NO SQL - Student Database	38 - 39
9	4-12-2024	NO SQL - Customer Database	40 - 42
10	4-12-2024	NO SQL – Restaurant Database	43 - 45

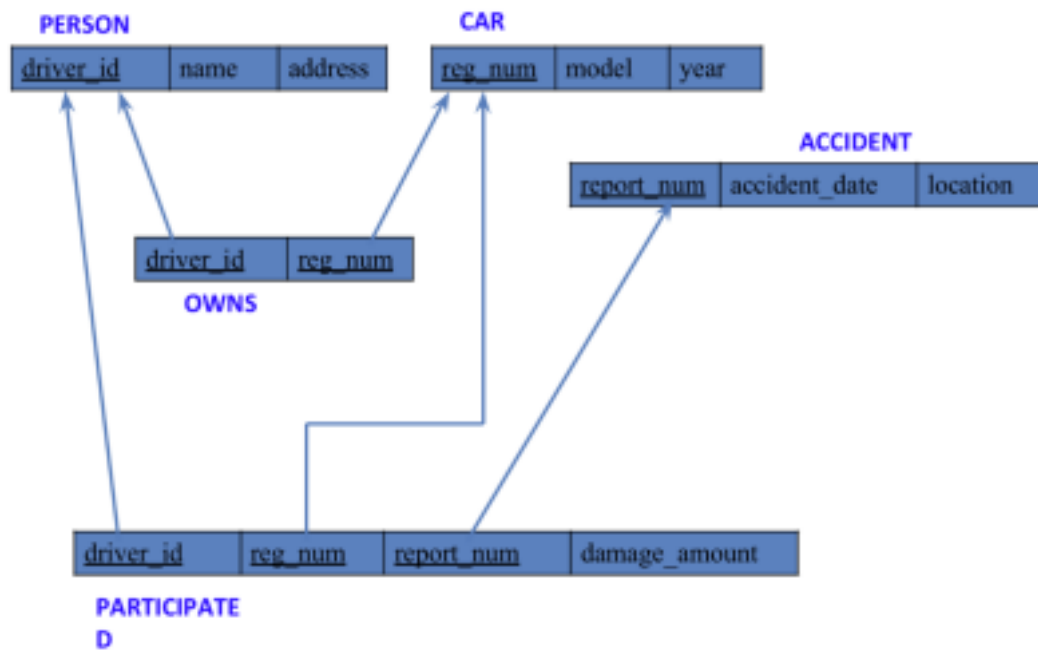
Insurance Database

Question

(Week 1)

- PERSON (driver_id: String, name: String, address: String)
- CAR (reg_num: String, model: String, year: int)
- ACCIDENT (report_num: int, accident_date: date, location: String)
- OWNS (driver_id: String, reg_num: String)
- PARTICIPATED (driver_id: String, reg_num: String, report_num: int, damage_amount: int)
- Create the above tables by properly specifying the primary keys and the foreign keys.
- Enter at least five tuples for each relation
- Display Accident date and location
- Update the damage amount to 25000 for the car with a specific reg_num (example 'KA053408') for which the accident report number was 12.
- Add a new accident to the database.
- To Do
- Display Accident date and location
- Display driver id who did accident with damage amount greater than or equal to Rs.25000

Schema Diagram



Create database

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

Create table

```
create table person  
(  
  driver_id varchar(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 accident;

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

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 owns;

	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(10)	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
•	NULL	NULL	NULL

```
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
•	NULL	NULL	NULL

```
insert into owns values('A01','KA052250');
```

```
insert into owns values('A02','KA053408');
```

```
insert into owns values('A03','KA031181');
```

```
insert into owns values('A04','KA095477');
```

```
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');
```

```
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,'2004-03-15','kanakapura road');
```

```
select * from accident;
```

	report_num	accident_date	location
▶	11	2003-01-01	Mysore road
	12	2004-02-02	south end
	13	2003-01-21	Bull temple road
	14	2008-02-17	Mysore road
	15	2004-03-15	kanakapura road
•	NULL	NULL	NULL

```
insert into participated values('A01','KA052250',11,10000);
```

```
insert into participated values('A02','KA053408',12,50000);
```

```
insert into participated values('A03','KA031181',13,25000);
```

```
insert into participated values('A04','KA095477',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	50000
	A03	KA031181	13	25000
	A04	KA095477	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;

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

- Add a new accident to the database.
insert into accident values(16,'2008-03-08',"Domlur");
select * from accident;

	report_num	accident_date	location
▶	11	2003-01-01	Mysore road
	12	2004-02-02	south end
	13	2003-01-21	Bull temple road
	14	2008-02-17	Mysore road
	15	2004-03-15	kanakapura road
	16	2008-03-08	Domlur
•	NULL	NULL	NULL

TO DO

- **Display Accident date and location.**

select accident_date, location from accident;

	accident_date	location
▶	2003-01-01	Mysore road
	2004-02-02	south end
	2003-01-21	Bull temple road
	2008-02-17	Mysore road
	2004-03-15	kanakapura road

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

	driver_id
▶	A02
	A03

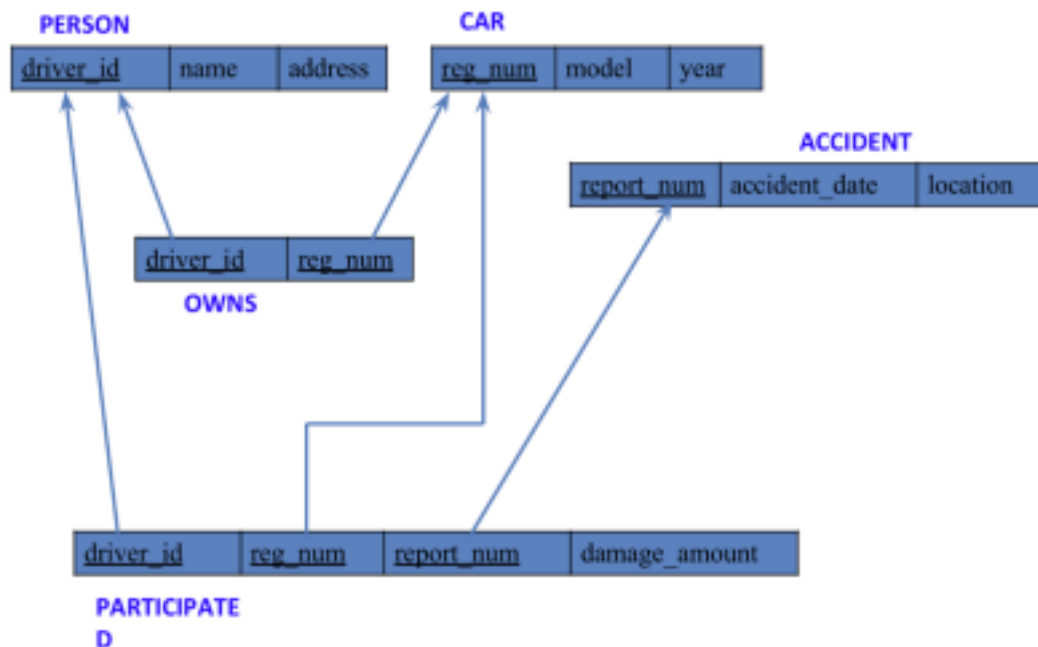
More Queries on Insurance Database

Question

(Week 2)

- PERSON (driver_id: String, name: String, address: String)
- CAR (reg_num: String, model: String, year: int)
- ACCIDENT (report_num: int, accident_date: date, location: String)
- OWNS (driver_id: String, reg_num: String)
- PARTICIPATED (driver_id: String, reg_num: String, report_num: int, damage_amount: int)
- LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.
- FIND THE AVERAGE DAMAGE AMOUNT.
- LIST THE NAME OF DRIVERS WHOSE DAMAGE IS GREATER THAN THE AVERAGE DAMAGE AMOUNT.
- FIND MAXIMUM DAMAGE AMOUNT.

Schema Diagram



Queries

- **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	KA031181	13	25000
	A01	KA052250	11	10000
	A05	KA041702	15	5000
	A04	KA095477	14	3000

- **FIND THE AVERAGE DAMAGE AMOUNT.**

```
SELECT AVG(damage_amount) FROM participated;
```

	AVG(damage_amount)
▶	13600.0000

- **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 MAX(damage_amount) FROM participated;
```

	MAX(damage_amount)
▶	25000

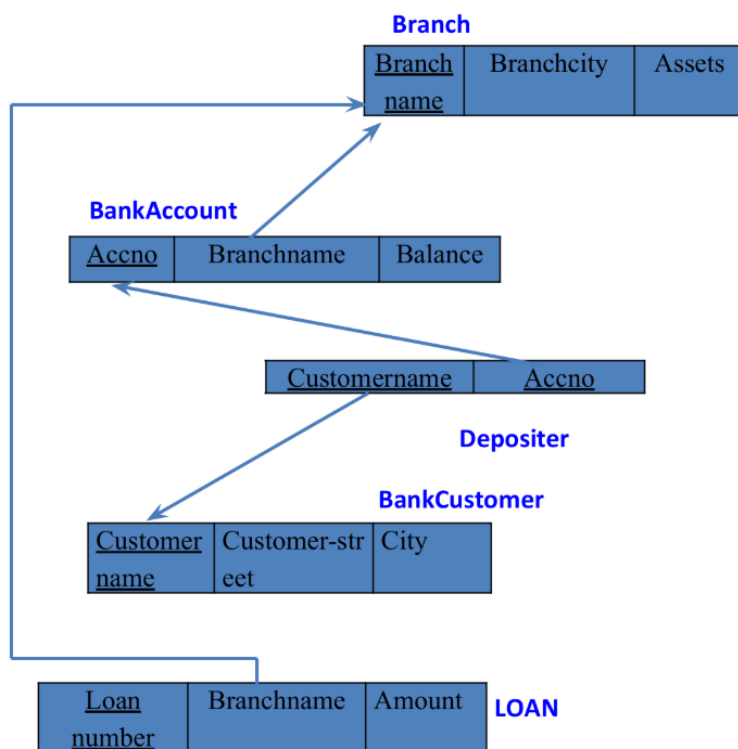
Bank Database

Question

(Week 3)

- Branch (branch-name: String, branch-city: String, assets: real)
- BankAccount(accno: int, branch-name: String, balance: real)
- BankCustomer (customer-name: String, customer-street: String, customer-city: String)
- Depositer(customer-name: String, accno: int)
- Loan (loan-number: int, branch-name: String, amount: real)
- 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 bank;
```

```
use bank;
```

Create table

```
create table branch
(
branch_name varchar(25),
branch_city varchar(15),
assets int,
primary key(branch_name)
);

create table bankAccount
(
accno int,
branch_name varchar(25),
balance int,
primary key(accno),
foreign key(branch_name) references branch(branch_name)
);

create table bankCustomer
(
customer_name varchar(10),
customer_street varchar(25),
customer_city varchar(15),
primary key(customer_name)
);
```

```
create table depositer
```

```
(
```

```

customer_name varchar(10),
accno int,
foreign key(customer_name) references bankCustomer(customer_name),
foreign key(accno) references bankAccount(accno)
);
create table loan
(
loan_number int,
branch_name varchar(25),
amount int,
primary key(loan_number),
foreign key(branch_name) references branch(branch_name)
);

```

Structure of the table

desc branch;

	Field	Type	Null	Key	Default	Extra
▶	branch_name	varchar(25)	NO	PRI	NULL	
	branch_city	varchar(15)	YES		NULL	
	assets	int	YES		NULL	

desc bankAccount;

	Field	Type	Null	Key	Default	Extra
▶	accno	int	NO	PRI	NULL	
	branch_name	varchar(25)	YES	MUL	NULL	
	balance	int	YES		NULL	

desc bankCustomer;

	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(10)	NO	PRI	NULL	
	customer_street	varchar(25)	YES		NULL	
	customer_city	varchar(15)	YES		NULL	

desc depositer;

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

desc loan;

	Field	Type	Null	Key	Default	Extra
►	loan_number	int	NO	PRI	NULL	
	branch_name	varchar(25)	YES	MUL	NULL	
	amount	int	YES		NULL	

Inserting Values to the table

```
insert into branch values('SBI_chamrajpet','Bangalore',5000);
insert into branch values('SBI_residencyRoad','Bangalore',10000);
insert into branch values('SBI_shivajiRoad','Bombay',20000);
insert into branch values('SBI_parlimentRoad','delhi',10000);
insert into branch values('SBI_jantarmanantar','delhi',20000);
select * from branch;
```

	branch_name	branch_city	assets
►	SBI_chamrajpet	Bangalore	5000
	SBI_jantarmanantar	delhi	20000
	SBI_parlimentRoad	delhi	10000
	SBI_residencyRoad	Bangalore	10000
	SBI_shivajiRoad	Bombay	20000
*	NULL	NULL	NULL

```
insert into bankAccount values(1,'SBI_chamrajpet',2000);
insert into bankAccount values(2,'SBI_residencyRoad',5000);
insert into bankAccount values(3,'SBI_shivajiRoad',6000);
insert into bankAccount values(4,'SBI_parlimentRoad',9000);
insert into bankAccount values(5,'SBI_jantarmanantar',8000);
insert into bankAccount values(6,'SBI_shivajiRoad',4000);
insert into bankAccount values(8,'SBI_residencyRoad',4000);
insert into bankAccount values(9,'SBI_parlimentRoad',3000);
insert into bankAccount values(10,'SBI_residencyRoad',5000);
insert into bankAccount values(11,'SBI_jantarmanantar',2000);
select * from bankAccount;
```

	accno	branch_name	balance
▶	1	SBI_chamrajpet	2000
	2	SBI_residencyRoad	5000
	3	SBI_shivajiRoad	6000
	4	SBI_parlimentRoad	9000
	5	SBI_jantarmantar	8000
	6	SBI_shivajiRoad	4000
	8	SBI_residencyRoad	4000
	9	SBI_parlimentRoad	3000
	10	SBI_residencyRoad	5000
	11	SBI_jantarmantar	2000
★	NULL	NULL	NULL

```

insert into bankCustomer values('avinash','BullTempleRoad','Bangalore');
insert into bankCustomer values('dinesh','BannerghattaRoad','Bangalore');
insert into bankCustomer values('mohan','nationalCollegeRoad','Bangalore');
insert into bankCustomer values('nikil','AkbarRoad','Delhi');
insert into bankCustomer values('ravi','prativirajRoad','Delhi');
select * from bankCustomer;

```

	customer_name	customer_street	customer_city
▶	avinash	BullTempleRoad	Bangalore
	dinesh	BannerghattaRoad	Bangalore
	mohan	nationalCollegeRoad	Bangalore
	nikil	AkbarRoad	Delhi
	ravi	prativirajRoad	Delhi
★	NULL	NULL	NULL

```

insert into depositer values('avinash',1);
insert into depositer values('dinesh',2);
insert into depositer values('nikil',4);
insert into depositer values('ravi',5);
insert into depositer values('avinash',8);
insert into depositer values('nikil',9);
insert into depositer values('dinesh',10);
insert into depositer values('nikil',11);
select * from depositer;

```

	customer_name	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);
insert into loan values(2,'SBI_residencyRoad',2000);
insert into loan values(3,'SBI_shivajiRoad',3000);
insert into loan values(4,'SBI_parlimentRoad',4000);
insert into loan values(5,'SBI_jantarmentar',5000);
select * from loan;

```

	loan_number	branch_name	amount
▶	1	SBI_chamrajpet	1000
	2	SBI_residencyRoad	2000
	3	SBI_shivajiRoad	3000
	4	SBI_parlimentRoad	4000
	5	SBI_jantarmentar	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'.**

alter table branch rename column assets to assets_inlakhs;

select branch_name, assets_inlakhs from branch;

	branch_name	assets_inlakhs
▶	SBI_chamrajpet	5000
	SBI_jantarmentar	20000
	SBI_parliamentRoad	10000
	SBI_residencyRoad	10000
	SBI_shivajiRoad	20000
*	NULL	NULL

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

select d.customer_name from depositor d, bankAccount b

where b.branch_name='SBI_residencyRoad' and d.accno=b.accno group by d.customer_name

having count(d.accno)>=2;

	customer_name
▶	dinesh

- **CREATE A VIEW WHICH GIVES EACH BRANCH THE SUM OF THE AMOUNT OF ALL THE LOANS AT THE BRANCH.**

create view br

as

select branch_name, sum(amount) from loan

group by branch_name;

select * from br;

	branch_name	sum(amount)
▶	SBI_chamrajpet	1000
	SBI_jantarmentar	5000
	SBI_parliamentRoad	4000
	SBI_residencyRoad	2000
	SBI_shivajiRoad	3000

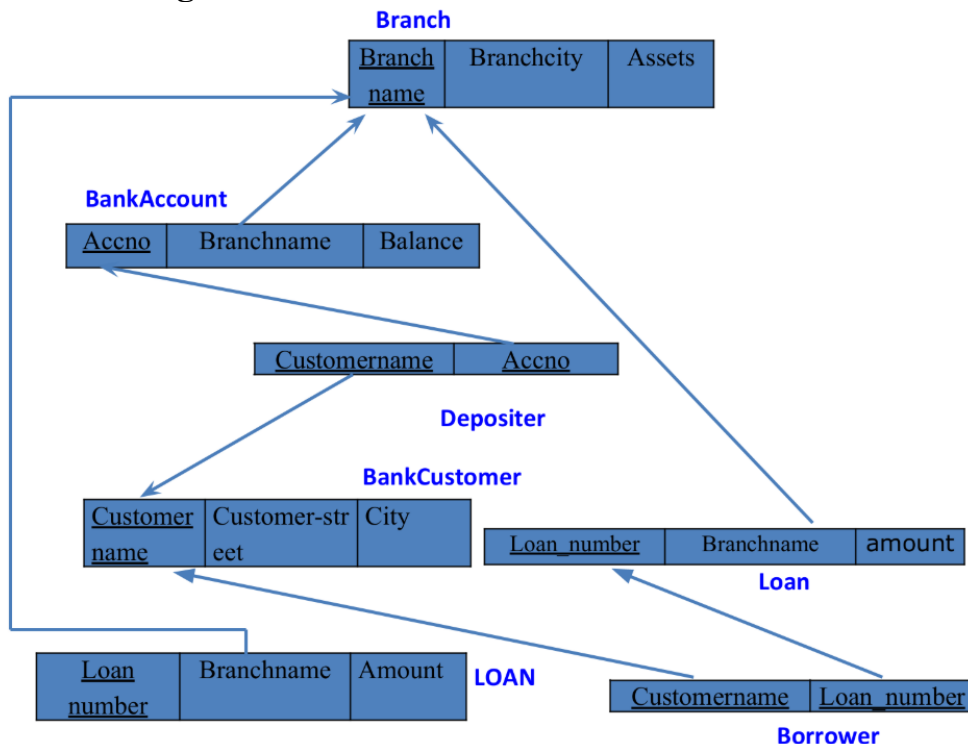
More Queries on Bank Database

Question

(Week 4)

- Branch (branch-name: String, branch-city: String, assets: real)
- BankAccount(accno: int, branch-name: String, balance: real)
- BankCustomer (customer-name: String, customer-street: String, customer-city: String)
- Depositer(customer-name: String, accno: int)
- Loan (loan-number: int, branch-name: String, amount: real)
- Borrower (customer-name: String, loan-number: int)
- Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).
- Find all customers who have a loan at the bank but do not have an account.
- Find all customers who have both an account and a loan at the Bangalore branch
- Find the names of all branches that have greater assets than all branches located in Bangalore.
- Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).
- Update the Balance of all accounts by 5%

Schema Diagram



Create table

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

Structure of table

desc borrower;

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

Insert values to the table

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

Queries

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

```
select distinct d.customer_name
from depositor d,bankAccount ba,branch b
where d.accno=ba.accno and ba.branch_name=b.branch_name and b.branch_city='delhi'
group by d.customer_name having count(b.branch_name)>1;
```

	customer_name
▶	nikil

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

```
select b.customer_name
from borrower b
where b.loan_number not in (select d.accno from depositor d where b.loan_number=d.accno);
```

	customer_name
▶	Mohan

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

```
select b.customer_name
from borrower b
where b.loan_number in (select d.accno from depositor d, bankAccount ba, branch b where
b.loan_number=d.accno and d.accno=ba.accno and ba.branch_name=b.branch_name and
b.branch_city='Bangalore');
```

	customer_name
▶	Avinash
	Dinesh

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

```
select branch_name
from branch
where assets_inlakhs > all (select assets_inlakhs from branch where branch_city='Bangalore');
```

	branch_name
▶	SBI_jantarmantar
	SBI_MantriMarg
	SBI_shivajiRoad
✱	NULL

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

```
delete from bankAccount ba
```


where ba.branch_name=(select b.branch_name from branch b where branch_city='Bombay');
select * from bankAccount;

	accno	branch_name	balance
▶	1	SBI_chamrajpet	2000
	2	SBI_residencyRoad	5000
	4	SBI_parlimentRoad	9000
	5	SBI_jantarmantar	8000
	8	SBI_residencyRoad	4000
	9	SBI_parlimentRoad	3000
	10	SBI_residencyRoad	5000
	11	SBI_jantarmantar	2000
	12	SBI_MantriMarg	2000
*	NULL	NULL	NULL

- **Update the Balance of all accounts by 5%**

update bankAccount

set balance=balance+((5*balance)/100);

select * from bankAccount;

	accno	name	balance
▶	1	SBI_Chamrajpet	2100
	2	SBI_Residency road	5250
	3	SBI_Shivaji road	6300
	4	SBI_Parliament road	9450
	5	SBI_Jantarmantar	8400
	6	SBI_Shivaji road	4200
	8	SBI_Residency road	4200
	9	SBI_Parliament road	3150
	10	SBI_Residency road	5250
	11	SBI_Jantarmantar	2100
	12	SBI_MantriMarg	2100
*	NULL	NULL	NULL

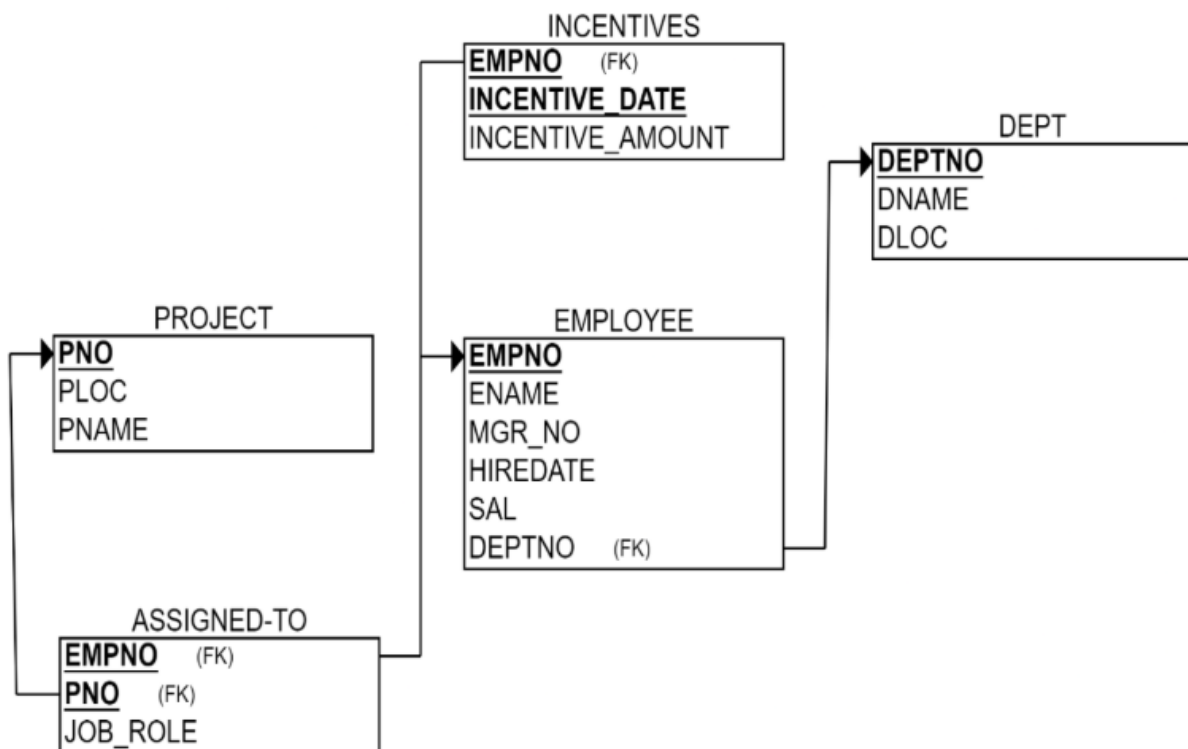
Employee Database

Question

(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 Employee;
use Employee;
```

Create table

```
create table dept
(
  deptno int,
  dname varchar(20),
  dloc varchar(20),
  primary key(deptno)
);
create table employee
(
  empno int,
  ename varchar(20),
  mgr_no int,
  hiredate varchar(20),
  sal int,
  deptno int,
  primary key(empno,deptno),
  foreign key(deptno) references dept(deptno)
);
create table incentives
(
  empno int,
  incentives_date varchar(20),
  amount int,
  primary key(empno,incentives_date),
  foreign key(empno) references employee(empno)
);
create table project
(
  pno int,
  ploc varchar(20),
  pname varchar(20),
  primary key(pno)
);
create table assignedto
(
  empno int,
  pno int,
  job_role varchar(20),
  primary key(empno,pno),
  foreign key(empno) references employee(empno),
  foreign key(pno) references project(pno));
```

Structure of the table

```
desc dept;
```

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

desc employee;

	Field	Type	Null	Key	Default	Extra
►	empno	int	NO	PRI	NULL	
	ename	varchar(20)	YES		NULL	
	mgr_no	int	YES		NULL	
	hiredate	varchar(20)	YES		NULL	
	sal	int	YES		NULL	
	deptno	int	NO	PRI	NULL	

desc incentives;

	Field	Type	Null	Key	Default	Extra
►	empno	int	NO	PRI	NULL	
	incentives_date	varchar(20)	NO	PRI	NULL	
	amount	int	YES		NULL	

desc project;

	Field	Type	Null	Key	Default	Extra
►	pno	int	NO	PRI	NULL	
	ploc	varchar(20)	YES		NULL	
	pname	varchar(20)	YES		NULL	

desc assignedto;

	Field	Type	Null	Key	Default	Extra
►	empno	int	NO	PRI	NULL	
	pno	int	NO	PRI	NULL	
	job_role	varchar(20)	YES		NULL	

Insert values to the tables

```
insert into dept values(1,'HR','new_delhi');
insert into dept values(2,'IT','bangalore');
insert into dept values(3,'Finance','mysuru');
```

```

insert into dept values(4,'development','hyderabad');
insert into dept values(5,'marketing','new_delhi');
select * from dept;

```

	deptno	dname	dloc
▶	1	HR	new_delhi
	2	IT	bangalore
	3	Finance	mysuru
	4	development	hyderabad
	5	marketing	new_delhi
*	NULL	NULL	NULL

```

insert into employee values(101,'raj',100,"12/01/1999",100000,1);
insert into employee values(201,'adhi',200,"17/01/2020",50000,2);
insert into employee values(301,'priyam',100,"01/09/2004",30000,3);
insert into employee values(401,'asha',101,"03/08/2000",10000,4);
insert into employee values(501,'shailesh',101,"29/2/2008",90000,5);
insert into employee values(601,'likith',102,"29/2/2008",90000,1);
select * from employee;

```

	empno	ename	mgr_no	hiredate	sal	deptno
▶	101	raj	100	12/01/1999	100000	1
	201	adhi	200	17/01/2020	50000	2
	301	priyam	100	01/09/2004	30000	3
	401	asha	101	03/08/2000	10000	4
	501	shailesh	101	29/2/2008	90000	5
	601	likith	102	29/2/2008	90000	1
*	NULL	NULL	NULL	NULL	NULL	NULL

```

insert into incentives values(101,"12/03/2004",50000);
insert into incentives values(201,"17/03/2024",25000);
insert into incentives values(301,"01/12/2019",15000);
insert into incentives values(401,"03/11/2019",5000);
insert into incentives values(501,"29/4/2019",45000);
select * from incentives;

```

	empno	incentives_date	amount
▶	101	12/03/2004	50000
	201	17/03/2024	25000
	301	01/12/2019	15000
	401	03/11/2019	5000
	501	29/4/2019	45000
★	NULL	NULL	NULL

```

insert into project values(10,"new_delhi","chatbot");
insert into project values(40,"bangalore","ml model");
insert into project values(50,"mysuru","blockchain");
insert into project values(30,"hyderabad","stocks");
insert into project values(80,"new_delhi","android app");
select * from project;

```

	pno	ploc	pname
▶	10	new_delhi	chatbot
	30	hyderabad	stocks
	40	bangalore	ml model
	50	mysuru	blockchain
	80	new_delhi	android app
★	NULL	NULL	NULL

```

insert into assignedto values(101,10,"devops");
insert into assignedto values(201,40,"sde");
insert into assignedto values(301,50,"manager");
insert into assignedto values(401,30,"jpa");
insert into assignedto values(501,80,"pa");
select * from assignedto;

```

	empno	pno	job_role
▶	101	10	devops
	201	40	sde
	301	50	manager
	401	30	jpa
	501	80	pa
★	NULL	NULL	NULL

Queries

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

```
select a.empno from assignedto a,project p
```

```
where a.pno=p.pno and (ploc='bangalore' or ploc='hyderabad' or ploc='mysuru');
```

	empno
▶	401
	201
	301

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

```
select e.empno from employee e
```

```
where e.empno!=all(select i.empno from incentives i);
```

	empno
▶	601

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

```
select e.empno,e.ename,d.dname,a.job_role,d.dloc,p.ploc
```

```
from employee e,dept d,assignedto a,project p
```

```
where e.deptno=d.deptno and e.empno=a.empno and a.pno=p.pno and d.dloc=p.ploc;
```

	empno	ename	dname	job_role	dloc	ploc
▶	101	raj	HR	devops	new_delhi	new_delhi
	201	adhi	IT	sde	bangalore	bangalore
	301	priyam	Finance	manager	mysuru	mysuru
	401	asha	development	jpa	hyderabad	hyderabad
	501	shailesh	marketing	pa	new_delhi	new_delhi

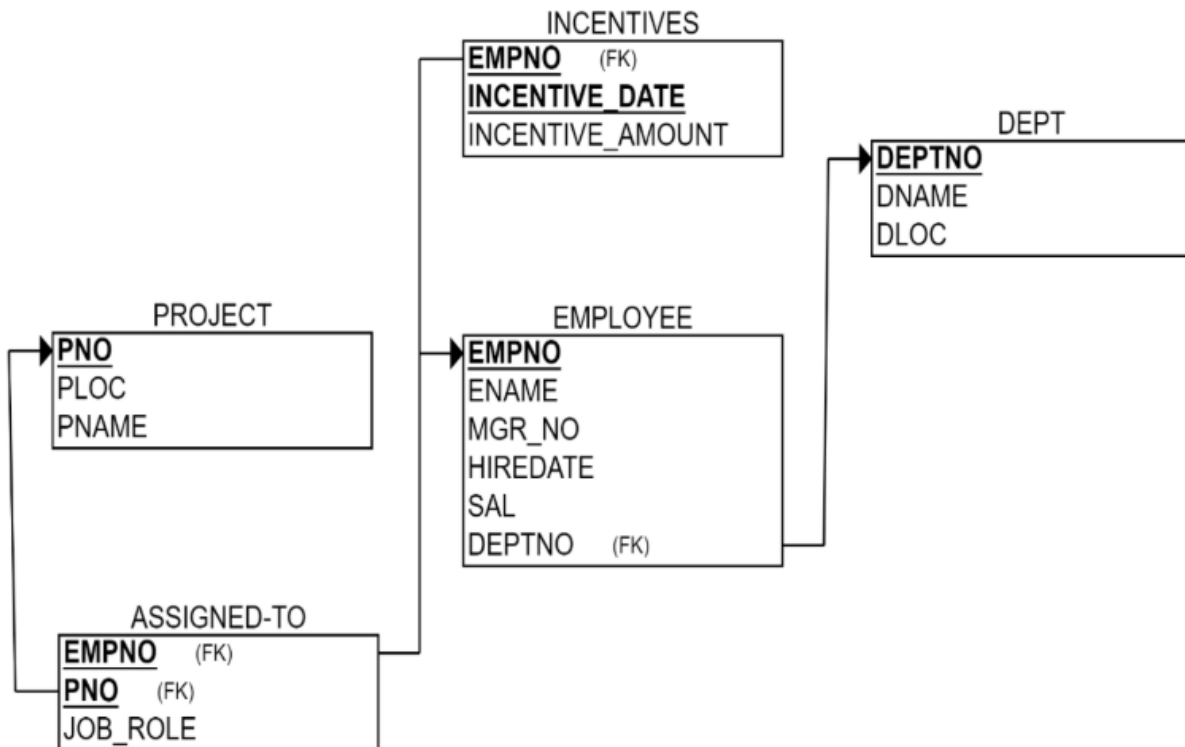
More Queries on Employee Database

Question

(Week 6)

- List the name of the managers with the maximum employees
- Display those managers' names of the manager whose salary is more than the 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 manager is working.

Schema Diagram



Queries

- **List the name of the managers with the maximum employees**
`select e.ename from employee e where e.empno in (select m.mgr_no from employee m group by m.mgr_no having count(*) = (select max(emp_count) from (select count(*) as emp_count from employee em group by em.mgr_no) as emp_count_subquery));`

	ename
▶	raj

- **Display those managers' names of the manager whose salary is more than the average salary of his employee.**

select e.ename from employee e where e.sal > (select avg(sub.sal) from employee sub where sub.mgr_no = e.empno);

	ename
▶	raj

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

select ename from employee where sal = (select max(sal) from employee where sal < (select max(sal) from employee));

	ename
▶	shailesh
	likith
	likith

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

select * from employee where empno = (select empno from incentives where amount = (select max(amount) from incentives where amount < (select max(amount) from incentives)));

	empno	ename	mgr_no	hiredate	sal	deptno
▶	501	shailesh	101	29/2/2008	90000	5
*	NULL	NULL	NULL	NULL	NULL	NULL

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

select e.ename from employee e , employee m where e.mgr_no = m.empno and e.deptno = m.deptno;

	ename
▶	likith

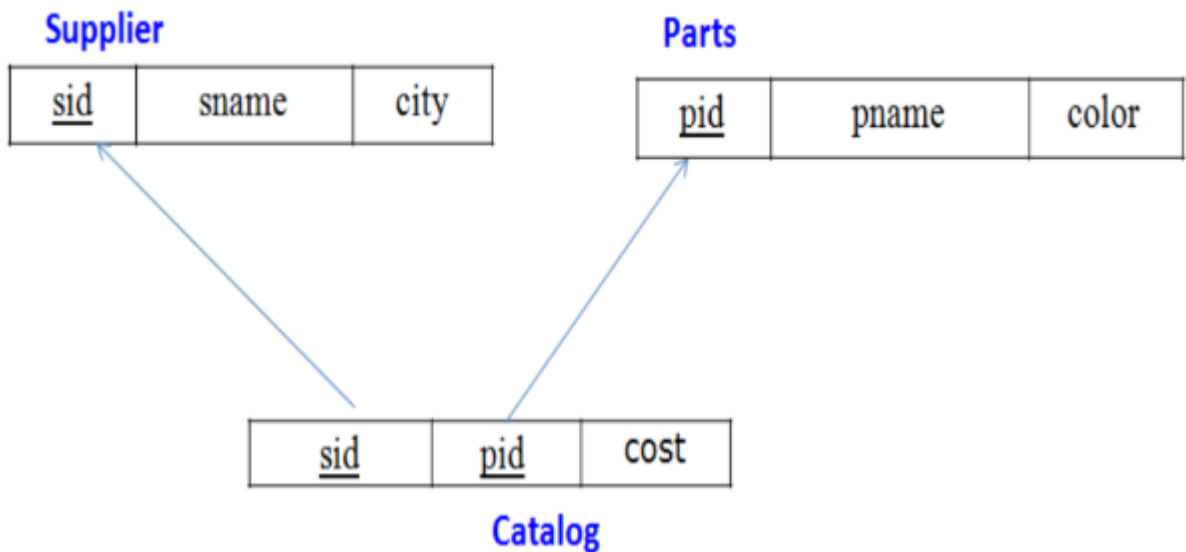
Supplier Database

Question

(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 supplier_database;  
use supplier_database;
```

Create table

```
create table supplier(  
  sid int,  
  sname varchar(20),  
  city varchar(20),  
  primary key(sid));  
create table parts(  
  pid int,  
  pname varchar(20),  
  color varchar(20),  
  primary key(pid));  
  
create table catalog(  
  sid int,  
  pid int,  
  cost int,  
  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(20)	YES		NULL	
	city	varchar(20)	YES		NULL	

desc parts;

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

desc catalog;

	Field	Type	Null	Key	Default	Extra
►	sid	int	YES	MUL	NULL	
	pid	int	YES	MUL	NULL	
	cost	int	YES		NULL	

Insert values to the tables

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

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

```

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

```

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

```

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

```

	sid	pid	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.pname from parts p,catalog c where p.pid=c.pid;

	pname
▶	Book
	Pen
	Pencil
	Mobile
	Charger

- Find the snames of suppliers who supply every part.

select distinct s.sname from supplier s

where Not exists(select p.pid from parts p

where not exists(select c.sid from catalog c where c.sid=s.sid and p.pid=c.pid));

	sname
▶	Acme Widget

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

select s.sname from supplier s

where Not exists(select p.pid from parts p

where p.color='red' and Not exists(select c.sid from catalog c where c.sid=s.sid and p.pid=c.pid));

	sname
▶	Acme Widget
	Johns

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

select p.pname from parts p,catalog c,supplier s

where c.sid=s.sid and p.pid=c.pid and s.sname='Acme Widget' and not exists(

select * from catalog c1,supplier s1 where c1.sid=s1.sid and p.pid=c1.pid and s1.sname!='Acme Widget');

	pname
▶	Mobile
	Charger

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

part (averaged over all the suppliers who supply that part).

```
select distinct c.sid from catalog c
```

```
where c.cost > (select avg(c1.cost) from catalog c1 where c1.pid=c.pid);
```

	sid
▶	10002
	10004

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

```
select p.pid,s.sname from parts p,catalog c,supplier s
```

```
where c.sid=s.sid and p.pid=c.pid and c.cost=(select max(c1.cost) from catalog c1 where c1.pid=p.pid);
```

	pid	sname
▶	20001	Acme Widget
	20004	Acme Widget
	20005	Acme Widget
	20001	Johns
	20002	Johns
	20003	Reliance

Question

(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 a query to update the Email-Id of a student with rollno 10.
- Replace the student name from "ABC" to "FEM" of rollno 11.
- Export the created table into local file system
- Drop the table
- Import a given csv dataset from the local file system into mongodb collection.

Create database

```
db.createCollection("Student");
```

Create table & Inserting Values to the table

```
db.Student.insertMany([
  {rollno:1,age:21,cont:9876,email:"anthara.de9@gmail.com"},
  {rollno:2,age:22,cont:9976,email:"anushka.de9@gmail.com"},
  {rollno:3,age:21,cont:5576,email:"anubhav.de9@gmail.com"},
  {rollno:10,age:20,cont:2276,email:"rekha.de9@gmail.com"}]);
db.student.find()
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.find()
[
  {
    _id: ObjectId("6746b87366152224f4779211"),
    RollNo: 1,
    Age: 21,
    Const: 9876,
    email: 'anthara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8ac66152224f4779212"),
    RollNo: 2,
    Age: 22,
    Const: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8d266152224f4779213"),
    RollNo: 3,
    Age: 21,
    Const: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8f166152224f4779214"),
    RollNo: 10,
    Age: 20,
    Const: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

Queries

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

`db.Student.update({rollno:5},{ $set:{email:"abhinav@gmail.com"}})`

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.update({RollNo:10},{ $set:{email:"Abhinav@gmail.com"}})
DeprecationWarning: Collection.update() is deprecated. Use updateOne, updateMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.find()
[
  {
    _id: ObjectId("6746b87366152224f4779211"),
    RollNo: 1,
    Age: 21,
    Const: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8ac66152224f4779212"),
    RollNo: 2,
    Age: 22,
    Const: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8d266152224f4779213"),
    RollNo: 3,
    Age: 21,
    Const: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8f166152224f4779214"),
    RollNo: 10,
    Age: 20,
    Const: 2276,
    email: 'Abhinav@gmail.com'
  }
]
```

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

`db.Student.insert({rollno:11,age:22,name:"ABC",cont:2276,email:"rea.de9@gmail.com"});`

```
db.Student.update({rollno:11,name:"ABC"},{$set:{name:"FEM"}})
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}})
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-10jjz6-shard-0 [primary] test> db.student.find()
[
  {
    _id: ObjectId("6746b87366152224f4779211"),
    RollNo: 1,
    Age: 21,
    Const: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8ac66152224f4779212"),
    RollNo: 2,
    Age: 22,
    Const: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8d266152224f4779213"),
    RollNo: 3,
    Age: 21,
    Const: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b8f166152224f4779214"),
    RollNo: 10,
    Age: 20,
    Const: 2276,
    email: 'Abhinav@gmail.com'
  },
  {
    _id: ObjectId("6746ba1266152224f4779215"),
    RollNo: 11,
    Age: 22,
    Const: 2276,
    email: 'rea.de9@gmail.com',
    Name: 'FEM'
  }
]
```

NO SQL - Customer Database

Question

(Week 9)

Perform the following DB operations using MongoDB.

- 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 'Checking' for each customer_id.
- Determine Minimum and Maximum account balance for each customer_id.
- Export the created collection into the local file system.
- Drop the table.
- Import a given csv dataset from the local file system into mongodb collection.

Create database

```
db.createCollection("Customer");
```

Inserting Values:

```
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"}]);
db.Customer.find();
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.Customer.find()
[
  {
    _id: ObjectId("6751fde06a59c75535ff9949"),
    custid: 1,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994a"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994b"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994c"),
    custid: 4,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId("6751fde06a59c75535ff994d"),
    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 'Checking' for each customer_id.

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

```
[test> db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});
[
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4c'),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4d'),
    custid: 3,
    acc_bal: 50000,
    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"}}}]);
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.Customer.aggregate([{$group: {_id:"$custid", minBal: {$min:"$acc_bal"}, maxBal:
... {$max:"$acc_bal"}}}]);
[
  { _id: 1, minBal: 10000, maxBal: 20000 },
  { _id: 4, minBal: 10000, maxBal: 10000 },
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 5, minBal: 2000, maxBal: 2000 }
]
```

- Export the created collection into the local file system.

```
mongoexport mongodb+srv://msshailshcs23:@cluster0.wujqr.mongodb.net/test --
collection=Customer --out C:\Users\shail\OneDrive\Desktop\st.json
```

```
C:\Users\shail\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\mongodb-database-tools-windows-x86_64-100.10.0\b
in>mongoexport mongodb+srv://msshailshcs23:@cluster0.wujqr.mongodb.net/test --collection=Student --out C:\Users\shail\O
neDrive\Desktop\st.json
Enter password for mongo user:

2024-12-14T21:56:18.781+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.wujqr.mongodb.net/test
2024-12-14T21:56:18.860+0530    exported 0 records

C:\Users\shail\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\mongodb-database-tools-windows-x86_64-100.10.0\b
in>
```

- Drop the table.

```
db.Customer.drop();
```

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

```
mongoimport mongodb+srv://msshailshcs23:@cluster0.wujqr.mongodb.net/test --
collection=New_Customer --file C:\Users\shail\OneDrive\Desktop\st.json
```

```
C:\Users\shail\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\mongodb-database-tools-windows-x86_64-100.10.0\bin>mongoimport mongodb+srv://msshailshcs23:cluster0.wujqr.mongodb.net/test --collection=New_Customer --file C:\Users\shail\OneDrive\Desktop\st.json
Enter password for mongo user:

2024-12-14T22:48:49.039+0530    connected to: mongodb+srv://[**REDACTED**]@cluster0.wujqr.mongodb.net/test
2024-12-14T22:48:49.149+0530    5 document(s) imported successfully. 0 document(s) failed to import.

C:\Users\shail\Downloads\mongodb-database-tools-windows-x86_64-100.10.0\mongodb-database-tools-windows-x86_64-100.10.0\bin>S
```

NO SQL - Restaurant 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 order 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("Restaurant");
```

Inserting Values:

```
db.Restaurant.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" } } ]}
```

Queries

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

```
db.Restaurant.find();
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.insertMany([
... {name:"Meghna Foods",town:"Jayanagar", cuisine: "Indian", score: 8, address: { zipcode: "10001", street: "Jayanagar" }},
... { name: "Empire", town: "MG Road", cuisine: "Indian", score: 7, address: { zipcode: "10100", street: "MG Road" } },
... { name: "Chinese WOK", town: "Indiranagar", cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar" }},
... { name: "Kyotos", town: "Majestic", cuisine: "Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" } },
... { name: "WOW Momos", town: "Malleshwaram", cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" } }])
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("6751f5566a59c75535ff9944"),
    '1': ObjectId("6751f5566a59c75535ff9945"),
    '2': ObjectId("6751f5566a59c75535ff9946"),
    '3': ObjectId("6751f5566a59c75535ff9947"),
    '4': ObjectId("6751f5566a59c75535ff9948")
  }
}
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({})
[
  {
    _id: ObjectId("6751f5566a59c75535ff9944"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9945"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian',
    score: 7,
    address: { zipcode: '10100', street: 'MG Road' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9946"),
    name: 'Chinese WOK',
    town: 'Indiranagar',
    cuisine: 'Chinese',
    score: 12,
    address: { zipcode: '20000', street: 'Indiranagar' }
  }
]
```

- Write a MongoDB query to arrange the name of the restaurants in descending order

along with all the columns.

```
db.Restaurant.find().sort({ "name": -1 });
```

```
{
  _id: ObjectId("6751f5566a59c75535ff9947"),
  name: 'Kyotos',
  town: 'Majestic',
  cuisine: 'Japanese',
  score: 9,
  address: { zipcode: '10300', street: 'Majestic' }
},
{
  _id: ObjectId("6751f5566a59c75535ff9948"),
  name: 'WOW Momos',
  town: 'Mallechwaram',
  cuisine: 'Indian',
  score: 5,
  address: { zipcode: '10400', street: 'Mallechwaram' }
}
]
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({}).sort({ name: -1 })
[
  {
    _id: ObjectId("6751f5566a59c75535ff9948"),
    name: 'WOW Momos',
    town: 'Mallechwaram',
    cuisine: 'Indian',
    score: 5,
    address: { zipcode: '10400', street: 'Mallechwaram' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9944"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian',
    score: 8,
    address: { zipcode: '10001', street: 'Jayanagar' }
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9947"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese',
    score: 9,
  }
]
```

- 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.Restaurant.find({ "grades.score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1,
```

```
restaurant_id: 1 }));
```

```
name: 'Empire',
town: 'MG Road',
cuisine: 'Indian',
score: 7,
address: { zipcode: '10100', street: 'MG Road' }
},
{
  _id: ObjectId("6751f5566a59c75535ff9946"),
  name: 'Chinese WOK',
  town: 'Indiranagar',
  cuisine: 'Chinese',
  score: 12,
  address: { zipcode: '20000', street: 'Indiranagar' }
}
]
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({ "score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1 })
[
  {
    _id: ObjectId("6751f5566a59c75535ff9944"),
    name: 'Meghna Foods',
    town: 'Jayanagar',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9945"),
    name: 'Empire',
    town: 'MG Road',
    cuisine: 'Indian'
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9947"),
    name: 'Kyotos',
    town: 'Majestic',
    cuisine: 'Japanese'
  },
  {
    _id: ObjectId("6751f5566a59c75535ff9948"),
    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" } } } ])
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } } ]
... ])
[
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Empire', average_score: 7 }
]
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
[
  { name: 'Meghna Foods', address: { street: 'Jayanagar' } },
  { name: 'Empire', address: { street: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
```

- 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 })
```

```
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } }
... ])
[
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Chinese WOK', average_score: 12 },
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'Empire', average_score: 7 }
]
Atlas atlas-10jjz6-shard-0 [primary] test> db.restaurants.find({ "address.zipcode": /^10/ }, { name: 1, "address.street": 1, _id: 0 })
[
  { name: 'Meghna Foods', address: { street: 'Jayanagar' } },
  { name: 'Empire', address: { street: 'MG Road' } },
  { name: 'Kyotos', address: { street: 'Majestic' } },
  { name: 'WOW Momos', address: { street: 'Malleshwaram' } }
]
```