VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

PRATHEEKSHA PAI (1BM23CS245)

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 **PRATHEEKSHA PAI (1BM23CS245),** 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.

Lab faculty Incharge Name Assistant Professor Department of CSE, BMSCE	Dr. Joythi S Nayak Professor HOD Department of CSE, BMSCE

Index

Sl. No.	Date	Experiment Title	Page No.
1		Insurance Database	4-11
2		More Queries on Insurance Database	12-15
3		Bank Database	16-24
4		More Queries on Bank Database	25-30
5		Employee Database	31-40
6		More Queries on Employee Database	41-43
7		Supplier Database	44-50
8		NO SQL - Student Database	51-56
9		NO SQL - Customer Database	57-61
10		NO SQL – Restaurant Database	62-86

Insurance Database

Question

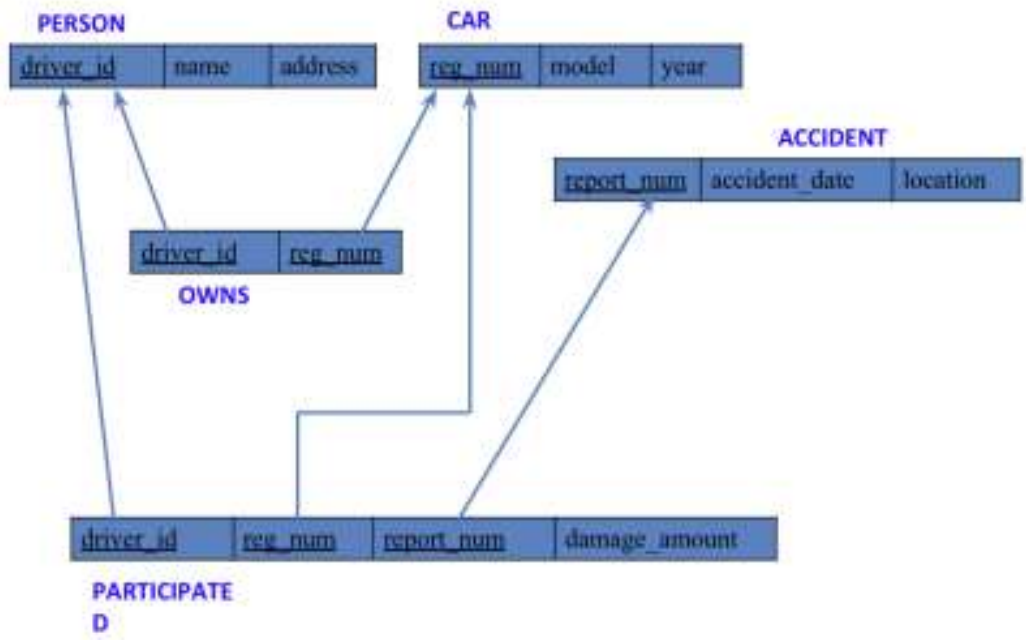
(Week 1)

- 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

```
create database insurance;

use insurance;
```

Create table

```
create table insurance.person(

driver_id varchar(20),

name varchar(30),

address varchar(50),

PRIMARY KEY(driver_id)

);

create table insurance.car(

reg_num varchar(15),

model varchar(10),

year int,
```

```
PRIMARY KEY(reg_num)
);
create table insurance.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 insurance.accident(
report_num int,
accident_date date,
location varchar(50),
PRIMARY KEY(report_num)
);
create table insurance.participated(
driver_id varchar(20),
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	
name	varchar(20)	YES		NULL	
address	varchar(20)	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(20)	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(20)	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(20)	NO	PRI	NULL	
model	varchar(20)	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(20)	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;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

driver_id	name	address
A01	Richard	Srinivas nagar
A02	Pradeep	Rajaji nagar
A03	Smith	Ashok nagar
A04	Venu	N R 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: | Edit: | Export/Import: | Wrap Cell Content:

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","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: | Edit: | Export/Import: | Wrap Cell Content:

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

insert into accident values(11,'2003-01-01","Mysore Road");
insert into accident values(12,'2004-02-02","South end Circle");

insert into accident values(13,'2003-01-21","Bull temple Road");

insert into accident values(14,'2008-02-17","Mysore Road");

insert into accident values(15,'2004-03-05","Kanakpura Road");

select * from accident;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

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: | Export/Import: | Wrap Cell Content:

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

- **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: | Export/Import: | Wrap Cell Content:

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

- **Display Accident date and location**

```
select accident_date, location
from ACCIDENT;
```

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

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

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

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

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

report_num	accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2005-03-04	Kanakpura Road
16	2008-03-08	Domlur
NULL	NULL	NULL

- **Display Accident date and location**

```
select accident_date, location
from ACCIDENT;
```

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

accident_date	location
2003-01-01	Mysore Road
2004-02-02	South end Circle
2003-01-21	Bull temple Road
2008-02-17	Mysore Road
2005-03-04	Kanakpura Road
2008-03-08	Domlur

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

```
select driver_id
from PARTICIPATED
where damage_amount >=25000;
```

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

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)
- Create the above tables by properly specifying the primary keys and the foreign keys as done in “Program-1”week’s lab and Enter at least five tuples for each relation.
- Display the entire CAR relation in the ascending order of manufacturing year.
- Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
- Find the total number of people who owned cars that involved in accidents in 2008.
- List the Entire Participated Relation in the Descending Order Of Damage Amount. Find the Average Damage Amount
- Delete the Tuple Whose Damage Amount is below the Average Damage Amount
- List the Name Of Drivers Whose Damage is Greater than The Average Damage Amount.
- Find Maximum Damage Amount.

Queries

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

  select * from car order by year asc;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

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

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

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

CNT
1

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

  select count(distinct driver_id) CNT
  from participated a, accident b
  where a.report_num=b.report_num and b.accident_date like '2008%';

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

CNT
1

- **List the Entire Participated Relation in the Descending Order Of Damage Amount. Find the Average Damage Amount**

SELECT * FROM PARTICIPATED ORDER BY DAMAGE_AMOUNT DESC;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

SELECT AVG(DAMAGE_AMOUNT) FROM PARTICIPATED;

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

AVG(DAMAGE_AMOUNT)
18600.0000

- **Delete the Tuple Whose Damage Amount is below the Average Damage Amount**

  DELETE FROM PARTICIPATED WHERE DAMAGE_AMOUNT < 18600;

  SELECT * FROM participated;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

driver_id	reg_num	report_num	damage_amount
A02	KA053408	12	50000
A03	KA095477	13	25000
NULL	NULL	NULL	NULL

- **List the Name Of Drivers Whose Damage is Greater than The Average Damage Amount.**

  SELECT NAME FROM PERSON A, PARTICIPATED B WHERE A.DRIVER_ID = B.DRIVER_ID AND DAMAGE_AMOUNT > (SELECT AVG(DAMAGE_AMOUNT) FROM PARTICIPATED);

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

NAME
Pradeep
Smith

- **Find Maximum Damage Amount.**

  SELECT MAX(DAMAGE_AMOUNT) FROM PARTICIPATED;

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

	MAX(DAMAGE_AMOUNT)
▶	50000

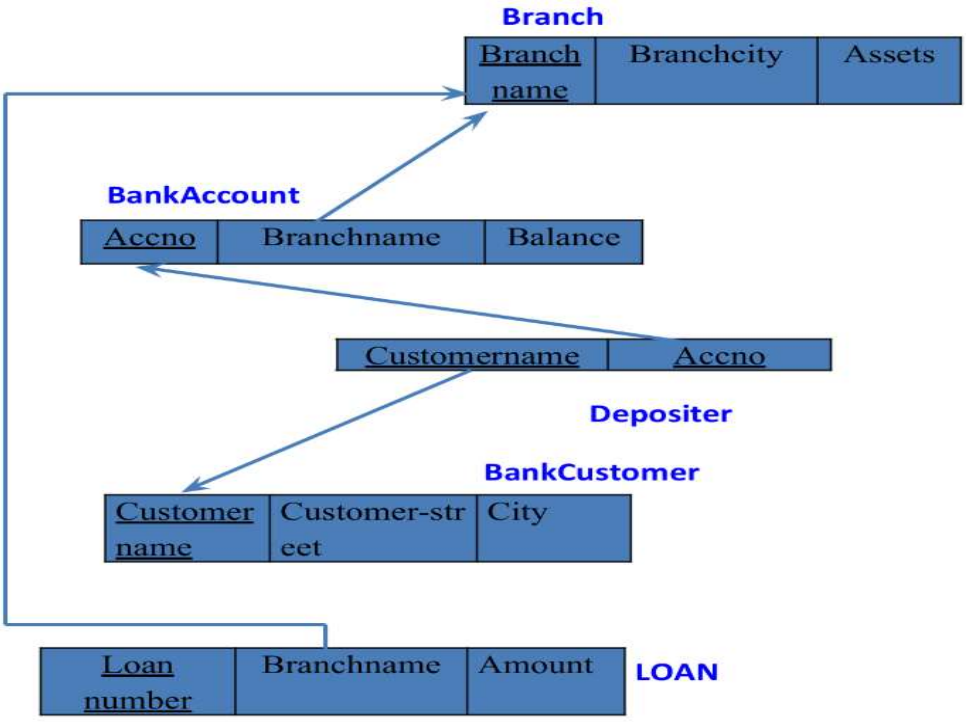
Bank Database

Question

(Week3)

- 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 bank.branch(

  Branch_name varchar(30),

  Branch_city varchar(25),

  assets int,

  PRIMARY KEY (Branch_name)
);


create table bank.BankAccount(
```

```
  Accno int,

  Branch_name varchar(30),

  Balance int,

  PRIMARY KEY(Accno),

  foreign key (Branch_name) references branch(Branch_name)
);


create table bank.BankCustomer(

  Customername varchar(20),

  Customer_street varchar(30),

  CustomerCity varchar (35),

  PRIMARY KEY(Customername)
);


create table bank.Depositer(

  Customername varchar(20),

  Accno int,

  PRIMARY KEY(Customername,Accno),

  foreign key (Accno) references BankAccount(Accno),

  foreign key (Customername) references BankCustomer(Customername)
);


create table bank.Loan(

  Loan_number int,

  Branch_name varchar(30),
```

```
  Amount int,

  PRIMARY KEY(Loan_number),

  foreign key (Branch_name) references branch(Branch_name)
);
```

Structure of the table

desc branch;

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

Field	Type	Null	Key	Default	Extra
Branch_name	varchar(30)	NO	PRI	NULL	
Branch_city	varchar(25)	YES		NULL	
assets	int	YES		NULL	

desc BankAccount;

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

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

desc BankCustomer;

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

Field	Type	Null	Key	Default	Extra
Customername	varchar(20)	NO	PRI	NULL	
Customer_street	varchar(30)	YES		NULL	
CustomerCity	varchar(35)	YES		NULL	

desc Depositer;

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

Field	Type	Null	Key	Default	Extra
Customername	varchar(20)	YES	MUL	NULL	
Accno	int	YES	MUL	NULL	

desc Loan;

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

Field	Type	Null	Key	Default	Extra
Loan_number	int	NO	PRI	NULL	
Branch_name	varchar(30)	YES	MUL	NULL	
Amount	int	YES		NULL	

Inserting values to the table

insert into branch values("SBI_Chamrajpet","Bangalore",50000);

insert into branch values("SBI_ResidencyRoad","Bangalore",10000);

insert into branch values("SBI_ShivajiRoad","Bombay",20000);

insert into branch values("SBI_ParlimentRoad","Delhi",10000);

insert into branch values("SBI_Jantarmantar","Delhi",20000);

select * from branch;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Branch_name	Branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	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_Jantarmantar",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_Jantarmantar",2000);

select * from BankAccount;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

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

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Customername	Customer_street	CustomerCity
Avinash	Bull_Temple_Road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	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;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Customername	Accno
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
NULL	NULL

```
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: | Edit: | Export/Import: | Wrap Cell Content:

Loan_number	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 branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.**

```
select Branch_name, CONCAT(assets/100000, 'lakhs')assets_in_lakhs from branch;
```

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

Branch_name	assets_in_lakhs
SBI_Chamrajpet	0.5000 lakhs
SBI_Jantarmantar	0.2000 lakhs
SBI_ParlimentRoad	0.1000 lakhs
SBI_ResidencyRoad	0.1000 lakhs
SBI_ShivajiRoad	0.2000 lakhs

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

```
select d.Customername from Depositer d, BankAccount b where
b.Branch_name='SBI_ResidencyRoad' and d.Accno=b.Accno group by d.Customername
having count(d.Accno)>=2;
```

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

Customername
Dinesh

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

```
create view sum_of_loan
as select Branch_name, SUM(Balance)
from BankAccount
group by Branch_name;

select * from sum_of_loan;
```

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

Branch_name	SUM(Balance)
SBI_Chamrajpet	2000
SBI_Jantarmantar	10000
SBI_ParlimentRoad	12000
SBI_ResidencyRoad	14000
SBI_ShivajiRoad	10000

More Queries on Bank Database

Question

(Week4)

- 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)
- 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%

Creating borrower table

```
create table borrower(
Customername varchar(20),
Loan_number int,
foreign key(Customername) references bankcustomer(Customername),
foreign key(Loan_number) references loan(Loan_number)
);
desc borrower;
```

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

Field	Type	Null	Key	Default	Extra
Customername	varchar(20)	YES	MUL	NULL	
Loan_number	int	YES	MUL	NULL	

Inserting values

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

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

Branch_name	Branch_city	assets
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

```
select * from bankaccount;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Accno	Branch_name	Balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
3	SBI_ShivajiRoad	6300
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
6	SBI_ShivajiRoad	4200
8	SBI_ResidencyRoad	4200
9	SBI_ParlimentRoad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
12	SBI_MantriMarg	2000
NULL	NULL	NULL

select * from depositor;

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

Customername	Accno
Avinash	1
Dinesh	2
Nikil	4
Ravi	5
Avinash	8
Nikil	9
Dinesh	10
Nikil	11
Nikil	12

select * from borrower;

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

Customername	Loan_number
Avinash	1
Dinesh	2
Mohan	3
Nikil	4
Ravi	5

Queries

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

```
select distinct S.Customername
from Depositer as S
where not exists ( (select branch_name
                        from branch
                        where branch_city = 'Delhi') except
                    (select R.branch_name
                    from Depositer as T, BankAccount as R
                    where T.Accno = R.Accno
                    and S.Customername = T.Customername));
```

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

Customername
Nikil

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

```
select distinct Customername
    from BankCustomer where Customername not in
    (select Customername from Depositer);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Customername
Mohan
NULL

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

```
select distinct Customername from depositer
where Customername in (select depositer.Customername from branch,
bankaccount, depositer
where branch.Branch_city = 'Bangalore' and branch.Branch_name =
bankaccount.Branch_name and bankaccount.Accno = depositer.Accno) and
Customername in (select Customername from borrower, loan where Branch_name in
(select Branch_name from branch where Branch_city = 'Bangalore'));
```

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

Customername
Dinesh
Avinash

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

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

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

branch_name

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

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

select * from bankaccount;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

Accno	Branch_name	Balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
3	SBI_ShivajiRoad	6300
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
6	SBI_ShivajiRoad	4200
8	SBI_ResidencyRoad	4200
9	SBI_ParlimentRoad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
NULL	NULL	NULL

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

```
delete from bankaccount where Branch_name in (select Branch_name
from branch where Branch_city = 'Bombay');

select * from bankaccount;
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

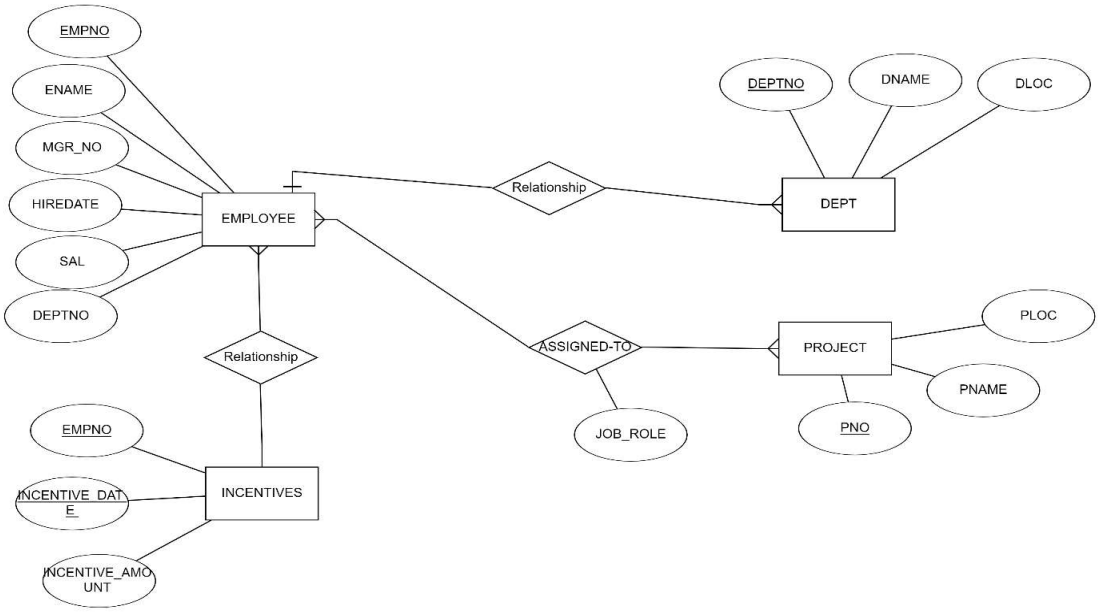
Accno	Branch_name	Balance
1	SBI_Chamrajpet	2100
2	SBI_ResidencyRoad	5250
4	SBI_ParlimentRoad	9450
5	SBI_Jantarmantar	8400
8	SBI_ResidencyRoad	4200
9	SBI_ParlimentRoad	3150
10	SBI_ResidencyRoad	5250
11	SBI_Jantarmantar	2100
12	SBI_MantriMarg	2000
NULL	NULL	NULL

Employee Database

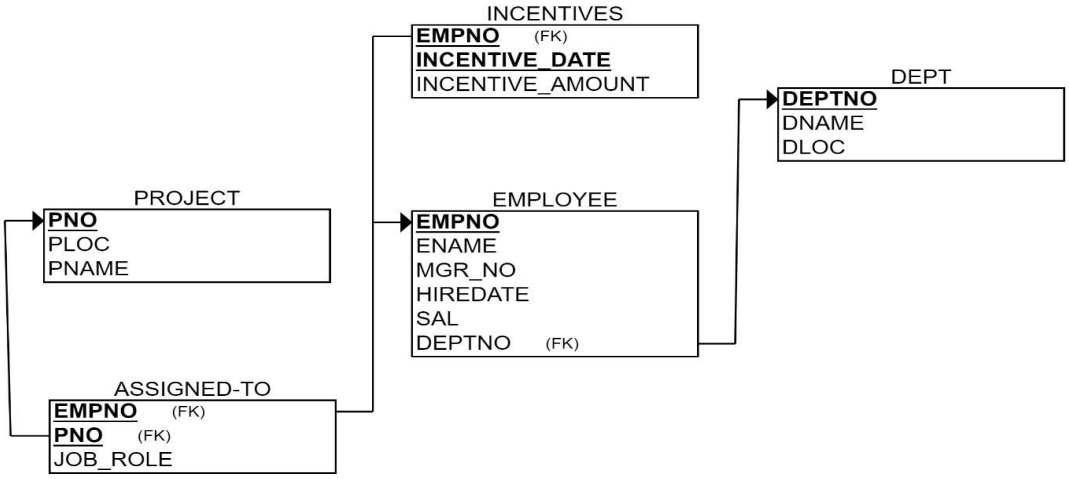
Question

(Week 5)

ER Diagram



Schema Diagram



Question

- 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.

Create database

```
create database emp;

use emp;
```

Create table

```
CREATE TABLE dept (

  deptno decimal(2,0) primary key,

  dname varchar(14) default NULL,

  loc varchar(13) default NULL
);


CREATE TABLE emp (

  empno decimal(4,0) primary key,

  ename varchar(10) default NULL,

  mgr_no decimal(4,0) default NULL,

  hiredate date default NULL,

  sal decimal(7,2) default NULL,

  deptno decimal(2,0) references dept(deptno) on delete cascade on update cascade
);


create table incentives (

  empno decimal(4,0) references emp(empno) on delete cascade on update cascade,

  incentive_date date,

  incentive_amount decimal(10,2),

  primary key(empno,incentive_date)
  );


Create table project (

 pno int primary key,
```

```
pname varchar(30) not null,

ploc varchar(30)
);
```

```
Create table assigned_to (

empno decimal(4,0) references emp(empno) on delete cascade on update cascade,

pno int references project(pno) on delete cascade on update cascade,

job_role varchar(30),

primary key(empno,pno)
);
```

Structure of the table

desc dept;

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

Field	Type	Null	Key	Default	Extra
deptno	decimal(2,0)	NO	PRI	NULL	
dname	varchar(14)	YES		NULL	
loc	varchar(13)	YES		NULL	

desc emp;

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

Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
ename	varchar(10)	YES		NULL	
mgr_no	decimal(4,0)	YES		NULL	
hiredate	date	YES		NULL	
sal	decimal(7,2)	YES		NULL	
deptno	decimal(2,0)	YES		NULL	

desc incentives;

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

Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
incentive_date	date	NO	PRI	NULL	
incentive_amount	decimal(10,2)	YES		NULL	

desc project;

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

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

desc assigned_to;

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

Field	Type	Null	Key	Default	Extra
empno	decimal(4,0)	NO	PRI	NULL	
pno	int	NO	PRI	NULL	
job_role	varchar(30)	YES		NULL	

Inserting values to the table

INSERT INTO dept VALUES (10,'ACCOUNTING','MUMBAI');

INSERT INTO dept VALUES (20,'RESEARCH','BENGALURU');

INSERT INTO dept VALUES (30,'SALES','DELHI');

INSERT INTO dept VALUES (40,'OPERATIONS','CHENNAI');

Select * from dept;

deptno	dname	loc
10	ACCOUNTING	MUMBAI
20	RESEARCH	BENGALURU
30	SALES	DELHI
40	OPERATIONS	CHENNAI
NULL	NULL	NULL

INSERT INTO emp VALUES (7369,'Adarsh',7902,'2012-12-17','80000.00','20');

INSERT INTO emp VALUES (7499,'Shruthi',7698,'2013-02-20','16000.00','30');

INSERT INTO emp VALUES (7521,'Anvitha',7698,'2015-02-22','12500.00','30');

INSERT INTO emp VALUES (7566,'Tanvir',7839,'2008-04-02','29750.00','20');

INSERT INTO emp VALUES (7654,'Ramesh',7698,'2014-09-28','12500.00','30');

INSERT INTO emp VALUES (7698,'Kumar',7839,'2015-05-01','28500.00','30');

INSERT INTO emp VALUES (7782,'CLARK',7839,'2017-06-09','24500.00','10');

INSERT INTO emp VALUES (7788,'SCOTT',7566,'2010-12-09','30000.00','20');

INSERT INTO emp VALUES ('7839','KING',NULL,'2009-11-17','50000.00','10');

INSERT INTO emp VALUES ('7844','TURNER',7698,'2010-09-08','15000.00','30');

INSERT INTO emp VALUES ('7876','ADAMS',7788,'2013-01-12','11000.00','20');

INSERT INTO emp VALUES ('7900','JAMES',7698,'2017-12-03','9500.00','30');

INSERT INTO emp VALUES ('7902','FORD','7566','2010-12-03','30000.00','20');

Select * from emp;

Result Grid | Filter Rows: | Edit:

empno	ename	mgr_no	hiredate	sal	deptno
7369	Adarsh	7902	2012-12-17	80000.00	20
7499	Shruthi	7698	2013-02-20	16000.00	30
7521	Anvitha	7698	2015-02-22	12500.00	30
7566	Tanvir	7839	2008-04-02	29750.00	20
7654	Ramesh	7698	2014-09-28	12500.00	30
7698	Kumar	7839	2015-05-01	28500.00	30
7782	CLARK	7839	2017-06-09	24500.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
7839	KING	NULL	2009-11-17	50000.00	10
7844	TURNER	7698	2010-09-08	15000.00	30
7876	ADAMS	7788	2013-01-12	11000.00	20
7900	JAMES	7698	2017-12-03	9500.00	30
7902	FORD	7566	2010-12-03	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

emp 1

INSERT INTO incentives VALUES(7499,'2019-02-01',5000.00);

INSERT INTO incentives VALUES(7521,'2019-03-01',2500.00);

INSERT INTO incentives VALUES(7566,'2022-02-01',5070.00);

INSERT INTO incentives VALUES(7654,'2020-02-01',2000.00);

INSERT INTO incentives VALUES(7654,'2022-04-01',879.00);

INSERT INTO incentives VALUES(7521,'2019-02-01',8000.00);

INSERT INTO incentives VALUES(7698,'2019-03-01',500.00);

INSERT INTO incentives VALUES(7698,'2020-03-01',9000.00);

INSERT INTO incentives VALUES(7698,'2022-04-01',4500.00);

Select * from incentives;

Result Grid | Filter Rows:

empno	incentive_date	incentive_amount
7499	2019-02-01	5000.00
7521	2019-02-01	8000.00
7521	2019-03-01	2500.00
7566	2022-02-01	5070.00
7654	2020-02-01	2000.00
7654	2022-04-01	879.00
7698	2019-03-01	500.00
7698	2020-03-01	9000.00
7698	2022-04-01	4500.00
NULL	NULL	NULL

INSERT INTO project VALUES(101,'AI Project','BENGALURU');

INSERT INTO project VALUES(102,'IOT','HYDERABAD');

INSERT INTO project VALUES(103,'BLOCKCHAIN','BENGALURU');

INSERT INTO project VALUES(104,'DATA SCIENCE','MYSURU');

INSERT INTO project VALUES(105,'AUTONOMUS SYSTEMS','PUNE');

Select * from project;

Result Grid | Filter Rows:

pno	pname	ploc
101	AI Project	BENGALURU
102	IOT	HYDERABAD
103	BLOCKCHAIN	BENGALURU
104	DATA SCIENCE	MYSURU
105	AUTONOMUS SYSTEMS	PUNE
NULL	NULL	NULL

INSERT INTO assigned_to VALUES(7499,101,'Software Engineer');

INSERT INTO assigned_to VALUES(7521,101,'Software Architect');

INSERT INTO assigned_to VALUES(7566,101,'Project Manager');

INSERT INTO assigned_to VALUES(7654,102,'Sales');

INSERT INTO assigned_to VALUES(7521,102,'Software Engineer');

INSERT INTO assigned_to VALUES(7499,102,'Software Engineer');

INSERT INTO assigned_to VALUES(7654,103,'Cyber Security');

INSERT INTO assigned_to VALUES(7698,104,'Software Engineer');

INSERT INTO assigned_to VALUES(7900,105,'Software Engineer');

INSERT INTO assigned_to VALUES(7839,104,'General Manager');

Select * from assigned_to;

Result Grid | Filter Rows:

empno	pno	job_role
7499	101	Software Engineer
7499	102	Software Engineer
7521	101	Software Architect
7521	102	Software Engineer
7566	101	Project Manager
7654	102	Sales
7654	103	Cyber Security
7698	104	Software Engineer
7839	104	General Manager
7900	105	Software Engineer
NULL	NULL	NULL

Queries

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

```
select e.empno
from emp e, assigned_to a, project p
where e.empno=a.empno and a.pno= p.pno and p.ploc in ('Bengaluru','Hyderabad','Mysuru');
```

Result Grid | Filter Rows:

empno
7499
7499
7521
7521
7566
7654
7654
7698
7839

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

```
select empno from emp
where empno not in (select empno from incentives);
```

Result Grid

empno
7369
7782
7788
7839
7844
7876
7900
7902
NULL

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

```
select e.ename, e.empno, d.dname, a.job_role, d.loc, p.ploc
from emp e, dept d, assigned_to a, project p
where e.deptno = d.deptno and e.empno = a.empno and a.pno = p.pno and d.loc = p.ploc;
```

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

ename	empno	dname	job_role	loc	ploc
Tanvir	7566	RESEARCH	Project Manager	BENGALURU	BENGALURU

More Queries on Employee Database

Question

(Week 6)

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

Queries

- **List the name of the managers with the maximum employees**

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

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

ename	count(*)
Kumar	5

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

```
select * from emp m
where m.empno in (select mgr_no from emp) and
m.sal > (select avg(e.sal) from emp e where e.mgr_no=m.empno);
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Conten

empno	ename	mgr_no	hiredate	sal	deptno
7698	Kumar	7839	2015-05-01	28500.00	30
7839	KING	NULL	2009-11-17	50000.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

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

```
select * from emp where empno=
(select i.empno from incentives i
where i.incentive_amount= (select max(n.incentive_amount) from incentives n
where n.incentive_amount < (select max(inc.incentive_amount) from incentives inc
where inc.incentive_date between '2019-02-01' and '2019-02-29') and incentive_date
between '2019-02-01' and '2019-02-29'));
```

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	ename	mgr_no	hiredate	sal	deptno
NULL	NULL	NULL	NULL	NULL	NULL

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

```
SELECT *
FROM EMP E
WHERE E.DEPTNO = (SELECT E1.DEPTNO
                  FROM EMP E1
                  WHERE E1.EMPNO=E.MGR_NO);
```

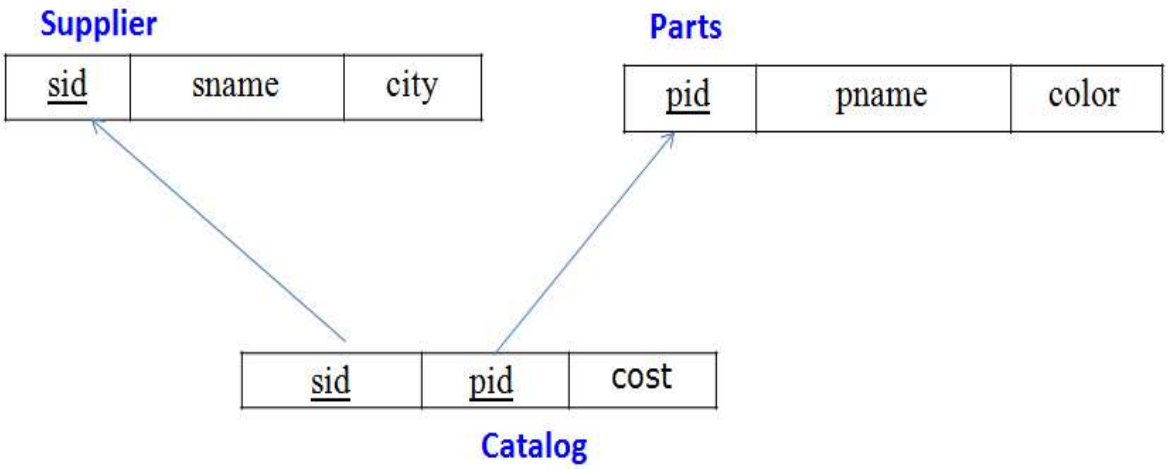
Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell Content:

empno	ename	mgr_no	hiredate	sal	deptno
7369	Adarsh	7902	2012-12-17	80000.00	20
7499	Shruthi	7698	2013-02-20	16000.00	30
7521	Anvitha	7698	2015-02-22	12500.00	30
7654	Ramesh	7698	2014-09-28	12500.00	30
7782	CLARK	7839	2017-06-09	24500.00	10
7788	SCOTT	7566	2010-12-09	30000.00	20
7844	TURNER	7698	2010-09-08	15000.00	30
7876	ADAMS	7788	2013-01-12	11000.00	20
7900	JAMES	7698	2017-12-03	9500.00	30
7902	FORD	7566	2010-12-03	30000.00	20
NULL	NULL	NULL	NULL	NULL	NULL

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.

Create database

```
create database supplier;
use supplier;
```

Create tables

```
create table supplier(
        sid int,
        sname varchar(20),
        city varchar(30),
        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,
        primary key(sid, pid),
        foreign key(sid) references supplier(sid),
        foreign key(pid) references parts(pid));
```

Structure of the table

```
desc supplier;
```

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

Field	Type	Null	Key	Default	Extra
sid	int	NO	PRI	NULL	
sname	varchar(20)	YES		NULL	
city	varchar(30)	YES		NULL	

desc parts;

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

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

desc catalog;

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

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

Inserting values to the table

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;

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap C

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

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

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

Result Grid | Filter Rows: | Edit: | Export/Import: | Wrap Cell

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
NULL	NULL	NULL

Queries

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

```
select pname from parts
where pid in (select pid from catalog);
```

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

pname
Book
Pen
Pencil
Mobile
Charger

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

```
select sname from supplier
where sid in (select sid from catalog
group by sid having count(distinct pid)=(select count(distinct pid) from parts));
```

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

sname
Acme widget

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

```
select distinct s.sname
from supplier s, parts p, catalog c
where s.sid = c.sid and p.pid = c.pid
and p.color = 'Red';
```

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

sname
Acme widget
Johns

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

```
select distinct pname
from parts
where pid not in (select pid from catalog
where sid in (select sid from supplier
where sname != 'Acme widget'));
```

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

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 sid from catalog a
where a.cost > (select avg(b.cost) from catalog b
where a.pid = b.pid group by b.pid);
```

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

sid
10002
10004

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

```
select distinct pid, sname from catalog a, supplier
where a.cost = (select max(b.cost) from catalog b where a.pid = b.pid group by b.pid)
and supplier.sid = a.sid;
```

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

pid	sname
20001	Acme widget
20004	Acme widget
20005	Acme widget
20001	Johns
20002	Johns
20003	Reliance

NOSQL - Student database

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

5. Export the created table into local file system

6. Drop the table

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

QUERIES

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

db.createCollection("Student");

-Insert appropriate values

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

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

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

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

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

```
> db.Student.insertOne({RollNo:1,Age:21,Cont:9876,email:"antara.de9@gmail.com"});
< {
    acknowledged: true,
    insertedId: ObjectId('67613f4cfa32f36a75f77673')
  }
> db.Student.insertOne({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});
< {
    acknowledged: true,
    insertedId: ObjectId('67613f65fa32f36a75f77674')
  }
> db.Student.insertOne({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});
< {
    acknowledged: true,
    insertedId: ObjectId('67613f75fa32f36a75f77675')
  }
> db.Student.insertOne({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
< {
    acknowledged: true,
    insertedId: ObjectId('67613f81fa32f36a75f77676')
  }
> db.Student.insertOne({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
< {
    acknowledged: true,
```

-Structure of student

```
Atlas atlas-a8ybrh-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("676144746073613f58f84d5d"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6761448d6073613f58f84d5e"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("676144976073613f58f84d5f"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("676144a26073613f58f84d60"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
```

```
  {
    _id: ObjectId("676144a26073613f58f84d60"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("676144ab6073613f58f84d61"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

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

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

```
Atlas atlas-a8ybrh-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-a8ybrh-shard-0 [primary] test> db.Student.find()
[
  {
    _id: ObjectId("676144746073613f58f84d5d"),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId("6761448d6073613f58f84d5e"),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId("676144976073613f58f84d5f"),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId("676144a26073613f58f84d60"),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId("676144ab6073613f58f84d61"),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'Abhinav@gmail.com'
  }
]
Atlas atlas-a8ybrh-shard-0 [primary] test>
```

- **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"});

```
  },
  {
    _id: ObjectId("676147056073613f58f84d62"),
    RollNo: 11,
    Age: 22,
    Name: 'FEM',
    Cont: 2276,
    email: 'rea.de9@gmail.com'
  }
]
Atlas atlas-a8ybrh-shard-0 [primary] test>
```

- **Export the created table into local file system**

mongoexport mongodb+srv://antararc:Test1234@cluster0.mfnfeys.mongodb.net/myDB --collection=Student --out C:\Users\BMSCECSE\Downloads\output.json

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

- **Drop the table**

db.Student.drop();

```
Atlas atlas-mdgaz1-shard-0 [primary] myDB> db.Student.drop();
true
Atlas atlas-mdgaz1-shard-0 [primary] myDB> db.Student.find()
```

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

mongoimport mongodb+srv://antararc:Test1234@cluster0.mfnfeys.mongodb.net/myDB --collection=New_Student --type json --file C:\Users\BMSCECSE\Downloads\output.json

```
C:\Users\BMSCECSE>mongoimport  mongodb+srv://antararc:Test1234@cluster0.mfnfeys.mongodb.net/myDB --collection=New_Student  --type json --file C:\Users\BMSCECSE\Downloads\output.json
2023-01-12T15:17:35.523+0530   connected to: mongodb+srv://[**REDACTED**]@cluster0.mfnfeys.mongodb.net/myDB
2023-01-12T15:17:35.640+0530   7 document(s) imported successfully. 0 document(s) failed to import.
```

NoSQL - Customer database

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 ‘Z’ for each customer_id.

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

5. Export the created collection into local file system

6. Drop the table

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

Create Table:

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()

```
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('6761445e6941d899bfb78130'),
    '1': ObjectId('6761445e6941d899bfb78131'),
    '2': ObjectId('6761445e6941d899bfb78132'),
    '3': ObjectId('6761445e6941d899bfb78133'),
    '4': ObjectId('6761445e6941d899bfb78134')
  }
}
db.Customer.find();
{
  _id: ObjectId('6761445e6941d899bfb78130'),
  custid: 1,
  acc_bal: 10000,
  acc_type: 'Saving'
}
{
  _id: ObjectId('6761445e6941d899bfb78131'),
  custid: 1,
  acc_bal: 20000,
  acc_type: 'Checking'
}
{
  _id: ObjectId('6761445e6941d899bfb78132'),
  custid: 3,
```

```
  acc_type: 'Checking'
}
{
  _id: ObjectId('6761445e6941d899bfb78132'),
  custid: 3,
  acc_bal: 50000,
  acc_type: 'Checking'
}
{
  _id: ObjectId('6761445e6941d899bfb78133'),
  custid: 4,
  acc_bal: 10000,
  acc_type: 'Saving'
}
{
  _id: ObjectId('6761445e6941d899bfb78134'),
  custid: 5,
  acc_bal: 2000,
  acc_type: 'Checking'
}
```

Queries:

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

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

```
> db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});
< {
    _id: ObjectId('6761445e6941d899bfb78131'),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  }
  {
    _id: ObjectId('6761445e6941d899bfb78132'),
    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"}}}]);

```
> db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal: {$max:"$acc_bal"}}}]);
< {
    _id: 4,
    minBal: 10000,
    maxBal: 10000
  }
  {
    _id: 5,
    minBal: 2000,
    maxBal: 2000
  }
  {
    _id: 3,
    minBal: 50000,
    maxBal: 50000
  }
  {
    _id: 1,
    minBal: 10000,
    maxBal: 20000
  }
> mongoexport mongodb+srv://antararc:<password>@cluster0.mfnfeys.mongodb.net/myDB
```

- **Export the created collection into local file system**

  mongoexport mongodb+srv://Test1234@cluster0.xbmgopf.mongodb.net/test --collection=Customer -- out D:\1BM23CS268\l1.json


```
{"_id":{"$oid":"674ff8498df86e77109b3e36"},"custid":1,"acc_bal":10000,"acc_type":"Saving"}
{"_id":{"$oid":"674ff86c8df86e77109b3e37"},"custid":1,"acc_bal":20000,"acc_type":"Checking"}
{"_id":{"$oid":"674ff8b78df86e77109b3e38"},"custid":3,"acc_bal":50000,"acc_type":"Checking"}
{"_id":{"$oid":"674ff8c88df86e77109b3e39"},"custid":4,"acc_bal":10000,"acc_type":"Saving"}
{"_id":{"$oid":"674ff8e68df86e77109b3e3a"},"custid":5,"acc_bal":2000,"acc_type":"Checking"}
```


- **Drop the table**

  db.Customer.drop();

```
> db.Customer.drop();
< true
```

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

  mongoimport mongodb+srv://Test1234@cluster0.xbmgopf.mongodb.net/test --collection=Customer --file D:\1BM23CS268\l1.json

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'
  }
]
```

NOSQL - Restaurant database

QUESTION

(WEEK 10)

- Write NoSQL Queries on “Restaurant” collection
- Write a MongoDB query to display all the documents in the collection restaurants.
- Write a MongoDB query to arrange the name of the restaurants in descending along with all the columns.
- Write a MongoDB query to find the restaurant Id, name, town and cuisine for those restaurants which achieved a score which is not more than 10.
- Write a MongoDB query to find the average score for each restaurant.
- Write a MongoDB query to find the name and address of the restaurants that have a zipcode that starts with '10'.

Creating Table:

db.createCollection("Restaurant");

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

Inserting Values:

db.Restraunt.insertMany([{ "address": { "building": "1007", "coord": [-73.856077, 48.848447], "street": "Morris Park Ave", "zipcode": "18462", "borough": "Bronx" }, "cuisine": "Bakery", "grades": [{"date": new Date("2014-03-03"), "grade": "A", "score": 2}, {"date": new Date("2013-09-11"), "grade": "A", "score": 6}, {"date": new Date("2013-01-24"), "grade": "A", "score": 10}, {"date": new Date("2011-11-23"), "grade": "A", "score": 9}, {"date": new Date("2011-03-10"), "grade": "B", "score": 14}

"name": "Morris Park Bake Shop",
"restaurant_id": "30075445"
},
{
"address": {
"building": "2001",
"coord": [-74.005941, 40.712776],
"street": "Broadway",
"zipcode": "10001",
"borough": "Manhattan"
},
"cuisine": "Italian",
"grades": [
{"date": new Date("2015-08-20"), "grade": "A", "score": 8},
{"date": new Date("2014-06-10"), "grade": "B", "score": 4},
{"date": new Date("2013-12-15"), "grade": "A", "score": 11},
{"date": new Date("2012-09-30"), "grade": "A", "score": 9},
{"date": new Date("2011-05-12"), "grade": "A", "score": 12}
],
"name": "Pasta Paradise",
"restaurant_id": "40092138"
},
{
"address": {
"building": "3003",
"coord": [-118.243685, 34.052235],
"street": "Hollywood Blvd",
"zipcode": "90028",

"borough": "Los Angeles"
},
"cuisine": "Mexican",
"grades": [
{"date": new Date("2016-04-15"), "grade": "A", "score": 9},
{"date": new Date("2015-12-05"), "grade": "B", "score": 6},
{"date": new Date("2014-09-20"), "grade": "A", "score": 11},
{"date": new Date("2013-06-18"), "grade": "A", "score": 8},
{"date": new Date("2012-02-10"), "grade": "A", "score": 10}
],
"name": "Sizzling Tacos",
"restaurant_id": "50065432"
},
{
"address": {
"building": "4004",
"coord": [77.209021, 28.613939],
"street": "Connaught Place",
"zipcode": "110001",
"borough": "New Delhi"
},
"cuisine": "Indian",
"grades": [
{"date": new Date("2019-10-25"), "grade": "A", "score": 8},
{"date": new Date("2018-07-15"), "grade": "B", "score": 5},
{"date": new Date("2017-04-30"), "grade": "A", "score": 10},
{"date": new Date("2016-01-12"), "grade": "A", "score": 9},
{"date": new Date("2015-05-20"), "grade": "A", "score": 12}

```
    ],
    "name": "Spice Delight",
    "restaurant_id": "60098765"
  },
{
    "address": {
      "building": "5005",
      "coord": [76.780253, 30.728592],
      "street": "Balle Balle Lane",
      "zipcode": "160022",
      "borough": "Chandigarh"
    },
    "cuisine": "Punjabi",
    "grades": [
      {"date": new Date("2020-12-10"), "grade": "A", "score": 9},

      {"date": new Date("2019-08-25"), "grade": "B", "score": 7},
      {"date": new Date("2018-04-15"), "grade": "A", "score": 11},
      {"date": new Date("2017-01-22"), "grade": "A", "score": 8},
      {"date": new Date("2016-06-30"), "grade": "A", "score": 10}
    ],
    "name": "Pind Flavors",
    "restaurant_id": "70087654"
  },
{
    "address": {
      "building": "6006",
      "coord": [77.594562, 12.971598],
      "street": "Vidyarthi Bhavan Road",
```

"zipcode": "560004",
"borough": "Bangalore"
},
"cuisine": "Kannadiga",
"grades": [
{"date": new Date("2021-09-18"), "grade": "A", "score": 8},
{"date": new Date("2020-05-12"), "grade": "B", "score": 6},
{"date": new Date("2019-02-28"), "grade": "A", "score": 10},
{"date": new Date("2018-11-15"), "grade": "A", "score": 9},
{"date": new Date("2017-07-05"), "grade": "A", "score": 12}
],
"name": "Namma Oota",
"restaurant_id": "80076543"
},
{
"address": {
"building": "7007",
"coord": [73.856743, 18.520430],
"street": "Pune-Nashik Highway",
"zipcode": "411001",
"borough": "Pune"
},
"cuisine": "Maharashtrian",
"grades": [
{"date": new Date("2022-05-20"), "grade": "A", "score": 9},
{"date": new Date("2021-01-15"), "grade": "B", "score": 7},
{"date": new Date("2020-08-10"), "grade": "A", "score": 11},
{"date": new Date("2019-04-25"), "grade": "A", "score": 8},

{"date": new Date("2018-10-12"), "grade": "A", "score": 10}
],
"name": "Misal Junction",
"restaurant_id": "90065432"
},
{
"address": {
"building": "7007",
"coord": [73.856743, 18.520430],
"street": "Shivaji Road",
"zipcode": "411001",
"borough": "Pune"
},
"cuisine": "Maharashtrian",
"grades": [
{"date": new Date("2022-04-30"), "grade": "A", "score": 9},
{"date": new Date("2021-10-15"), "grade": "B", "score": 7},
{"date": new Date("2020-06-28"), "grade": "A", "score": 12},
{"date": new Date("2019-03-12"), "grade": "A", "score": 8},
{"date": new Date("2018-08-20"), "grade": "A", "score": 10}
],
"name": "Vyanjan Vihar",
"restaurant_id": "90065432"
},
{
"address": {
"building": "8008",
"coord": [79.312929, 9.288536],

"street": "Temple Road",
"zipcode": "623526",
"borough": "Rameshwaram"
},
"cuisine": "Cafe",
"grades": [
{"date": new Date("2021-07-22"), "grade": "A", "score": 8},
{"date": new Date("2020-02-10"), "grade": "B", "score": 5},
{"date": new Date("2019-09-05"), "grade": "A", "score": 10},
{"date": new Date("2018-04-18"), "grade": "A", "score": 9},
{"date": new Date("2017-11-30"), "grade": "A", "score": 12}
],
"name": "Rameshwaram Retreat",
"restaurant_id": "10076543"
},
{
"address": {
"building": "9009",
"coord": [80.270718, 13.082680],
"street": "Anna Salai",
"zipcode": "600002",
"borough": "Chennai"
},
"cuisine": "Tamil",
"grades": [
{"date": new Date("2022-01-15"), "grade": "A", "score": 8},
{"date": new Date("2021-06-05"), "grade": "B", "score": 6},
{"date": new Date("2020-11-20"), "grade": "A", "score": 11},

{"date": new Date("2019-08-12"), "grade": "A", "score": 9},
{"date": new Date("2018-03-25"), "grade": "A", "score": 10}
],
"name": "Tamil Delicacies",
"restaurant_id": "11076543"
}]);

QUERIES

1) db.Restraunt.find()

```
{
  _id: ObjectId('65e56db05b532e7900b71fef'),
  address: {
    building: '1007',
    coord: [ -73.856077, 48.848447 ],
    street: 'Morris Park Ave',
    zipcode: '18462',
    borough: 'Bronx'
  },
  cuisine: 'Bakery',
  grades: [
    {
      date: ISODate('2014-03-03T00:00:00.000Z'),
      grade: 'A',
      score: 2
    },
    {
      date: ISODate('2013-09-11T00:00:00.000Z'),
      grade: 'A',
      score: 6
    },
    {
      date: ISODate('2013-01-24T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2011-11-23T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2011-03-10T00:00:00.000Z'),
      grade: 'B',
      score: 14
    }
  ],
  name: 'Morris Park Bake Shop',
  restaurant_id: '30075445'
},
{
  _id: ObjectId('65e56db05b532e7900b71ff0'),
  address: {
    building: '2001',
    coord: [ -74.123456, 40.789012 ],
    street: 'Broadway',
    zipcode: '10001'
```

```
},
{
  _id: ObjectId('65e56db05b532e7900b71ff1'),
  address: {
    building: '3003',
    coord: [ -118.243685, 34.052235 ],
    street: 'Hollywood Blvd',
    zipcode: '90028',
    borough: 'Los Angeles'
  },
  cuisine: 'Mexican',
  grades: [
    {
      date: ISODate('2016-04-15T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2015-12-05T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2014-09-20T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2013-06-18T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2012-02-10T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
  name: 'Sizzling Tacos',
  restaurant_id: '50065432'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff2'),
  address: {
    building: '4004',
    coord: [ 77.209021, 28.613939 ],
    street: 'Connaught Place',
    zipcode: '110001',
    borough: 'New Delhi'
  },
  cuisine: 'Indian'
```

```
    borough: 'New Delhi'
  },
  cuisine: 'Indian',
  grades: [
    {
      date: ISODate('2019-10-25T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2018-07-15T00:00:00.000Z'),
      grade: 'B',
      score: 5
    },
    {
```

```
{
  _id: ObjectId('65e56ec65b532e7900b71ff3'),
  address: {
    building: '5005',
    coord: [ 76.780253, 30.728592 ],
    street: 'Balle Balle Lane',
    zipcode: '160022',
    borough: 'Chandigarh'
  },
  cuisine: 'Punjabi',
  grades: [
    {
      date: ISODate('2020-12-10T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2019-08-25T00:00:00.000Z'),
      grade: 'B',
      score: 7
    },
    {
      date: ISODate('2018-04-15T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2017-01-22T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2016-06-30T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
  name: 'Pind Flavors',
  restaurant_id: '70087654'
},
```

```
  restaurant_id: '70007034'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff4'),
  address: {
    building: '6006',
    coord: [ 77.594562, 12.971598 ],
    street: 'Vidyarthi Bhavan Road',
    zipcode: '560004',
    borough: 'Bangalore'
  },
  cuisine: 'Kannadiga',
  grades: [
    {
      date: ISODate('2021-09-18T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2020-05-12T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2019-02-28T00:00:00.000Z'),
```

```
      date: ISODate('2017-07-05T00:00:00.000Z'),
      grade: 'A',
      score: 12
    }
  ],
  name: 'Namma Oota',
  restaurant_id: '80076543'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff5'),
  address: {
    building: '7007',
    coord: [ 73.856743, 18.52043 ],
    street: 'Pune-Nashik Highway',
    zipcode: '411001',
    borough: 'Pune'
  },
  cuisine: 'Maharashtrian',
  grades: [
    {
      date: ISODate('2022-05-20T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2021-01-15T00:00:00.000Z'),
      grade: 'B',
      score: 7
    },
    {
      date: ISODate('2020-08-10T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2019-04-25T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2018-10-12T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
```

```
    },
    ],
    name: 'Misal Junction',
    restaurant_id: '90065432'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff6'),
    address: {
      building: '7007',
      coord: [ 73.856743, 18.52043 ],
      street: 'Shivaji Road',
      zipcode: '411001',
      borough: 'Pune'
    },
    cuisine: 'Maharashtrian',
    grades: [
      {
        date: ISODate('2022-04-30T00:00:00.000Z'),
        grade: 'A',
        score: 9
```

```
    },
    {
      date: ISODate('2021-10-15T00:00:00.000Z'),
      grade: 'B',
      score: 7
    },
    {
      date: ISODate('2020-06-28T00:00:00.000Z'),
      grade: 'A',
      score: 12
    },
    {
      date: ISODate('2019-03-12T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2018-08-20T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
  name: 'Vyanjan Vihar',
  restaurant_id: '90065432'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff7'),
  address: {
    building: '9009',
    coord: [ 80.270718, 13.08268 ],
    street: 'Anna Salai',
    zipcode: '600002',
    borough: 'Chennai'
  },
  cuisine: 'Tamil',
  grades: [
    {
      date: ISODate('2022-01-15T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2021-06-05T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2020-11-20T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2019-08-12T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
```

```
    {
      date: ISODate('2022-01-15T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2021-06-05T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2020-11-20T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2019-08-12T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2018-03-25T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
  name: 'Tamil Delicacies',
```

2) db.Restraunt.find().sort({ "name": -1 });

```
[
  {
    _id: ObjectId('65e56ec65b532e7900b71ff6'),
    address: {
      building: '7007',
      coord: [ 73.856743, 18.52043 ],
      street: 'Shivaji Road',
      zipcode: '411001',
      borough: 'Pune'
    },
    cuisine: 'Maharashtrian',
    grades: [
      {
        date: ISODate('2022-04-30T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2021-10-15T00:00:00.000Z'),
        grade: 'B',
        score: 7
      },
      {
        date: ISODate('2020-06-28T00:00:00.000Z'),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate('2019-03-12T00:00:00.000Z'),
        grade: 'A',
        score: 8
      },
      {
        date: ISODate('2018-08-20T00:00:00.000Z'),
        grade: 'A',
        score: 10
      }
    ],
    restaurant_id: '90065432'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff7'),
    address: {
      building: '9009',
      coord: [ 80.270718, 13.08268 ],
      street: 'Anna Salai',
      zipcode: '600002',
      borough: 'Chennai'
    },
    cuisine: 'Tamil',
    grades: [
      {
        date: ISODate('2022-01-15T00:00:00.000Z'),
        grade: 'A',
```

```
    },
    cuisine: 'Tamil',
    grades: [
      {
        date: ISODate('2022-01-15T00:00:00.000Z'),
        grade: 'A',
        score: 8
      },
      {
        date: ISODate('2021-06-05T00:00:00.000Z'),
        grade: 'B',
        score: 6
      },
      {
        date: ISODate('2020-11-20T00:00:00.000Z'),
        grade: 'A',
        score: 11
      },
      {
        date: ISODate('2019-08-12T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2018-03-25T00:00:00.000Z'),
        grade: 'A',
        score: 10
      }
    ],
    name: 'Tamil Delicacies',
    restaurant_id: '11076543'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff2'),
    address: {
      building: '4004',
      coord: [ 77.209021, 28.613939 ],
      street: 'Connaught Place',
      zipcode: '110001',
      borough: 'New Delhi'
    },
    cuisine: 'Indian',
    grades: [
      {
        date: ISODate('2019-10-25T00:00:00.000Z'),
        grade: 'A',
        score: 8
      },
      {
        date: ISODate('2019-10-25T00:00:00.000Z'),
        grade: 'A',
        score: 8
      },
      {
        date: ISODate('2018-07-15T00:00:00.000Z'),
        grade: 'B',
        score: 5
      },
      {
        date: ISODate('2017-04-30T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2016-01-12T00:00:00.000Z'),
        grade: 'A',
        score: 9
```

```
      score: 12
    }
  ],
  name: 'Spice Delight',
  restaurant_id: '60098765'
},
{
  _id: ObjectId('65e56db05b532e7900b71ff1'),
  address: {
    building: '3003',
    coord: [ -118.243685, 34.052235 ],
    street: 'Hollywood Blvd',
    zipcode: '90028',
    borough: 'Los Angeles'
  },
  cuisine: 'Mexican',
  grades: [
    {
      date: ISODate('2016-04-15T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2015-12-05T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2014-09-20T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2013-06-18T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2012-02-10T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
  restaurant_id: '50065432'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff3'),
  address: {
    building: '5005',
    coord: [ 76.780253, 30.728592 ],
    street: 'Balle Balle Lane',
    zipcode: '160022',
    borough: 'Chandigarh'
  },
  cuisine: 'Punjabi',
  grades: [
    {
      date: ISODate('2020-12-10T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
```

```
      score: 10
    }
  ],
  name: 'Pind Flavors',
  restaurant_id: '70087654'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff4'),
  address: {
    building: '6006',
    coord: [ 77.594562, 12.971598 ],
    street: 'Vidyarthi Bhavan Road',
    zipcode: '560004',
    borough: 'Bangalore'
  },
  cuisine: 'Kannadiga',
  grades: [
    {
      date: ISODate('2021-09-18T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2020-05-12T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2019-02-28T00:00:00.000Z'),
      grade: 'A',
      score: 10
    },
    {
      date: ISODate('2018-11-15T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2017-07-05T00:00:00.000Z'),
      grade: 'A',
      score: 12
    }
```


```
      date: ISODate('2017-07-05T00:00:00.000Z'),
      grade: 'A',
      score: 12
    }
```


```
  ],
  name: 'Namma Oota',
  restaurant_id: '80076543'
},
{
  _id: ObjectId('65e56db05b532e7900b71fef'),
  address: {
    building: '1007',
    coord: [ -73.856077, 40.848447 ],
    street: 'Morris Park Ave',
```

```
    name: 'Namma Oota',
    restaurant_id: '80076543'
  },
  {
    _id: ObjectId('65e56db05b532e7900b71fef'),
    address: {
      building: '1007',
      coord: [ -73.856077, 48.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462',
      borough: 'Bronx'
    },
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        grade: 'A',
        score: 2
      },
      {
        date: ISODate('2013-09-11T00:00:00.000Z'),
        grade: 'A',
        score: 6
      },
      {
        date: ISODate('2013-01-24T00:00:00.000Z'),
        grade: 'A',
        score: 10
      },
      {
        date: ISODate('2011-11-23T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2011-03-10T00:00:00.000Z'),
        grade: 'B',
        score: 14
      }
    ],
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  },
  {
    restaurant_id: '30075445'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff5'),
    address: {
      building: '7007',
      coord: [ 73.856743, 18.52043 ],
      street: 'Pune-Nashik Highway',
      zipcode: '411001',
      borough: 'Pune'
    },
    cuisine: 'Maharashtrian',
    grades: [
      {
        date: ISODate('2022-05-20T00:00:00.000Z'),
        grade: 'A',
        score: 9
      },
      {
        date: ISODate('2021-01-15T00:00:00.000Z'),
        grade: 'B',
        score: 7
```

```
},
{
  _id: ObjectId('65e50ec65b532e7900b71ff5'),
  address: {
    building: '7007',
    coord: [ 73.856743, 18.52043 ],
    street: 'Pune-Nashik Highway',
    zipcode: '411001',
    borough: 'Pune'
  },
  cuisine: 'Maharashtrian',
  grades: [
    {
      date: ISODate('2022-05-20T00:00:00.000Z'),
      grade: 'A',
      score: 9
    },
    {
      date: ISODate('2021-01-15T00:00:00.000Z'),
      grade: 'B',
      score: 7
    },
    {
      date: ISODate('2020-08-10T00:00:00.000Z'),
      grade: 'A',
      score: 11
    },
    {
      date: ISODate('2019-04-25T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2018-10-12T00:00:00.000Z'),
      grade: 'A',
      score: 10
    }
  ],
  name: 'Misal Junction',
  restaurant_id: '90005432'
},
{
  _id: ObjectId('65e56db05b532e7900b71ff0'),
  address: {
    building: '2001',
    coord: [ -74.123456, 40.789012 ],
    street: 'Broadway',
    zipcode: '10001'
  },
  borough: 'Manhattan',
  cuisine: 'Italian',
  grades: [
    { date: { '$date': 1420070400000 }, grade: 'A', score: 8 },
    { date: { '$date': 1396358400000 }, grade: 'B', score: 7 },
    { date: { '$date': 1372646400000 }, grade: 'A', score: 12 },
    { date: { '$date': 1348924800000 }, grade: 'A', score: 9 },
    { date: { '$date': 1325203200000 }, grade: 'C', score: 5 }
  ],
  name: 'Italian Delight',
  restaurant_id: '40098765'
}
```

3) db.Restraunt.find({ "grades.score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1, restaurant_id: 1 });

```
Atlas atlas-wqilky-shard-0 [primary] test> db.Restraunt.find(
...   { "grades.score": { $lte: 10 } },
...   { _id: 1, name: 1, town: 1, cuisine: 1, restaurant_id: 1 }
... );
[
  {
    _id: ObjectId('65e56db05b532e7900b71fef'),
    cuisine: 'Bakery',
    name: 'Morris Park Bake Shop',
    restaurant_id: '30075445'
  },
  {
    _id: ObjectId('65e56db05b532e7900b71ff0'),
    cuisine: 'Italian',
    name: 'Italian Delight',
    restaurant_id: '40098765'
  },
  {
    _id: ObjectId('65e56db05b532e7900b71ff1'),
    cuisine: 'Mexican',
    name: 'Sizzling Tacos',
    restaurant_id: '50065432'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff2'),
    cuisine: 'Indian',
    name: 'Spice Delight',
    restaurant_id: '60098765'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff3'),
    cuisine: 'Punjabi',
    name: 'Pind Flavors',
    restaurant_id: '70087654'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff4'),
    cuisine: 'Kannadiga',
    name: 'Namma Oota',
    restaurant_id: '80076543'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff5'),
    cuisine: 'Maharashtrian',
    name: 'Misal Junction',
    restaurant_id: '90065432'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff6'),
    cuisine: 'Maharashtrian',
    name: 'Vyanjan Vihar',
    restaurant_id: '90065432'
  },
  {
    _id: ObjectId('65e56ec65b532e7900b71ff7'),
    cuisine: 'Tamil',
    name: 'Tamil Delicacies',
    restaurant_id: '11076543'
  }
]
```

4)db.Restraunt.aggregate([{ $unwind: "$grades" }, { $group: { _id: "$restaurant_id", 62 name: { $first: "$name" }, averageScore: { $avg: "$grades.score" } } }, { $project: { _id: 1, name: 1, averageScore: 1 } }])

```
Atlas atlas-wqilky-shard-0 [primary] test> db.Restraunt.aggregate([
...   {
...     $unwind: "$grades"
...   },
...   {
...     $group: {
...       _id: "$restaurant_id",
...       name: { $first: "$name" },
...       averageScore: { $avg: "$grades.score" }
...     }
...   },
...   {
...     $project: {
...       _id: 1,
...       name: 1,
...       averageScore: 1
...     }
...   }
... ]);
[
  { _id: '30075445', name: 'Morris Park Bake Shop', averageScore: 8.2 },
  { _id: '50065432', name: 'Sizzling Tacos', averageScore: 8.8 },
  { _id: '70087654', name: 'Pind Flavors', averageScore: 9 },
  { _id: '80076543', name: 'Namma Oota', averageScore: 9 },
  { _id: '60098765', name: 'Spice Delight', averageScore: 8.8 },
  { _id: '40098765', name: 'Italian Delight', averageScore: 8.2 },
  { _id: '90065432', name: 'Misal Junction', averageScore: 9.1 },
  { _id: '11076543', name: 'Tamil Delicacies', averageScore: 8.8 }
```

5) db.Restraunt.find({ "address.zipcode": { $regex: /^10/ } }, { _id: 0, name: 1, "address.street": 1, "address.zipcode": 1 });

```
Atlas atlas-wqilky-shard-0 [primary] test> db.Restraunt.find(
...   { "address.zipcode": { $regex: /^10/ } },
...   { _id: 0, name: 1, "address.street": 1, "address.zipcode": 1 }
... );
[
  {
    address: { street: 'Broadway', zipcode: '10001' },
    name: 'Italian Delight'
  }
]
```