VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

Nagaraja G (24BECS426)

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 **Nagaraja G (24BECS426),** 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 2024. 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.

Dr. Sheethala VA Assistant Professor Department of CSE, BMSCE	Dr. Joythi S Nayak Professor & HOD Department of CSE, BMSCE

Index

Insurance Database

Question

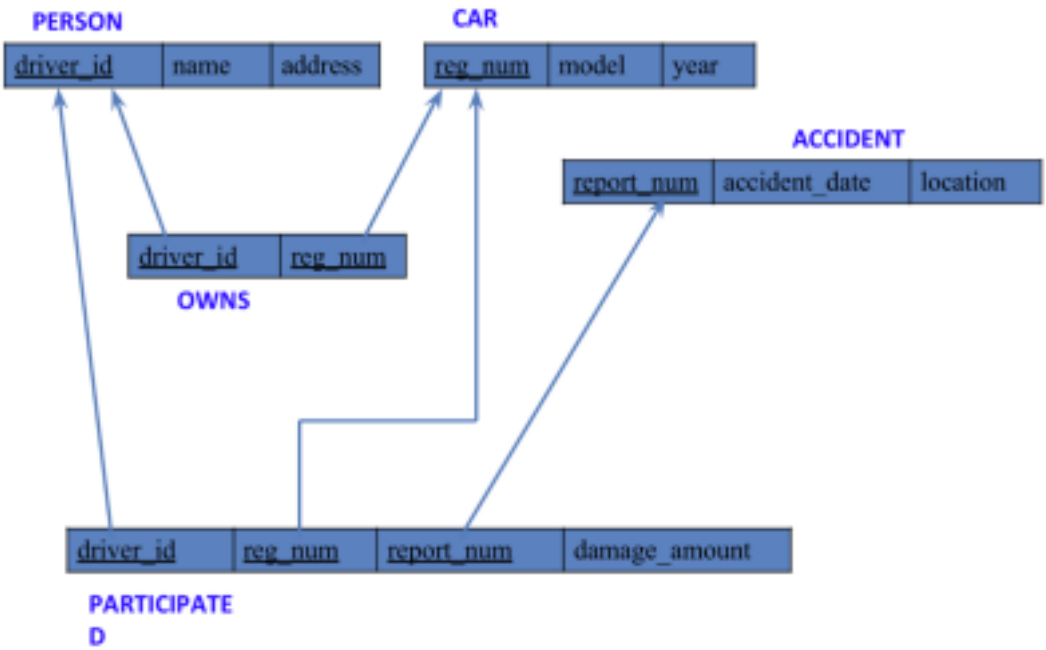
(Week 1)

- 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 '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

show databases;

create database insurance_426;

use insurance_426;

Create table

```
create table person(
driver_id varchar(20),
name varchar(20),
address varchar(20),
primary key(driver_id)
);
create table car(
reg_num varchar(20),
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(20),
reg_num varchar(10),
primary key(driver_id,reg_num),
foreign key(driver_id) references person(driver_id),
foreign key(reg_num) references car(reg_num)
);
create table participated(
driver_id varchar(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;

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

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

desc accident;

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

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

desc participated;

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

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

desc car;

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

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

desc owns;

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

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

Inserting Values to 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","NR Colony");

insert into person values("A05","Jhon","Hanumanth nagar");

select * from person;

Result Grid | Filter Rows:

driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	NR Colony
A05	Jhon	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;

Result Grid | Filter Rows:

reg_num	model	year
KA031181	Lancer	1957
KA041702	Audi	2005
KA052250	Indica	1990
KA053408	Honda	2008
KA095477	Toyota	1998
NULL	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,"2008-02-27","Mysore road");

insert into accident values (15,"2004-03-05","Kankapura");

select * from accident;

Result Grid | Filter Rows: | Edit

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	2008-02-27	Mysore road
15	2004-03-05	Kankapura
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;

Result Grid | Filter Rows:

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

Result Grid | Filter Rows: | Edit:

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

Queries

- **Display Accident date and location.**

select accident_date , location from accident;

Result Grid | Filter Rows:

accident_date	location
2003-01-01	Mysore Road
2004-02-02	South end Circle
2003-01-21	Bull temple Road
2008-02-27	Mysore road
2004-03-05	Kankapura

- **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;

Result Grid | Filter Rows: | Edit:

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

- **Add a new accident to the database.**

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

Result Grid | Filter Rows: | Edit:

report_num	accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Rc Bull temple Road
14	2008-02-27	Mysore road
15	2004-03-05	Kankapura
16	2008-03-08	Domlur
NULL	NULL	NULL

- **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;

Result Grid | Filter

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)

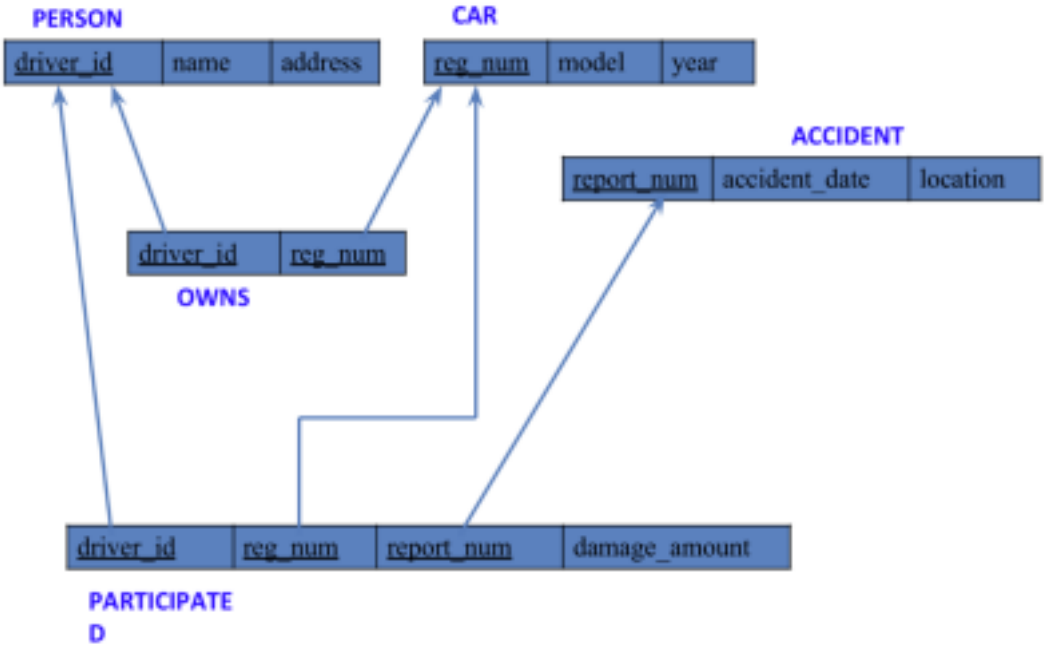
●PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)

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

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

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

Schema Diagram



Queries

● LIST THE ENTIRE PARTICIPATED RELATION IN THE DESCENDING ORDER OF DAMAGE AMOUNT.

SELECT * FROM participated ORDER BY damage_amount DESC;

Result Grid | Filter Rows: | Edit:

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

- **FIND THE AVERAGE DAMAGE AMOUNT.**

SELECT AVG(damage_amount) FROM participated;

Result Grid | Filter Rows:

AVG(damage_amount)
13600.0000

● FIND MAXIMUM DAMAGE AMOUNT.

SELECT MAX(damage_amount) FROM participated;

Result Grid | Filter Rows:

MAX(damage_amount)
25000

- **DELETE THE RECORDS WHICH IS LESS THAN AVERAGE DAMAGE AMOUNT.**

DELETE FROM participated WHERE DAMAGE_AMOUNT < (SELECT AVG(DAMAGE_AMOUNT) FROM (SELECT DAMAGE_AMOUNT FROM participated) as sub);

Result Grid | Filter Rows: | Edit:

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

Bank Database

Question
(Week 3)

- branch_426 (branch_name: String, branch_city: String, assets: real)

- bank_account_426 (accno: int, branch_name: String, balance: real)

- bank_customer_426 (customer_name: String, customer_street: String, customer_city: String)

- depositor_426 (customer_name: String, accno: int)

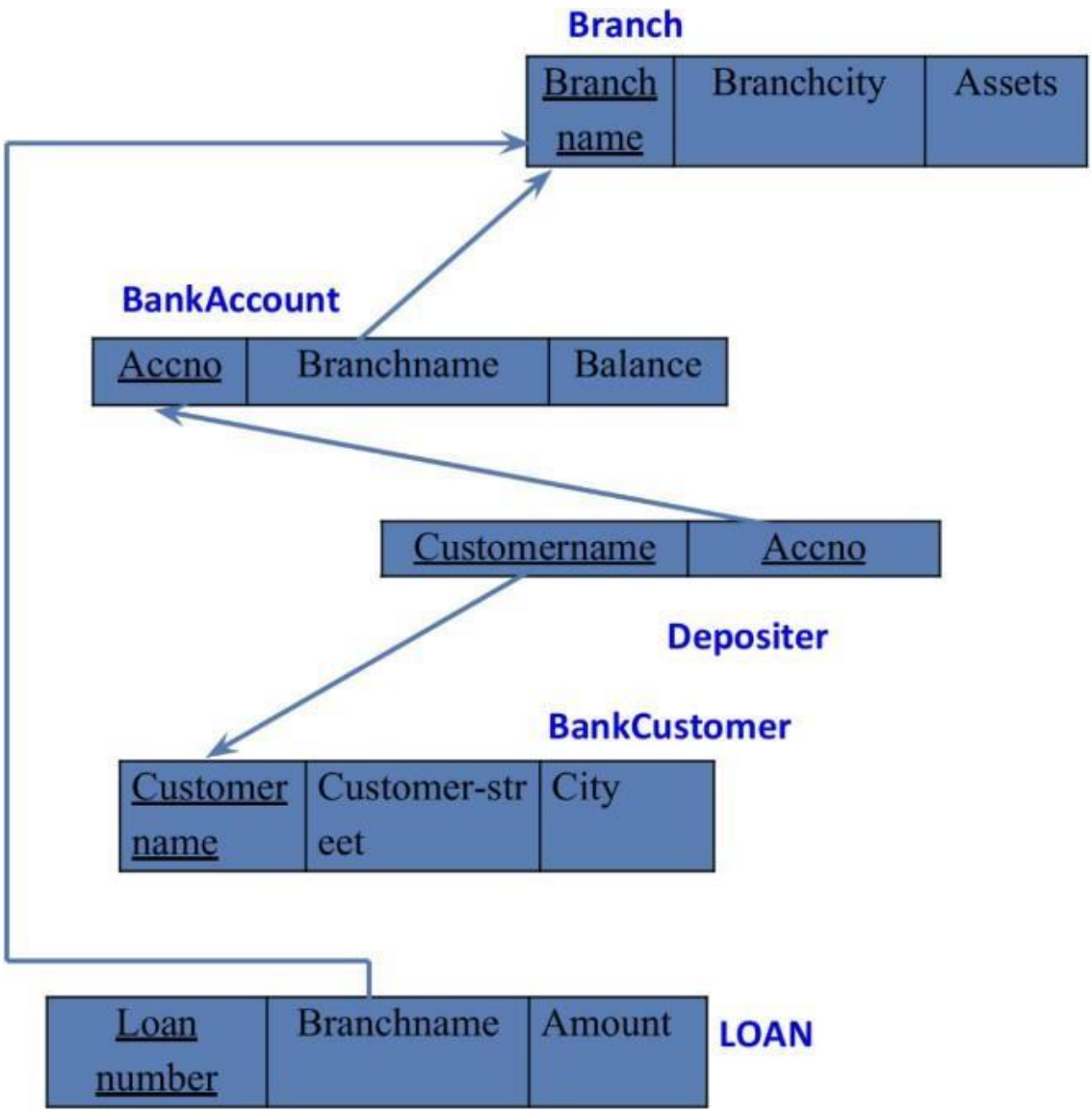
- loan (loan_number: int, branch_name: String, amount: real)

- 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 branchs 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_426 the sum of the amount of all the loans at the branch.

Schema Diagram



Create database

```
create database bank_426;
use  bank_426;
```

Create table

```
create table branch_426(
branch_name varchar(25),
branch_city varchar(25),
assets int,
primary key(branch_name));
```

```
create table bank_account_426(
acc_no int,
balance int,
branch_name varchar(25),
primary key(acc_no),
foreign key(branch_name) references branch_426(branch_name)
);
create table bank_customer_426(
customer_name varchar(10),
customer_street varchar(25),
customer_city varchar(25),
primary key (customer_name)
);
create table depositor_426(
customer_name varchar(10),
acc_no int,
primary key (customer_name,acc_no),
foreign key(customer_name) references bank_customer_426(customer_name),
foreign key(acc_no) references bank_account_426(acc_no)
);
create table loan(
loan_no int,
branch_name varchar(25),
amount int,
primary key (loan_no),
foreign key(branch_name) references branch_426(branch_name)
);
```

Inserting Values to the tables

```
insert into branch_426 values("SBI_Chamrajpet","Bangalore",50000);
insert into branch_426 values("SBI_Residencyroad","Bangalore",10000);
insert into branch_426 values("SBI_Shivajiroad","Bombay",20000);
insert into branch_426 values("SBI_Parlimentroad","Delhi",10000);
insert into branch_426 values("SBI_Jantarmantar","Delhi",20000);

select * from branch_426;
```

Result Grid | Filter Rows: | Edit:

branch_name	branch_city	assets_inlakhs
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_MantriMarg	Delhi	200000
SBI_Parlimentroad	Delhi	10000
SBI_Residencyroad	Bangalore	10000
SBI_Shivajiroad	Bombay	20000
NULL	NULL	NULL

```
insert into bank_account_426 values (1,2000,"SBI_Chamrajpet");
insert into bank_account_426 values (2,5000,"SBI_Residencyroad");
insert into bank_account_426 values (3,6000,"SBI_Shivajiroad");
insert into bank_account_426 values (4,9000,"SBI_Parlimentroad");
insert into bank_account_426 values (5,8000,"SBI_Jantarmantar");
insert into bank_account_426 values (6,4000,"SBI_Shivajiroad");
insert into bank_account_426 values (8,4000,"SBI_Residencyroad");
insert into bank_account_426 values (9,3000,"SBI_Parlimentroad");
insert into bank_account_426 values (10,5000,"SBI_Residencyroad");
insert into bank_account_426 values (11,2000,"SBI_Jantarmantar");
```

```
select * from bank_account_426;
```

Result Grid | Filter Rows:

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

bank_account_426 4 ×

```
insert into bank_customer_426 values("Avinash","Bull_temple_road","Bangalore");
insert into bank_customer_426 values("Dinesh","Bannergatta_road","Bangalore");
insert into bank_customer_426 values("Mohan","NationalCollege_road","Bangalore");
insert into bank_customer_426 values("Nikil","Akabar_road","Delhi");
insert into bank_customer_426 values("Ravi","Prithviraj_road","Delhi");
```

```
select * from bank_customer_426;
```

Result Grid | Filter Rows: | Edit:

customer_name	customer_street	customer_city
Avinash	Bull_temple_road	Bangalore
Dinesh	Bannergatta_road	Bangalore
Mohan	NationalCollege_road	Bangalore
Nikil	Akabar_road	Delhi
Ravi	Prithviraj_road	Delhi
NULL	NULL	NULL

```
insert into depositor_426 values("Avinash",1);
insert into depositor_426 values("Dinesh",2);
insert into depositor_426 values("Nikil",4);
insert into depositor_426 values("Ravi",5);
insert into depositor_426 values("Avinash",8);
insert into depositor_426 values("Nikil",9);
insert into depositor_426 values("Dinesh",10);
insert into depositor_426 values("Nikil",11);

select * from depositor_426;
```

Result Grid | Filter Rows:

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

depositor_426 7

```
insert into loan values(1,"SBI_Chamrajpet",1000);
insert into loan values(2,"SBI_Residencyroad",2000);
insert into loan values(3,"SBI_Shivajiroad",3000);
insert into loan values(4,"SBI_Parlimentroad",4000);
insert into loan values(5,"SBI_Jantarmantar",5000);

select * from loan;
```

Result Grid | Filter Rows:

loan_no	branch_name	amount
1	SBI_Chamrajpet	1000
2	SBI_Residencyroad	2000
3	SBI_Shivajiroad	3000
4	SBI_Parlimentroad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

Queries

- **Display the branch_name and assets from all branchs in lakhs of rupees and rename the assets column to 'assets in lakhs'.**

alter table branch_426 change assets

assets_inlakhs real; select branch_name,

assets_inlakhs from branch_426;

branch_name	assets_inlakhs
SBI_chamrajpet	5000
SBI_jantarmantar	20000
SBI_parlimentRoad	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_426 d, bank_account_426 b where b.branch_name='SBI_ResidencyRoad' and d.acc_no=b.acc_no group by d.customer_name having
count(d.acc_no)>=2;

Result Grid | Filter Rc

customer_name
Dinesh

- **Create a view which gives each branch_426 the sum of the amount of all the loans at the branch.**

```
create view br1 as
select branch_name, sum(amount)
from loan
group by branch_name;
select * from br1;
```

Result Grid | Filter Rows:

branch_name	sum(amount)
SBI_Chamrajpet	1000
SBI_Jantarmantar	5000
SBI_Parlimentroad	4000
SBI_Residencyroad	2000
SBI_Shivajiroad	3000

More Queries on Bank Database

Question

(Week 4)

- branch_426 (branch_name: String, branch_city: String, assets: real)

- bank_account_426 (accno: int, branch_name: String, balance: real)

- bank_customer_426 (customer_name: String, customer_street: String, customer_city: String)

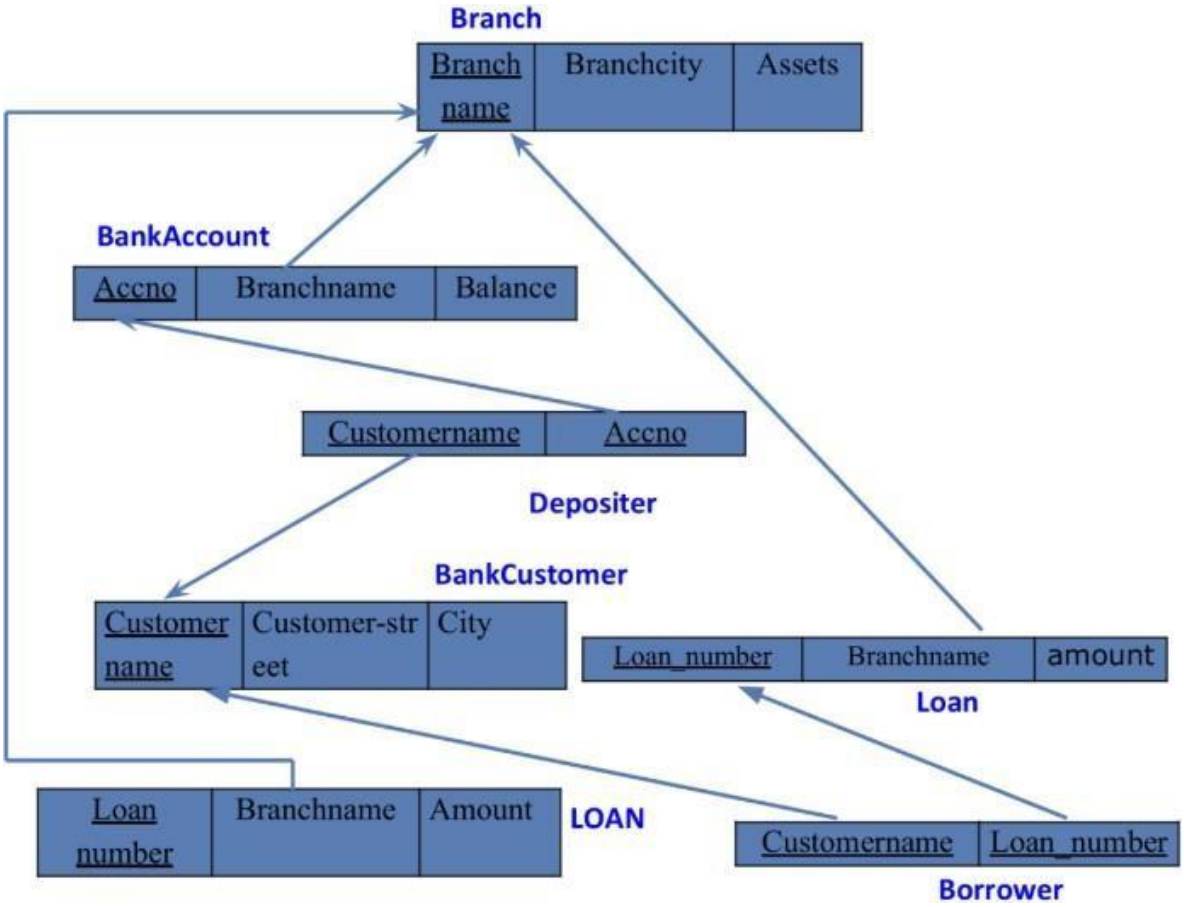
- depositor_426 (customer_name: String, accno: int)

- loan (loan_number: int, branch_name: String, amount: real)

-borrower_426(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 branchs that have greater assets than all branchs 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

```
use  bank_426;

create table borrower_426(
customer_name varchar(10),
loan_no int,
primary key(customer_name,loan_no),
foreign key(customer_name) references bank_customer_426(customer_name),
foreign key(loan_no) references loan(loan_no)
);
```

Inserting values to the table

```
insert into borrower_426 values("Avinash",1);
insert into borrower_426 values("Dinesh",2);
insert into borrower_426 values("Mohan",3);
insert into borrower_426 values("Nikil",4);
insert into borrower_426 values("Ravi",5);
select * from borrower_426;
```

Result Grid | Filter Rows:

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

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_426 d, bank_account_426 ba,
branch_426 b
where d.acc_no=ba.acc_no and
ba.branch_name=b.branch_name
and b.branch_city="Delhi"
group by d.customer_name having
count(b.branch_name)>1;
```

Result Grid | Filter Rc

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_426 b
```

where b.loan_no not in(select d.acc_no from depositor_426 d
where b.loan_no=d.acc_no);

Result Grid | Filter

customer_name
Mohan

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

select b.customer_name
from borrower_426 b
where b.loan_no in(select d.acc_no from depositor_426 d,bank_account_426 ba, branch_426 b
where b.loan_no=d.acc_no and d.acc_no=ba.acc_no and
ba.branch_name=b.branch_name
and b.branch_city="Bangalore");

Result Grid | Filte

customer_name
Avinash
Dinesh

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

select branch_name
from branch_426
where assets_inlakhs>all(select assets_inlakhs from branch_426
where branch_city="Bangalore");

Result Grid | Fi

branch_name
SBI_MantriMarg
NULL

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

delete from bank_account_426 ba
where ba.branch_name=(select b.branch_name from branch_426 b where branch_city="Bombay");

Result Grid | Filter Rows:

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

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

update bank_account_426
set balance=balance+((5*balance)/100) where acc_no in(1,2,3,4,5,6,8,9,10,11,12);
select * from bank_account_426;

Result Grid | Filter Rows:

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

Employee Database

Question

(Week 5)

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

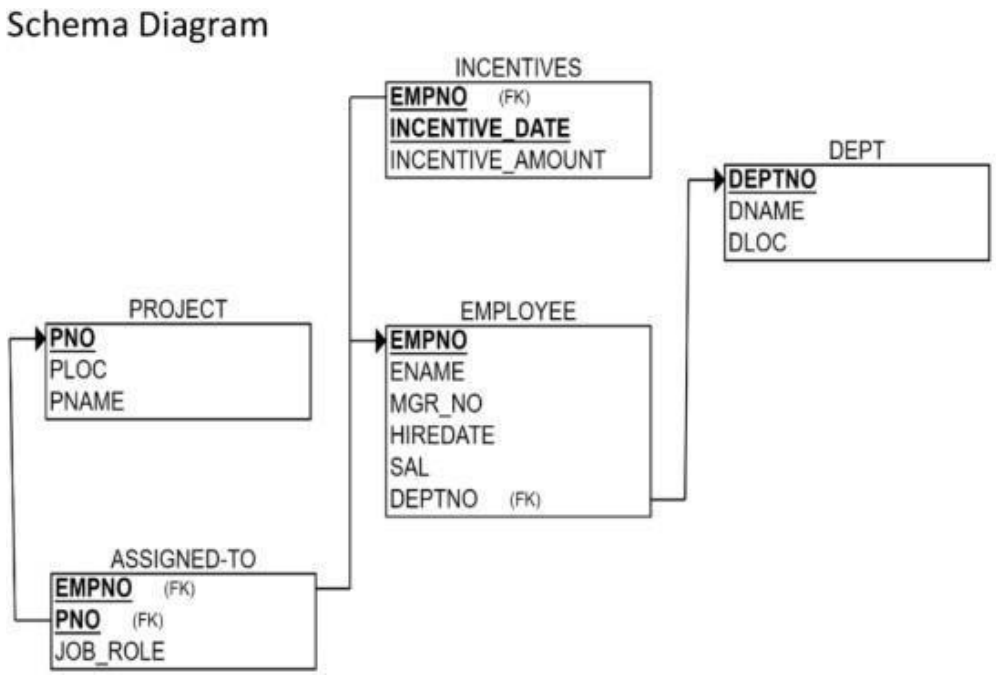
2.Enter greater than five tuples for each table.

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

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

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

Schema Diagram



Create database

```
create database employee_24becs426;
use employee_24becs426;
```

Create table

```
create table dept_426(
d_no int,
d_name varchar(20),
d_loc varchar(20),
primary key(d_no)
);

create table employee_426(
e_no int,
e_name varchar(20),
mngr_no int,
hire_date varchar(10),
salary float,
d_no int,
primary key(e_no,d_no),
```

```
foreign key(d_no) references dept_426(d_no)
 );

create table incentives_426(
 e_no int,
 incentive_date varchar(20),
 incentive_amt float,
 primary key(e_no,incentive_date),
 foreign key(e_no) references employee_426(e_no)
 );

 create table project_426(
 p_no int,
 p_loc varchar(20),
 p_name varchar(20),
 primary key(p_no)
 );

create table assigned_to_426(
 e_no int,
 p_no int,
 jobe_role text,
 primary key(e_no,p_no),
 foreign key(e_no) references employee_426(e_no),
 foreign key(p_no) references project_426(p_no)
 );
```

Inserting Values to the table

```
insert into dept_426 values(1,"cse","pj");
 insert into dept_426 values(2,"ise","pj");
 insert into dept_426 values(3,"csds","pg");
 insert into dept_426 values(4,"ece","pg");
 insert into dept_426 values(5,"aiml","pj");
 select * from dept_426;
```

Result Grid | Filter Rows:

d_no	d_name	d_loc
1	cse	pj
2	ise	pj
3	csds	pg
4	ece	pg
5	aiml	pj
NULL	NULL	NULL

```
insert into employee_426 values(101,"mahesh",100,"12/01/1999",100000,1);
 insert into employee_426 values(201,"kushal",200,"17/01/2020",50000,2);
 insert into employee_426 values(301,"naga",100,"01/09/2004",30000,3);
 insert into employee_426 values(401,"chaithanya",101,"03/08/2000",10000,4);
 insert into employee_426 values(501,"aman",101,"19/02/2008",90000,5);
 select * from employee_426;
```

Result Grid | Filter Rows: | Edit: | Expor

e_no	e_name	mngr_no	hire_date	salary	d_no
101	mahesh	100	12/01/1999	100000	1
201	kushal	200	17/01/2020	50000	2
301	naga	100	01/09/2004	30000	3
401	chaithanya	101	03/08/2000	10000	4
501	aman	101	19/02/2008	90000	5
NULL	NULL	NULL	NULL	NULL	NULL

```
insert into incentives_426 values(101,"12/03/2004",50000);
insert into incentives_426 values(201,"17/03/2024",25000);
insert into incentives_426 values(301,"01/12/2019",15000);
insert into incentives_426 values(401,"03/11/2019",5000);
insert into incentives_426 values(501,"29/4/2019",45000);

select * from incentives_426;
```

Result Grid | Filter Rows:

e_no	incentive_date	incentive_amt
101	12/03/2004	50000
201	17/03/2024	25000
301	01/12/2019	NULL
401	03/11/2019	5000
501	29/4/2019	45000
NULL	NULL	NULL

```
insert into project_426 values(10,"bangalore","chatbot");
insert into project_426 values(40,"delhi","ml model");
insert into project_426 values(50,"bombay","blockchain");
insert into project_426 values(30,"chennai","stocks");
insert into project_426 values(80,"mysore","android app");
select * from project_426;
```

Result Grid | Filter Rows:

p_no	p_loc	p_name
10	bangalore	chatbot
30	chennai	stocks
40	delhi	ml model
50	bombay	blockchain
80	mysore	android app
NULL	NULL	NULL

```
insert into assigned_to_426 values(101,10,"devops");
insert into assigned_to_426 values(201,40,"sde");
insert into assigned_to_426 values(301,50,"manager");
insert into assigned_to_426 values(401,30,"jpa");
insert into assigned_to_426 values(501,80,"pa");
select * from assigned_to_426;
```

Result Grid | Filter Rows:

e_no	p_no	jobe_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 e_no
from assigned_to_426 a,project_426 p
where a.p_no=p.p_no and p.p_loc in ("bangalore","mysure","Hyderabad");
```

Result Grid | Filter Rows:

e_no
101

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

```
update incentives_426 i
set i.incentive_amt=null
where i.e_no=301;
select e_no
from incentives_426
where incentive_amt is null;
```

Result Grid | Filter Rows:

e_no
301

- **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_name,e.e_no,d.d_name,a.jobe_role,d.d_loc,p.p_loc
from employee_426 e,dept_426 d,project_426 p,assigned_to_426 a
where e.e_no=a.e_no and e.d_no=d.d_no and p.p_no=a.p_no and p.p_loc=d.d_loc;
```

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

	e_name	e_no	d_name	jobe_role	d_loc	p_loc

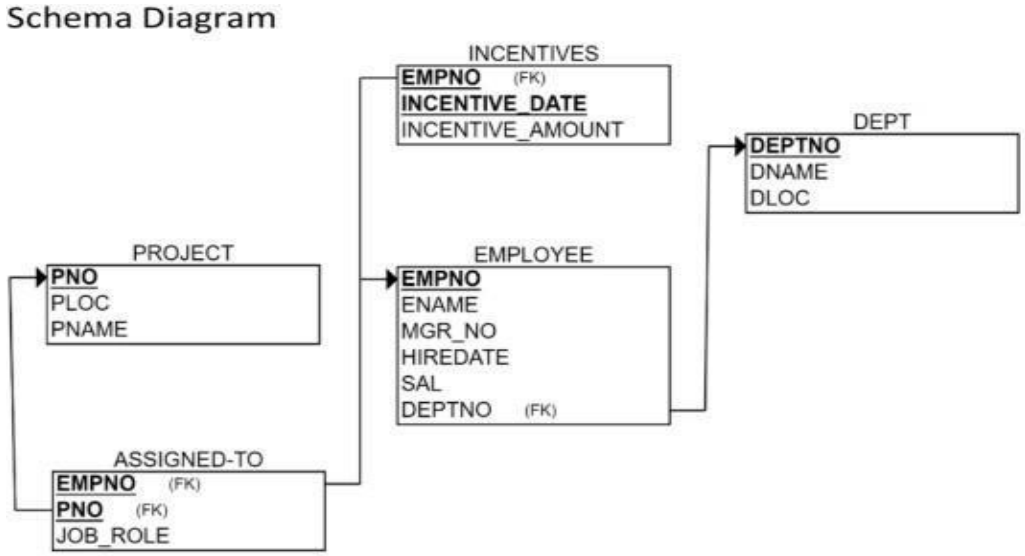
More Queries on Employee Database

Question

(Week 6)

1.Using Scheme diagram, Create tables by properly specifying the primary keys and the foreign keys.
2.Enter greater than five tuples for each table.
3.List the name of the managers with the maximum employees.
4.Display those managers name whose salary is more than average salary of his employee.
5.Find the name of the second top level managers of each department.
6.Find the employee details who got the second maximum incentive in January 2019.
7. Display those employees who are working in the same department where his the manager is working.

Schema Diagram



Queries

- **List the name of the managers with the maximum employees.**
  select e_name
  from employee_426
  where mngr_no=(select MAX(mngr_no) from employee_426);

Result Grid | Filter Rows:

e_name
kushal

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

```
select e_name
from employee_426
where salary > (select AVG(salary) from employee_426);
```

Result Grid | Filter Rows:

e_name
mahesh
aman
akash
santhosh

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

```
select e_name
from employee_426
where salary=(select MAX(salary) from employee_426
        where salary < (select MAX(salary) from employee_426));
```

Result Grid | Filter Rows:

e_name
aman
santhosh

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

```
select * from employee_426
where e_no=(select e_no from incentives_426 where incentive_amt=(select
MAX(incentive_amt) from incentives_426 where incentive_amt < (select
MAX(incentive_amt) from incentives_426)) );
```

Result Grid | Filter Rows: | Edit:

e_no	e_name	mngr_no	hire_date	salary	d_no
501	aman	101	19/02/2008	90000	5
NULL	NULL	NULL	NULL	NULL	NULL

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

```
select * from employee_426 e
where e.d_no=(select e1.d_no from employee_426 e1
                                  where e.mngr_no=e1.e_no);
```

Result Grid | Filter Rows: | Edit:

e_no	e_name	mngr_no	hire_date	salary	d_no
601	akash	101	29/12/2004	80000	1
NULL	NULL	NULL	NULL	NULL	NULL

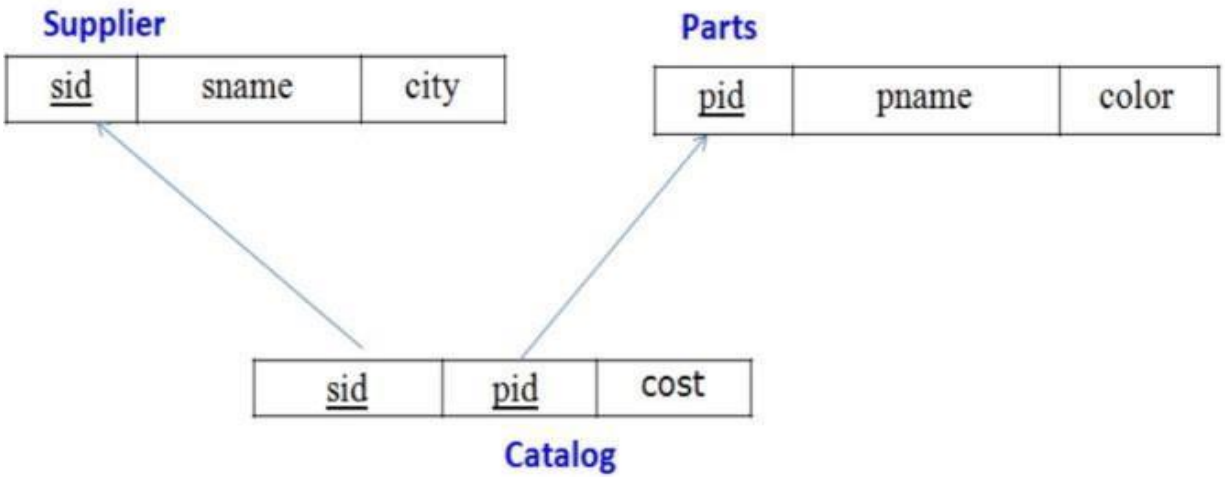
Supplier Database

Question

(Week 7)

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

Schema Diagram



Create database

```
create database Supplier_426;
use Supplier_426;
```

Create table

```
create table supplierT_426(
s_id int,
```

```
s_name varchar(20),
city varchar(20),
primary key(s_id)
);

create table parts_426(
p_id int,
p_name varchar(20),
color varchar(20),
primary key(p_id)
);

create table catalog_426(
s_id int,
p_id int,
cost int,
primary key(s_id,p_id),
foreign key(s_id) references supplierT_426(s_id),
foreign key(p_id) references parts_426(p_id)
);
```

Inserting Values to the table

```
insert into supplierT_426 values(10001,"Acme widget","Bangalore");
insert into supplierT_426 values(10002,"Johns","Kolkata");
insert into supplierT_426 values(10003,"Vimal","Mumbai");
insert into supplierT_426 values(10004,"Reliance","Delhi");
select * from supplierT_426;

insert into parts_426 values(20001,"Book","Red");
insert into parts_426 values(20002,"Pen","Red");
insert into parts_426 values(20003,"Pencil","Green");
insert into parts_426 values(20004,"Mobile","Green");
insert into parts_426 values(20005,"Charger","Black");
select * from parts_426;

insert into catalog_426 values(10001,20001,10);
insert into catalog_426 values(10001,20002,10);
insert into catalog_426 values(10001,20003,30);
insert into catalog_426 values(10001,20004,10);
insert into catalog_426 values(10001,20005,10);
insert into catalog_426 values(10002,20001,10);
```

```
insert into catalog_426 values(10002,20002,20);
insert into catalog_426 values(10003,20003,30);
insert into catalog_426 values(10004,20003,40);
select * from catalog_426;
```

Queries

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

```
select distinct p.p_name
from parts_426 p,catalog_426 c
where p.p_id=c.p_id;
```

Result Grid | Filter Rows

p_name
Book
Pen
Pencil
Mobile
Charger

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

```
select distinct s.s_name
from catalog_426 c,supplierT_426 s
where c.s_id=s.s_id and NOT EXISTS(select p.p_id from parts_426 p where not
exists(select c1.s_id from catalog_426 c1
where c1.s_id=c.s_id and c1.p_id=p.p_id));
```

Result Grid | Filter Rows

s_name
Acme widget

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

```
select distinct s.s_name
from catalog_426 c,supplierT_426 s
where c.s_id=s.s_id and not exists(select p.p_id  from parts_426 p where p.color="Red"
 and not exists(select c1.s_id from catalog_426 c1
  where c1.s_id=c.s_id and c1.p_id=p.p_id and p.color="Red"));
```

Result Grid | Filter

s_name
Acme widget
Johns

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

```
select p.p_name
from parts_426 p,catalog_426 c,supplierT_426 s
where p.p_id=c.p_id and c.s_id=s.s_id
and s.s_name="Acme widget" and not exists(select * from catalog_426 c1,supplierT_426 s1 where p.p_id=c.p_id and c1.s_id=s1.s_id and s1.s_name!="Acme widget");
```

Result Grid | Filter

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).**

```
select distinct c.s_id
from catalog_426 c
where c.cost>(select AVG(c1.cost) from catalog_426 c1
                                where c1.p_id=c.p_id);
```

Result Grid | Filte

s_id
10002
10004

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

```
select p.p_id,s.s_name
from parts_426 p,catalog_426 c,supplierT_426 s
where c.p_id=p.p_id and c.s_id=s.s_id
and c.cost=(select MAX(c1.cost)
                              from catalog_426 c1 where c1.p_id=p.p_id);
```

Result Grid | Filter Rows:

p_id	s_name
20001	Acme widget
20004	Acme widget
20005	Acme widget
20001	Johns
20002	Johns
20003	Reliance

NoSQL Lab -1

Question
(Week 8)

Perform the following DB operations using MongoDB:

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

2.Insert appropriate values

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

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

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,a ge: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();

```
mongosh mongodb+srv://<credentials>@cluster0.muzcl.mongodb.net/
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\STUDENT>mongosh "mongodb+srv://cluster0.muzcl.mongodb.net/" --apiVersion 1 --username NagaDBMS1400 --password BmsceDBMS
Current Mongosh Log ID: 6746b7cfd38cdb5babe3a2fb
Connecting to:          mongodb+srv://<credentials>@cluster0.muzcl.mongodb.net/?appName=mongosh+2.0.0
Using MongoDB:          8.0.3 (API Version 1)
Using Mongosh:          2.0.0
mongosh 2.2.3 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-575vpw-shard-0 [primary] test> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-575vpw-shard-0 [primary] test> show dbs
test     8.00 KiB
admin  328.00 KiB
local   21.00 GiB
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.insert({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746b93cd38cdb5babe3a2fc") }
}
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746b993d38cdb5babe3a2fd") }
}
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746b9c3d38cdb5babe3a2fe") }
}
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746b9e7d38cdb5babe3a2ff") }
}
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId("6746ba0ed38cdb5babe3a300") }
```

```
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.find();
[
  {
    _id: ObjectId("6746b93cd38cdb5babe3a2fc"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b993d38cdb5babe3a2fd"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9c3d38cdb5babe3a2fe"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9e7d38cdb5babe3a2ff"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746ba0ed38cdb5babe3a300"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

Queries

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

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

```
Atlas atlas-575vpw-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-575vpw-shard-0 [primary] test> db.Student.find();
[
  {
    _id: ObjectId("6746b93cd38cdb5babe3a2fc"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b993d38cdb5babe3a2fd"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9c3d38cdb5babe3a2fe"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9e7d38cdb5babe3a2ff"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746ba0ed38cdb5babe3a300"),
    RollNo: 10,
    Age: 23,
    Cont: 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.co m”});

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

```
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.update({RollNo:12,Name:"ABC"},{$set:{Name:"FEM"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
Atlas atlas-575vpw-shard-0 [primary] test> db.Student.find();
[
  {
    _id: ObjectId("6746b93cd38cdb5babe3a2fc"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b993d38cdb5babe3a2fd"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9c3d38cdb5babe3a2fe"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("6746b9e7d38cdb5babe3a2ff"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("6746ba0ed38cdb5babe3a300"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  },
  {
    _id: ObjectId("6746bb33d38cdb5babe3a301"),
    RollNo: 11,
    Age: 22,
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  },
  {
    _id: ObjectId("6746bc28d38cdb5babe3a302"),
    RollNo: 12,
    Name: 'FEM',
    Age: 22,
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  }
]
```

NoSQL Lab -2

Question
(Week 9)

Perform the following DB operations using MongoDB.

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

2.Insert at least 5 values into the table

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

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

5.Export the created collection into local file system

6.Drop the table

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

Create Table

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

Inserting Values to the table

```
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"}]);
```

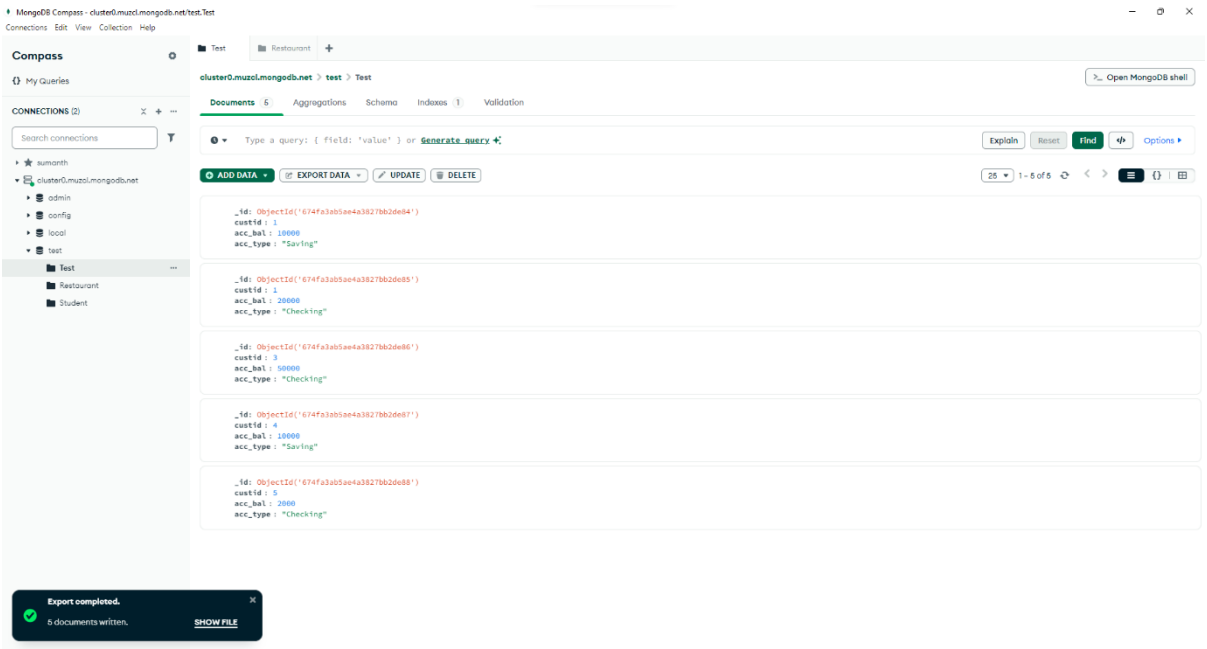
```
mongosh mongodb+srv://<credentials>@cluster0.muzcl.mongodb.net/
Microsoft Windows [Version 10.0.22000.2538]
(c) Microsoft Corporation. All rights reserved.

C:\Users\STUDENT>mongosh "mongodb+srv://cluster0.muzcl.mongodb.net/" --apiVersion 1 --username NagaDBMS1400
Enter password: ********
Current Mongosh Log ID: 674fa2f3416a7f56466733da
Connecting to:          mongodb+srv://<credentials>@cluster0.muzcl.mongodb.net/?appName=mongosh+2.0.0
MongoServerError: bad auth : authentication failed

C:\Users\STUDENT>mongosh "mongodb+srv://cluster0.muzcl.mongodb.net/" --apiVersion 1 --username NagaDBMS1400
Enter password: ************
Current Mongosh Log ID: 674fa31cc7e070dbfba45461
Connecting to:          mongodb+srv://<credentials>@cluster0.muzcl.mongodb.net/?appName=mongosh+2.0.0
MongoServerError: bad auth : authentication failed

C:\Users\STUDENT>mongosh "mongodb+srv://cluster0.muzcl.mongodb.net/" --apiVersion 1 --username NagaDBMS1400
Enter password: ************
Current Mongosh Log ID: 674fa3525ae4a3827bb2de83
Connecting to:          mongodb+srv://<credentials>@cluster0.muzcl.mongodb.net/?appName=mongosh+2.0.0
Using MongoDB:          8.0.3 (API Version 1)
Using Mongosh:          2.0.0
mongosh 2.3.3 is available for download: https://www.mongodb.com/try/download/shell

For mongosh info see: https://docs.mongodb.com/mongodb-shell/

Atlas atlas-575vpw-shard-0 [primary] test> db.createCollection("Customer");
{ ok: 1 }
Atlas atlas-575vpw-shard-0 [primary] test> 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"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId("674fa3ab5ae4a3827bb2de84"),
    '1': ObjectId("674fa3ab5ae4a3827bb2de85"),
    '2': ObjectId("674fa3ab5ae4a3827bb2de86"),
    '3': ObjectId("674fa3ab5ae4a3827bb2de87"),
    '4': ObjectId("674fa3ab5ae4a3827bb2de88")
  }
}
```

Queries

1.Write a query to display those records whose total account balance is greater than 1200 of account type 'Checking' for each customer_id.
2.Determine Minimum and Maximum account balance for each customer_id.(For both 1 & 2 Op below)

```
Atlas atlas-575vpw-shard-0 [primary] test> db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});
[
  {
    _id: ObjectId("674fa3ab5ae4a3827bb2de85"),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId("674fa3ab5ae4a3827bb2de86"),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  }
]
Atlas atlas-575vpw-shard-0 [primary] test> db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal:
... {$max:"$acc_bal"}}}]);
[
  { _id: 5, minBal: 2000, maxBal: 2000 },
  { _id: 4, minBal: 10000, maxBal: 10000 },
  { _id: 3, minBal: 50000, maxBal: 50000 },
  { _id: 1, minBal: 10000, maxBal: 20000 }
]
Atlas atlas-575vpw-shard-0 [primary] test>
```

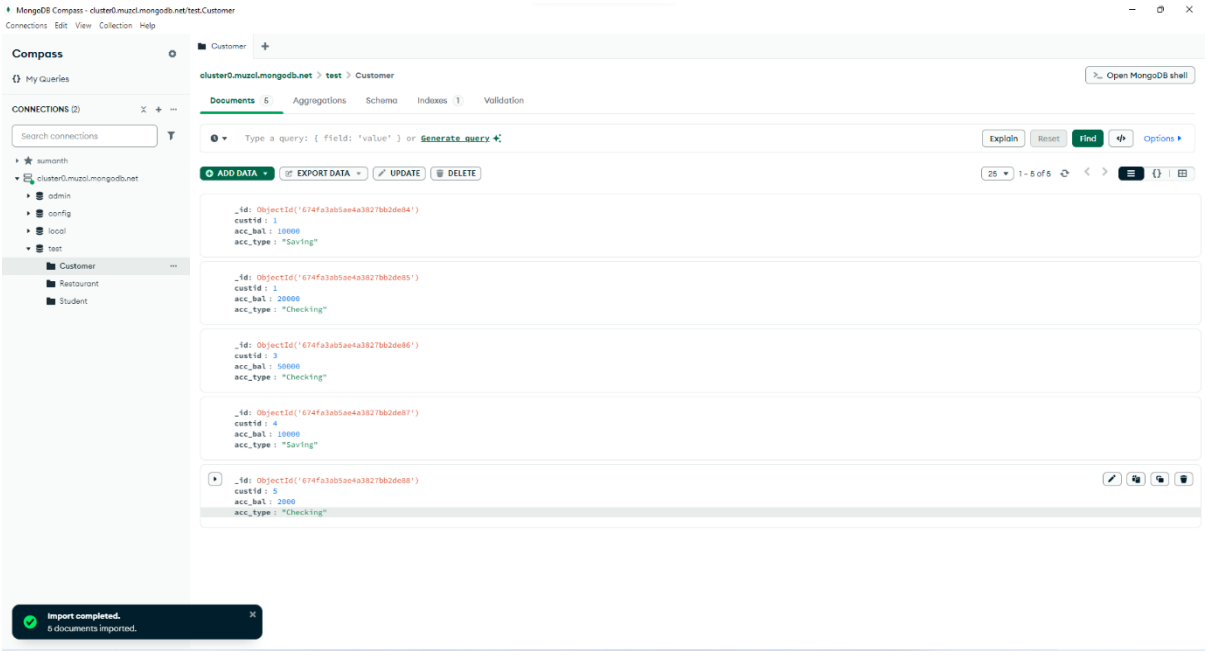
- **Export the created collection into local file system.(In GUI)**



- **Drop the table.**

  db.Customer.drop();

- **Import a given csv dataset from local file system into mongodb collection.(In GUI)**



NoSQL Lab -3

Question
(Week 10)

1.Write a MongoDB query to display all the documents in the collection restaurants.
2.Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
3.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.
4.Write a MongoDB query to find the average score for each restaurant.
5.Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

Create Table

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

Inserting Values to the table

```
db.Restaurants.insertMany([
   { _id: "675000fe0fc96cbaf4e17320", name: "WOW Momos", town: "Malleshwaram",
cuisine: "Indian", score: 5, address: { zipcode: "10400", street: "Malleshwaram" } },
   { _id: "675000fe0fc96cbaf4e1731c", name: "Meghna Foods", town: "Jayanagar",
cuisine: "Indian", score: 8, address: { zipcode: "10001", street: "Jayanagar" } },
   { _id: "675000fe0fc96cbaf4e1731f", name: "Kyotos", town: "Majestic", cuisine:
"Japanese", score: 9, address: { zipcode: "10300", street: "Majestic" } },
   { _id: "675000fe0fc96cbaf4e1731d", name: "Empire", town: "MG Road", cuisine:
"Indian", score: 7, address: { zipcode: "10100", street: "MG Road" } },
   { _id: "675000fe0fc96cbaf4e1731e", name: "Chinese WOK", town: "Indiranagar",
cuisine: "Chinese", score: 12, address: { zipcode: "20000", street: "Indiranagar" } }
]);
```

Queries

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

```
{
  _id: ObjectId("675000fe0fc96cbaf4e17320"),
  name: 'WOW Momos',
  town: 'Malleshwaram',
  cuisine: 'Indian',
  score: 5,
  address: { zipcode: '10400', street: 'Malleshwaram' }
},
{
  _id: ObjectId("675000fe0fc96cbaf4e1731c"),
  name: 'Meghna Foods',
  town: 'Jayanagar',
  cuisine: 'Indian',
  score: 8,
  address: { zipcode: '10001', street: 'Jayanagar' }
},
{
  _id: ObjectId("675000fe0fc96cbaf4e1731f"),
  name: 'Kyotos',
  town: 'Majestic',
  cuisine: 'Japanese',
  score: 9,
  address: { zipcode: '10300', street: 'Majestic' }
},
{
  _id: ObjectId("675000fe0fc96cbaf4e1731d"),
  name: 'Empire',
  town: 'MG Road',
  cuisine: 'Indian',
  score: 7,
  address: { zipcode: '10100', street: 'MG Road' }
},
{
  _id: ObjectId("675000fe0fc96cbaf4e1731e"),
  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 along with all the columns.**

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

```
[
  {
    "_id": "675000fe0fc96cbaf4e1731e",
    "name": "WOW Momos",
    "town": "Malleshwaram",
    "cuisine": "Indian",
    "score": 5,
    "address": {
      "zipcode": "10400",
      "street": "Malleshwaram"
    }
  },
  {
    "_id": "675000fe0fc96cbaf4e1731c",
    "name": "Meghna Foods",
    "town": "Jayanagar",
    "cuisine": "Indian",
    "score": 8,
    "address": {
      "zipcode": "10001",
      "street": "Jayanagar"
    }
  },
  {
    "_id": "675000fe0fc96cbaf4e1731f",
    "name": "Kyotos",
    "town": "Majestic",
    "cuisine": "Japanese",
    "score": 9,
    "address": {
```

```
    "cuisine": "Japanese",
    "score": 9,
    "address": {
      "zipcode": "10300",
      "street": "Majestic"
    }
  },
  {
    "_id": "675000fe0fc96cbaf4e1731d",
    "name": "Empire",
    "town": "MG Road",
    "cuisine": "Indian",
    "score": 7,
    "address": {
      "zipcode": "10100",
      "street": "MG Road"
    }
  },
  {
    "_id": "675000fe0fc96cbaf4e1731e",
    "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": "675000fe0fc96cbaf4e17320",
    "name": "WOW Momos",
    "town": "Malleshwaram",
    "cuisine": "Indian"
  },
  {
    "_id": "675000fe0fc96cbaf4e1731c",
    "name": "Meghna Foods",
    "town": "Jayanagar",
    "cuisine": "Indian"
  },
  {
    "_id": "675000fe0fc96cbaf4e1731f",
    "name": "Kyotos",
    "town": "Majestic",
    "cuisine": "Japanese"
  },
  {
    "_id": "675000fe0fc96cbaf4e1731d",
    "name": "Empire",
    "town": "MG Road",
    "cuisine": "Indian"
  }
```

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

```
Atlas atlas-6r0t5d-shard-0 [primary] test> db.restaurants.aggregate([ { $group: { _id: "$name", average_score: { $avg: "$score" } } } ])
[
  { _id: 'Kyotos', average_score: 9 },
  { _id: 'WOW Momos', average_score: 5 },
  { _id: 'Empire', average_score: 7 },
  { _id: 'Meghna Foods', average_score: 8 },
  { _id: 'Chinese WOK', average_score: 12 }
]
Atlas atlas-6r0t5d-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' } }
]
Atlas atlas-6r0t5d-shard-0 [primary] test>
```

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

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

```
[
  {
    "name": "WOW Momos",
    "address": {
      "zipcode": "10400",
      "street": "Malleshwaram"
    }
  },
  {
    "name": "Meghna Foods",
    "address": {
      "zipcode": "10001",
      "street": "Jayanagar"
    }
  },
  {
    "name": "Kyotos",
    "address": {
      "zipcode": "10300",
      "street": "Majestic"
    }
  },
  {
    "name": "Empire",
    "address": {
      "zipcode": "10100",
      "street": "MG Road"
    }
  }
]
```