VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

Database Management Systems (23CS3PCDBM)

Submitted by

Gopika Pushparajan (1BM23CS101)

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 **Gopika Pushparajan (1BM23CS101),** 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 2024-2025. 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.

M Lakshmi Neelima Assistant Professor Department of CSE, BMSCE	Dr. Kavitha Sooda Professor & HOD Department of CSE, BMSCE

Index

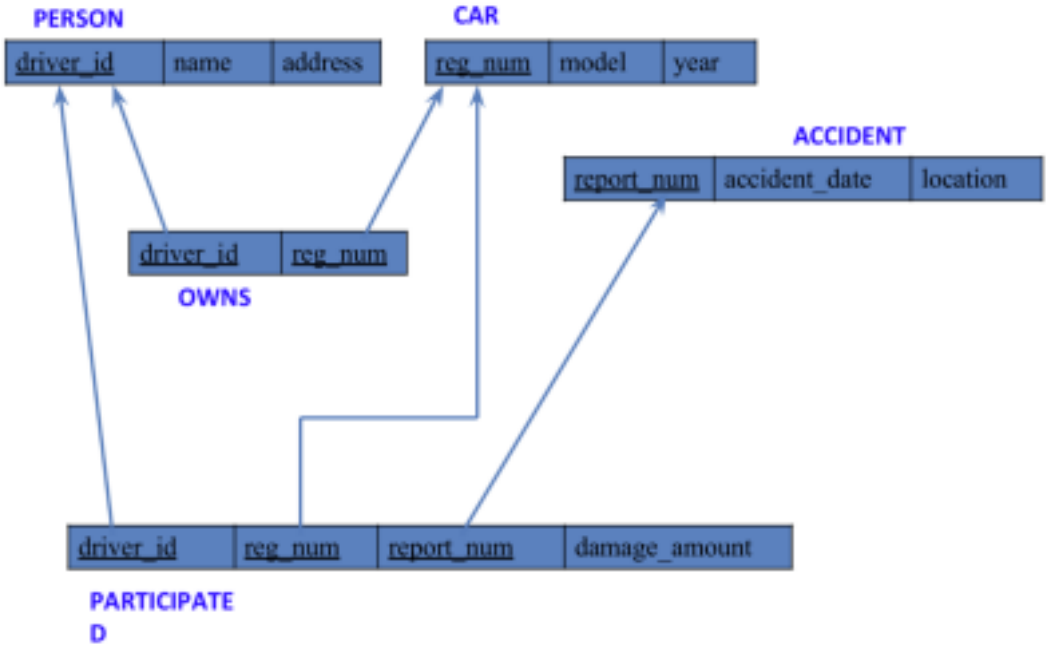
Insurance Database

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

```
create database insurance_101;
use insurance_101;
```

Create table

```
create table person_101(
driver_id varchar(20),
name varchar (30),
address varchar(50),
primary key(driver_id)
);


create table car_101(
reg_num varchar(10),
model varchar(15),
year int,
primary key(reg_num)
```

```
);
create table owns_101(
driver_id varchar(20),
reg_num varchar(10),
primary key(driver_id,reg_num),
foreign key(driver_id) references person_101(driver_id),
foreign key(reg_num) references car_101(reg_num)
);

create table accident_101(
report_num int,
accident_date date,
location varchar(50),
primary key(report_num)
);

create table participitated_101(
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_101(driver_id),
foreign key(reg_num) references car_101(reg_num),
foreign key(report_num) references accident_101(report_num)
);
```

Structure of the table

desc person_101;

Field	Type	Null	Key	Default	Extra
driver_id	varchar(20)	NO	PRI	NULL	
name	varchar(30)	YES		NULL	
address	varchar(50)	YES		NULL	

desc accident_101;

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

desc participitated_101;

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_101;

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

desc owns_101;

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_101 values("A01","Richard","Srinivas nagar");
insert into person_101 values("A02","Pradeep","Rajaji nagar");
insert into person_101 values("A03","Smith","Ashok nagar");
insert into person_101 values("A04","Venu","N R Colony");
insert into person_101 values("A05","John","Hanumanth nagar");
select * from person_101;
```

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

```
insert into car_101 values("KA052250","Indica",1990);
insert into car_101 values("KA031181","Lancer",1957);
insert into car_101 values("KA095477","Toyota",1998);
insert into car_101 values("KA053408","Honda",2008);
insert into car_101 values("KA041702","Audi",2005);
select * from car_101;
```

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_101 values("A01","KA052250");
insert into owns_101 values("A02","KA031181");
insert into owns_101 values("A03","KA095477");
insert into owns_101 values("A04","KA053408");
insert into owns_101 values("A05","KA041702");
```

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

insert into accident_101 values(11,"2003-01-01","MysoreRoad");

insert into accident_101 values(12,"2004-02-02","SouthEndCircle");

insert into accident_101 values(13,"2003-01-21","BullTempleRoad");

insert into accident_101 values(14,"2008-02-17","MysoreRoad");

insert into accident_101 values(15,"2003-01-01","KanakpuraRoad");

select * from accident_101;

	report_num	accident_date	location
▶	11	2003-01-01	MysoreRoad
	12	2004-02-02	SouthEndCircle
	13	2003-01-21	BullTempleRoad
	14	2008-02-17	MysoreRoad
	15	2003-01-01	KanakpuraRoad

insert into participitated_101 values("A01","KA052250",11,10000);
insert into participitated_101 values("A02","KA031181",12,50000);
insert into participitated_101 values("A03","KA095477",13,25000);
insert into participitated_101 values("A04","KA053408",14,3000);
insert into participitated_101 values("A05","KA041702",15,5000);
select * from participitated_101;

	driver_id	reg_num	report_num	damage_amount
▶	A01	KA052250	11	10000
	A02	KA031181	12	50000
	A03	KA095477	13	25000
	A04	KA053408	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 participitated_101

set damage_amount=25000

where reg_num='KA031181' and report_num=12;

commit;

select * from participitated_101;
```

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

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

```
select count(distinct driver_id) CNT
from participitated_101, accident_101
where participitated_101.report_num=accident_101.report_num and accident_101.accident_date like
'__08%';
```

CNT
1

Add a new accident to the database.

```
insert into accident values(16,'2008-03-08',"Domlur");
select * from accident_101;
```

report_num	accident_date	location
11	2003-01-01	MysoreRoad
12	2004-02-02	SouthEndCircle
13	2003-01-21	BullTempleRoad
14	2008-02-17	MysoreRoad
15	2003-01-01	KanakpuraRoad
16	2008-03-15	Dolmur
NULL	NULL	NULL

TO DO:

● Display accident date and location

	accident_date	location
▶	2003-01-01	MysoreRoad
	2004-02-02	SouthEndCircle
	2003-01-21	BullTempleRoad
	2008-02-17	MysoreRoad
	2003-01-01	KanakpuraRoad
	2008-03-15	Dolmur

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

```
select driver_id
from participitated_101
where damage_amount>=25000;
```

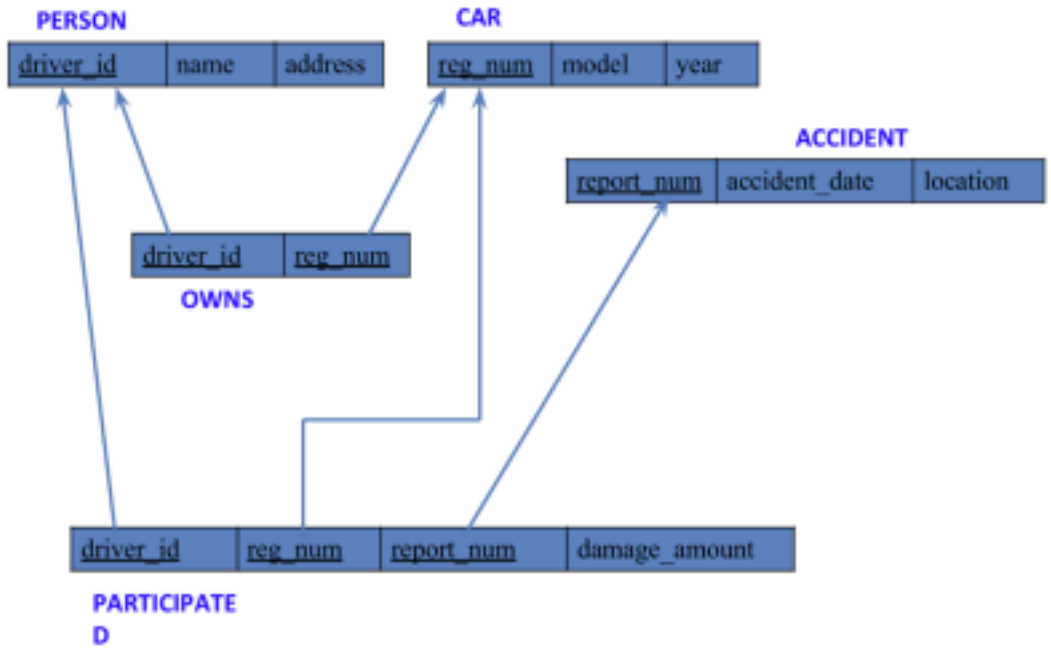
	driver_id
▶	A02
	A03

MORE QUERIES ON INSURANCE DATABASE

Question
(Week 2)

- PERSON (driver_id: String, name: String, address: String)
- CAR (reg_num: String, model: String, year: int)
- ACCIDENT (report_num: int, accident_date: date, location: String)
- OWNS (driver_id: String, reg_num: String)
- PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)
- 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

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

```
select * from car_101
order by year asc;
```

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

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

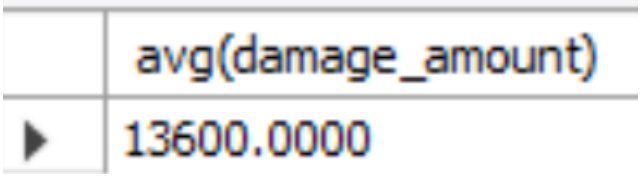
```
select count(report_num)
from car_101, participitated_101
where car_101.reg_num=participitated_101.reg_num and car_101.model='Lancer';
```

count(report_num)
1

TO DO:

- **Find the average damage amount**

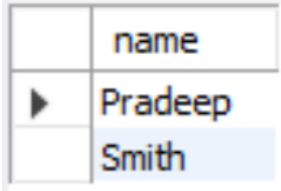
select avg(damage_amount) from participitated_101;



	avg(damage_amount)
▶	13600.0000

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

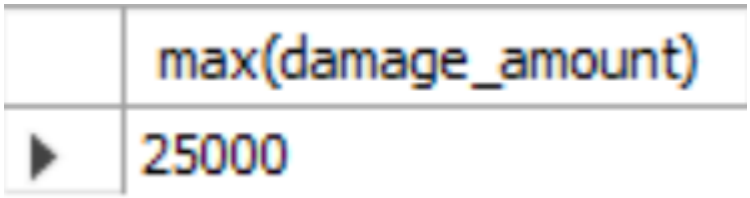
select name
from person_101, participitated_101
where person_101.driver_id = participitated_101.driver_id and participitated_101.damage_amt > (select avg(damage_amount) from participitated_101);



	name
▶	Pradeep
	Smith

- **Find maximum damage amount.**

select max(damage_amount) from participitated_101;



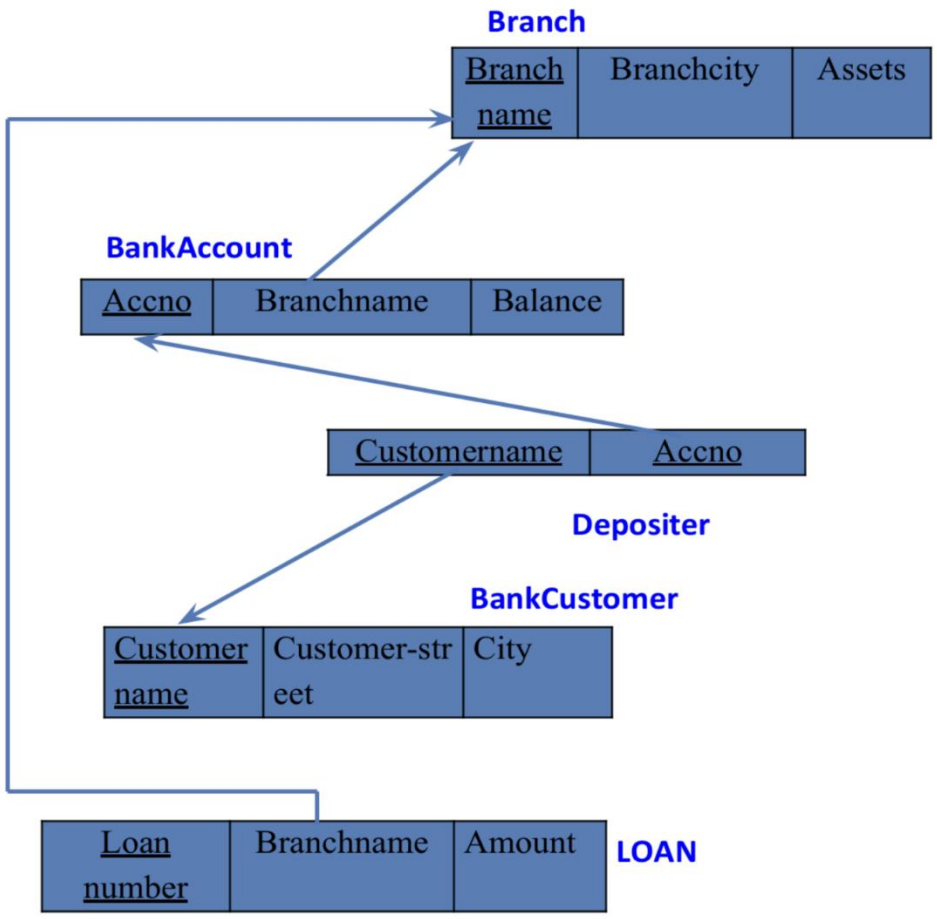
	max(damage_amount)
▶	25000

Bank Database

Question
(Week 3)

- Branch (branch-name: String, branch-city: String, assets: real)
- BankAccount(accno: int, branch-name: String, balance: real)
- BankCustomer (customer-name: String, customer-street: String, customer-city: String) Depositer(customer-name: String, accno: int)
- LOAN (loan-number: int, branch-name: String, amount: real)
- Create the above tables by properly specifying the primary keys and the foreign keys. - Enter at least five tuples for each relation.
- Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
- Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).
- Create a view which gives each branch the sum of the amount of all the loans at the branch.

Schema Diagram



Create database

```
create database bank_101;
USE bank_101;
```

Create table

```
create table branch_101(
branch_name varchar(20),
branch_city varchar(20),
assets int,
primary key(branch_name));

create table bankacc_101(
accno int,
branch_name varchar(20),
balance int,
primary key(accno),
foreign key(branch_name) references branch_101(branch_name));

create table bankcustomer_101(
customer_name varchar(20),
customer_street varchar(20),
city varchar(20),
primary key(customer_name));

create table depositor_101(
customer_name varchar(20),
accno int,
foreign key(customer_name) references bankcustomer_101(customer_name),
foreign key(accno) references bankacc_101(accno));

create table loan_101(
loan_number int,
branch_name varchar(20),
amount int,
primary key(loan_number),
foreign key (branch_name) references branch_101(branch_name));
```

Structure of tables

desc branch_101;

Field	Type	Null	Key	Default	Extra
branch_name	varchar(20)	NO	PRI	NULL	
branch_city	varchar(20)	YES		NULL	
assets	int	YES		NULL	

desc bankacc_101;

Field	Type	Null	Key	Default	Extra
accno	int	NO	PRI	NULL	
branch_name	varchar(20)	YES	MUL	NULL	
balance	int	YES		NULL	

desc bankcustomer_101;

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	NO	PRI	NULL	
customer_street	varchar(20)	YES		NULL	
city	varchar(20)	YES		NULL	

desc depositor_101;

Field	Type	Null	Key	Default	Extra
customer_name	varchar(20)	YES	MUL	NULL	
accno	int	YES	MUL	NULL	

desc loan_101;

Field	Type	Null	Key	Default	Extra
loan_number	int	NO	PRI	NULL	
branch_name	varchar(20)	YES	MUL	NULL	
amount	int	YES		NULL	

Inserting Values to the table

```
insert into branch_101 values("SBI_Chamarajpet","Bangalore",50000);
insert into branch_101 values("SBI_ResidencyRoad","Bangalore",10000);
insert into branch_101 values("SBI_ShivajiRoad","Bombay",20000);
insert into branch_101 values("SBI_ParliamentRoad","Delhi",10000);
insert into branch_101 values("SBI_Jantarmantar","Delhi",20000);

insert into bankacc_101 values(1,"SBI_Chamarajpet",2000);
insert into bankacc_101 values(2,"SBI_ResidencyRoad",5000);
insert into bankacc_101 values(3,"SBI_ShivajiRoad",6000);
insert into bankacc_101 values(4,"SBI_ParliamentRoad",9000);
insert into bankacc_101 values(5,"SBI_Jantarmantar",8000);
insert into bankacc_101 values(6,"SBI_ShivajiRoad",4000);
insert into bankacc_101 values(8,"SBI_ResidencyRoad",4000);
insert into bankacc_101 values(9,"SBI_ParliamentRoad",3000);
insert into bankacc_101 values(10,"SBI_ResidencyRoad",5000);
insert into bankacc_101 values(11,"SBI_Jantarmantar",2000);

insert into bankcustomer_101 values("Avinash","Bull_temple_road","Bangalore");
insert into bankcustomer_101 values("Dinesh","Bannergatta_Road","Bangalore");
insert into bankcustomer_101 values("Mohan","NationalCollege_Road","Bangalore");
insert into bankcustomer_101 values("Nikil","Akbar_Road","Delhi");
insert into bankcustomer_101 values("Ravi","Prithviraj_Road","Delhi");

insert into depositor_101 values("Avinash",1);
insert into depositor_101 values("Dinesh",2);
insert into depositor_101 values("Nikil",4);
insert into depositor_101 values("Ravi",5);
insert into depositor_101 values("Avinash",8);
insert into depositor_101 values("Nikil",9);
insert into depositor_101 values("Dinesh",10);
insert into depositor_101 values("Nikil",11);

insert into loan_101 values(1,"SBI_Chamarajpet",1000);
insert into loan_101 values(2,"SBI_ResidencyRoad",2000);
insert into loan_101 values(3,"SBI_ShivajiRoad",3000);
insert into loan_101 values(4,"SBI_ParliamentRoad",4000);
insert into loan_101 values(5,"SBI_Jantarmantar",5000);
```

```
select * from branch_101;
select * from depositor_101;
select * from loan_101;
select * from bankcustomer_101;
select * from bankacc_101;
```

branch_name	branch_city	assets
SBI_Chamarajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParliamentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

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

accno	branch_name	balance
1	SBI_Chamarajpet	2000
2	SBI_ResidencyRoad	5000
3	SBI_ShivajiRoad	6000
4	SBI_ParliamentRoad	9000
5	SBI_Jantarmantar	8000
6	SBI_ShivajiRoad	4000
8	SBI_ResidencyRoad	4000
9	SBI_ParliamentRoad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000
NULL	NULL	NULL

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

customer_name	customer_street	city
Avinash	Bull_temple_road	Bangalore
Dinesh	Bannergatta_Road	Bangalore
Mohan	NationalCollege_Road	Bangalore
Nikil	Akbar_Road	Delhi
Ravi	Prithviraj_Road	Delhi
NULL	NULL	NULL

Queries

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

```
select branch_name,assets/100000 as assets_in_lakhs from branch_101;
```

branch_name	assets_in_lakhs
SBI_Chamarajpet	0.5000
SBI_Jantarmantar	0.2000
SBI_ParliamentRoad	0.1000
SBI_ResidencyRoad	0.1000
SBI_ShivajiRoad	0.2000

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

```
select customer_name,count(*)
from depositor_101,bankacc_101
where bankacc_101.accno=depositor_101.accno
and branch_name="SBI_ResidencyRoad"
group by customer_name having count(*)>1;
```

customer_name	count(*)
Dinesh	2

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

```
create view loansum as (
select branch_name, sum(amout)
from loans_101
group by branch_name
);
select * from loansum;
```

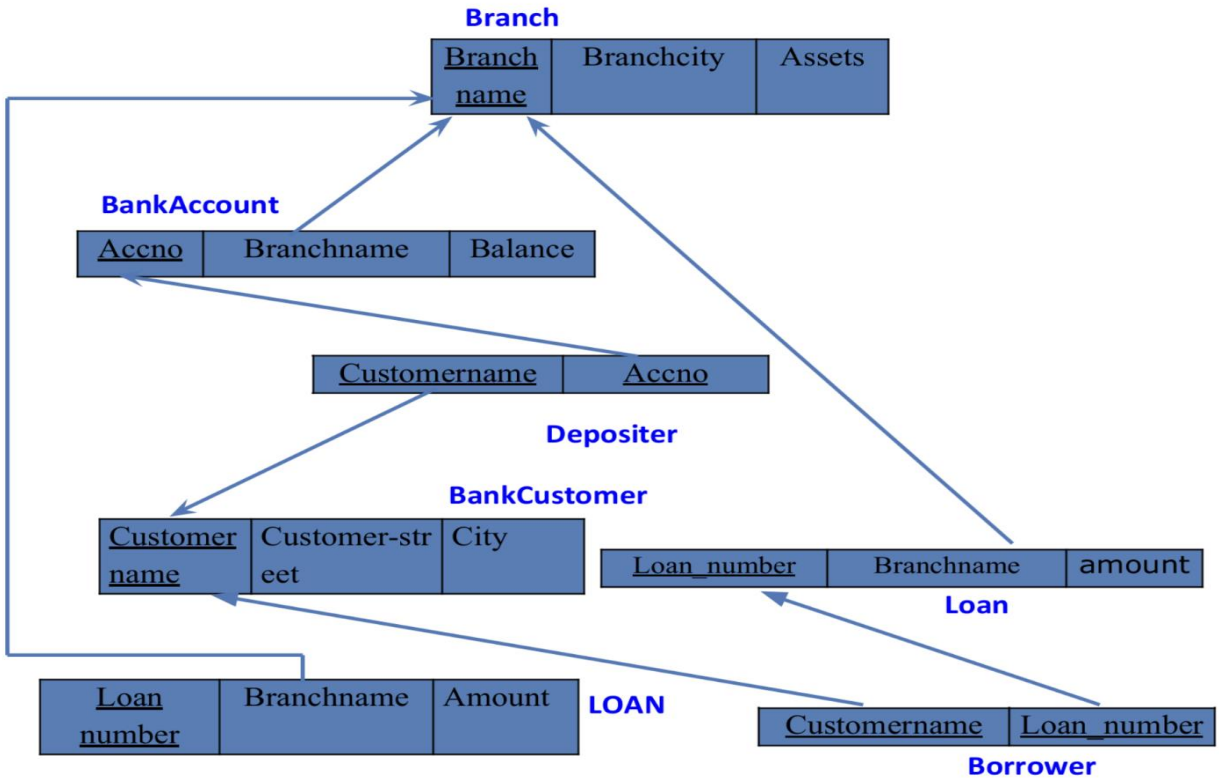
branch_name	sum(amount)
SBI_Chamarajpet	1000
SBI_Jantarmantar	5000
SBI_ParliamentRoad	4000
SBI_ResidencyRoad	2000
SBI_ShivajiRoad	3000

More Queries on Bank Database

Question
(Week 4)

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

Schema Diagram



Creating Table:

```
create table borrower_101(
customer_name varchar(20),
loan_number int,
foreign key(customer_name) references bankcustomer_101(customer_name),
foreign key(loan_number) references loan_101(loan_number)
);
```

Inserting values:

```
insert into branch_101 values ("SBI_MantriMarg", "Delhi", 200000);
insert into bankacc_101 values (12, "SBI_MantriMarg", 2000);
insert into depositor_101 values("Nikil", 12);
insert into borrower_101 values ("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 customer_name
from bankcustomer_101
where city="Delhi";
```

customer_name
Nikil
Ravi
NULL

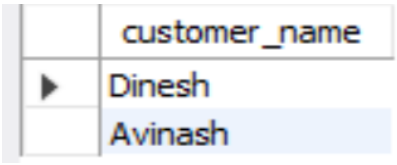
- **Find all customers who have a loan at the bank but do not have an account.**

```
select customer_name, loan_101.loan_number
from (borrower_101 right outer join loan_101
on loan_101.loan_number = borrower_101.loan_number)
where customer_name not in (select customer_name
from depositor_101, bankacc_101 where depositor_101.accno = bankacc_101.accno
group by customer_name, branch_name);
```

customer_name	loan_number
Mohan	3

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

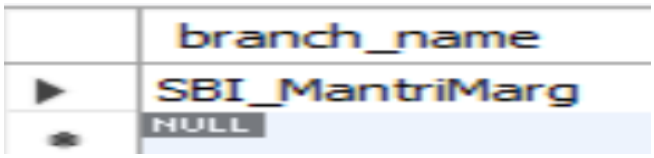
  select distinct customer_name from depositor_101
  where customer_name in (select depositor_101.customer_name from branch_101, bankacc_101, depositor_101
  where branch_101.branch_city = "Bangalore" and branch_101.branch_name = bankacc_101.branch_name and bankacc_101.accno = depositor_101.accno) and customer_name in (select customer_name from borrower_101, loan_101 where branch_name in (select branch_name from branch_101 where branch_city = "Bangalore"));



customer_name
Dinesh
Avinash

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

  select branch_name from branch_101
  where assets > all(select assets
  from branch_101 where branch_city = "Bangalore");



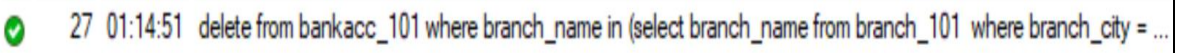
branch_name
SBI_MantriMarg
NULL

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

  update bankacc_101 set balance = 1.05*balance;

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

  delete from bankacc_101
  where branch_name in (select branch_name
  from branch_101
  where branch_city = "Bombay");

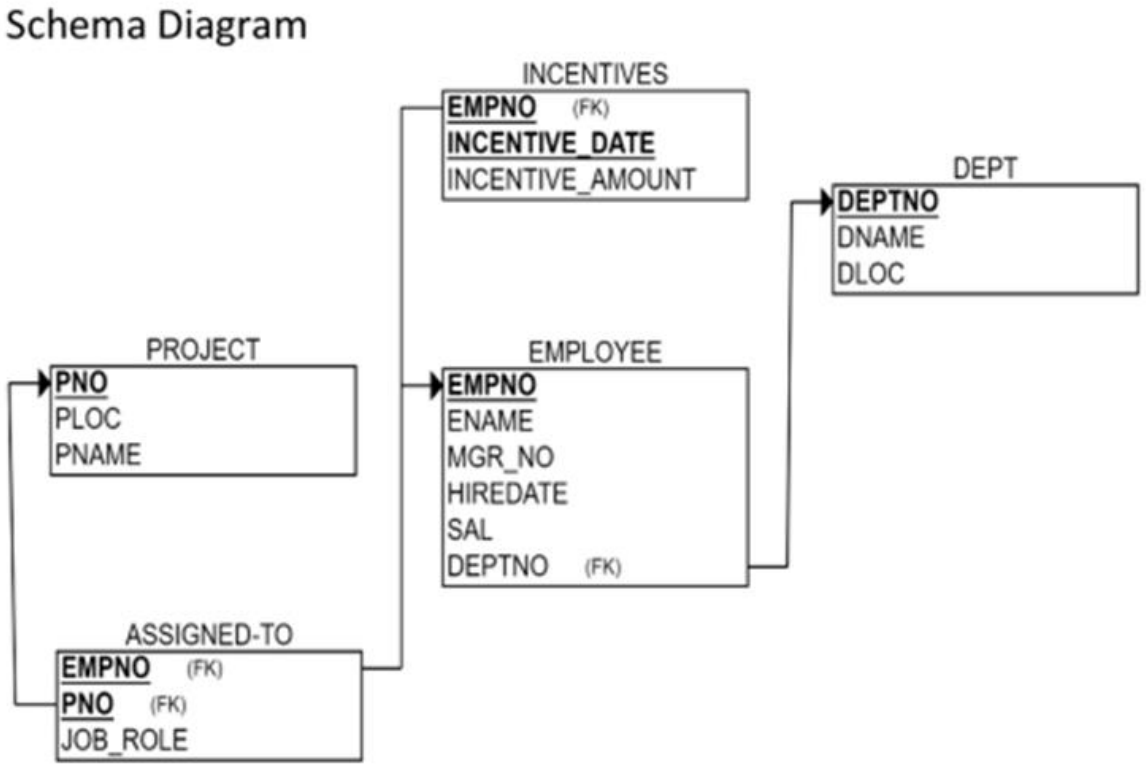


Employee Database

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_database_101;
use employee_database_101;
```

Create table

```
create table project_101(
    pno int primary key,
    ploc varchar(20),
    pname varchar(20));

create table dept_101(
    deptno int primary key,
    dname varchar(30),
    dloc varchar(20));

create table employee_101(
    empno int primary key,
    ename varchar(20),
    mgr_no int,
    hiredate date,
    sal double,
    deptno int,
    foreign key(deptno) references dept_101(deptno));

create table assigned_to_101(
    empno int primary key,
    pno int,
    job_role varchar(20),
    foreign key(empno) references employee_101(empno),
    foreign key(pno) references project_101(pno));

create table incentives_101(
    empno int,
    incentive_date date primary key,
    incentive_amount double,
    foreign key(empno) references employee_101(empno));
```

Structure of the table

desc project_101;

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

desc dept_101;

Field	Type	Null	Key	Default	Extra
deptno	int	NO	PRI	NULL	
dname	varchar(30)	YES		NULL	
dloc	varchar(20)	YES		NULL	

desc employee_101;

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

desc assigned_to_101;

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

desc incentives_101;

Field	Type	Null	Key	Default	Extra
empno	int	YES	MUL	NULL	
incentive_date	date	NO	PRI	NULL	
incentive_amount	double	YES		NULL	

Inserting Values to the table

insert into project_101 values (1,"bengaluru","abcd"), (2,"hyderabad","bcda"), (3,"bengaluru","abab"), (4,"bengaluru","baba"), (5,"hyderabad","cdcd"), (6, "mysuru","efef");
select * from project_101;

pno	ploc	pname
1	bengaluru	abcd
2	hyderabad	bcda
3	bengaluru	abab
4	bengaluru	baba
5	hyderabad	cdcd
6	mysuru	efef
NULL	NULL	NULL

insert into dept_101 values (1,"cse","bengaluru"), (2,"ise","hyderabad"), (3,"ece","bengaluru"), (4,"ete","hyderabad"), (5,"ime","bengaluru"), (6, "mech", "mysuru");
select * from dept_101;

deptno	dname	dloc
1	cse	bengaluru
2	ise	hyderabad
3	ece	bengaluru
4	ete	hyderabad
5	ime	bengaluru
6	mech	mysuru

insert into employee_101 values (1,"a",null,"2023-11-9",70000,1), (2,"b",2,"2023-8-9",70000,1), (3,"c",3,"2023-6-8",70000,2), (4,"d",null,"2023-8-6",70000,2), (5,"e",null,"2023-5-4",70000,3), (6, "f", null, "2023-6-1", 90000, 6);
select * from employee_101;

empno	ename	mgr_no	hiredate	sal	deptno
1	a	NULL	2023-11-09	70000	1
2	b	2	2023-08-09	70000	1
3	c	3	2023-06-08	70000	2
4	d	NULL	2023-08-06	70000	2
5	e	NULL	2023-05-04	70000	3
6	f	NULL	2023-06-01	90000	6
NULL	NULL	NULL	NULL	NULL	NULL

insert into incentives_101 values (1,"2023-12-9",10000), (2,"2023-8-9",10000), (3,"2023-6-8",10000), (4,"2023-5-4",10000), (5,"2023-12-8",10000);
select * from incentives_101;

empno	incentive_date	incentive_amount
4	2023-05-04	10000
3	2023-06-08	10000
2	2023-08-09	10000
5	2023-12-08	10000
1	2023-12-09	10000
NULL	NULL	NULL

insert into assigned_to_101 values (1,1, "employee"), (2,1, "manager"), (3,2, "manager"), (4,3, "employee"), (5,4, "employee"), (6, 6, "employee");
select * from assigned_to_101;

empno	pno	job_role
1	1	employee
2	1	manager
3	2	manager
4	3	employee
5	4	employee
6	6	employee
NULL	NULL	NULL

Queries

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

```
select assigned_to_101.empno
from assigned_to_101, project_101
where assigned_to_101.pno = project_101.pno and project_101.ploc in ("bengaluru", "mysuru",
"hyderabad");
```

	empno
▶	1
	2
	3
	4
	5
	6

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

```
select empno
from employee_101
where empno not in (select empno from incentives_101);
```

	empno
▶	6
*	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 employee_101.empno, ename, dname, job_role, dloc, ploc
from employee_101, assigned_to_101, project_101, dept_101
where ploc = dloc and assigned_to_101.empno = employee_101.empno and employee_101.deptno =
dept_101.deptno and project_101.pno = assigned_to_101.pno;
```

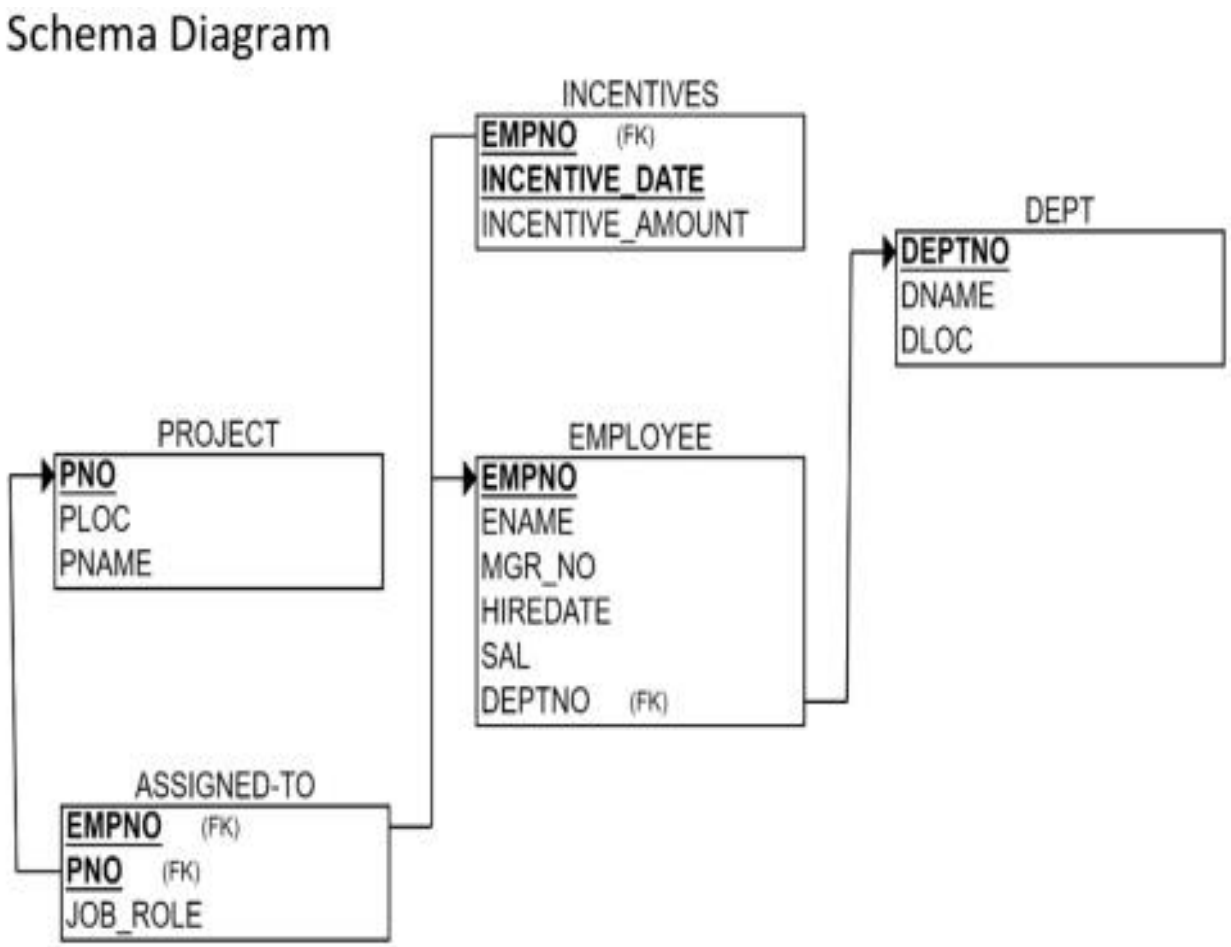
	empno	ename	dname	job_role	dloc	ploc
▶	1	a	cse	employee	bengaluru	bengaluru
	2	b	cse	manager	bengaluru	bengaluru
	3	c	ise	manager	hyderabad	hyderabad
	5	e	ece	employee	bengaluru	bengaluru
	6	f	mech	employee	mysuru	mysuru

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.

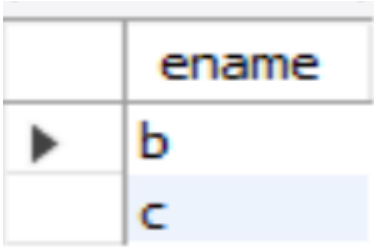
Schema Diagram



Queries

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

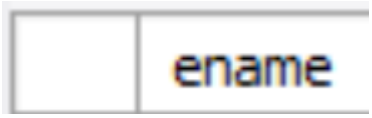
```
select e1.ename
from employee_101 e1, employee_101 e2
where e1.empno=e2.mgr_no group by e1.ename
having count(e1.mgr_no)=(select count(e1.ename)
from employee_101 e1, employee_101 e2 where e1.empno=e2.mgr_no
group by e1.ename order by count(e1.ename) desc limit 1);
```



	ename
▶	b
	c

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

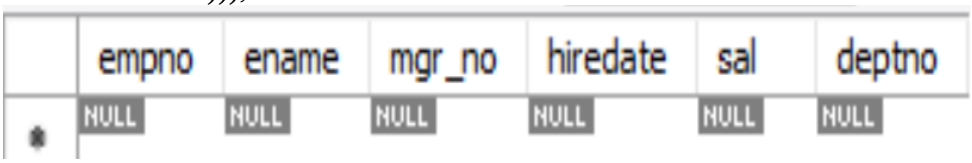
```
select m.ename from employee_101 m
where m.empno in
(select mgr_no from employee_101)
and m.sal>(select avg(n.sal) from employee_101 n
where n.mgr_no=m.empno);
```



	ename

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

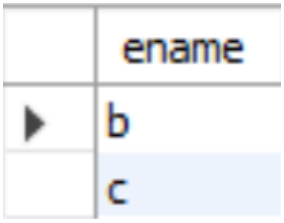
```
select * from employee_101 where empno=
(select i.empno from incentives_101 i
where i.incentive_amount= (select max(n.incentive_amount) from incentives_101 n
where n.incentive_amount < (select max(inc.incentive_amount) from incentives_101 inc
where inc.incentive_date between 2023-01-01 and 2023-12-31 and incentive_date between 2023-01-01
and 2023-12-31)));
```



	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 e2.ename
from employee_101 e1, employee_101 e2
where e1.empno=e2.mgr_no and e1.deptno=e2.deptno;
```



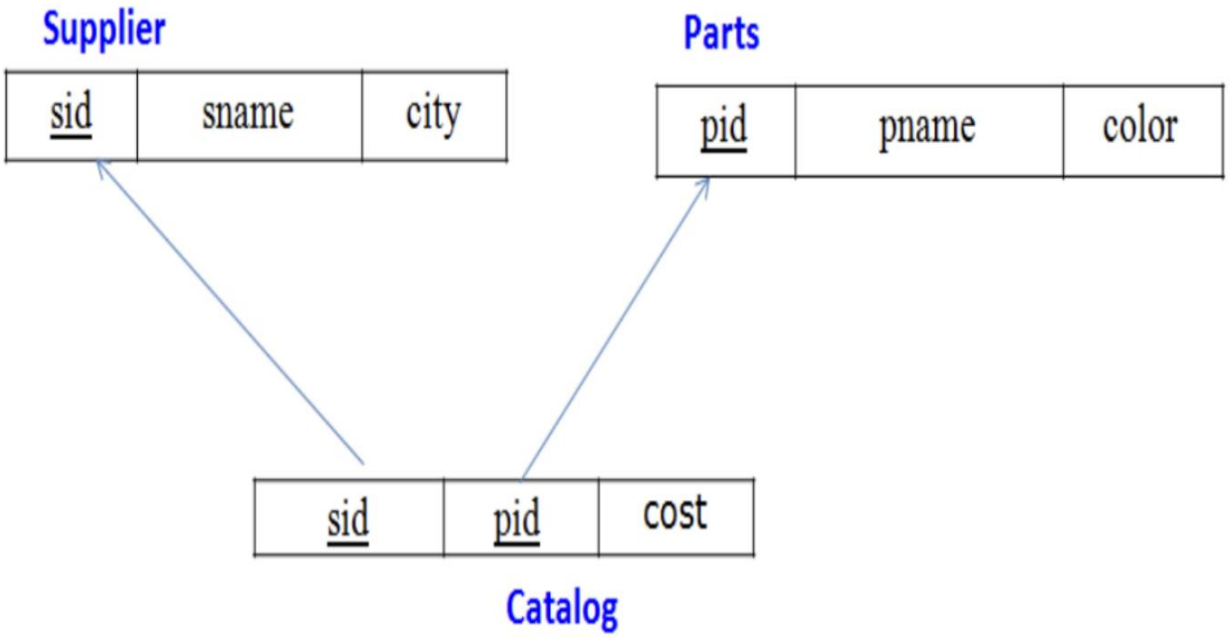
	ename
▶	b
	c

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 supply_101;
use supply_101;
```

Create table

```
create table supplier_101(
sid int primary key,
sname varchar(20),
city varchar(30)
);

create table parts_101(
pid int primary key,
pname varchar(20),
color varchar(20)
);
create table catalog_101(
sid int, pid int,
cost int,
foreign key(sid) references supplier_101(sid),
foreign key(pid) references parts_101(pid)
);
```

Structure of the table

desc supplier_101;

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

desc parts_101;

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

desc catalog_101;

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

Inserting Values to the table

insert into supplier_101 values (10001, "acne", "Bangalore"), (10002, "johns", "Kolkata"), (10003, "vimal", "Mumbai"), (10004, "reliance", "Delhi");

select * from supplier_101;

	sid	sname	city
▶	10001	acne	Bangalore
	10002	johns	Kolkata
	10003	vimal	Mumbai
	10004	reliance	Delhi
*	NULL	NULL	NULL

insert into parts_101 values (20001,"Book","Red"), (20002,"Pen","Red"), (20003, 'Pencil', 'Green'), (20004, 'Mobile', 'Green'), (20005, 'Charger', 'Black');

select * from parts_101;

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

insert into catalog_101 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);

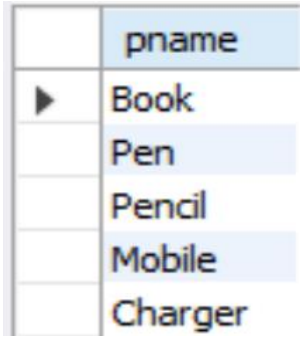
select * from catalog_101;

	sid	pid	cost
▶	10001	20001	10
	10001	20002	10
	10001	20003	30
	10001	20004	10
	10001	20005	10
	10002	20001	10
	10002	20002	20
	10003	20003	30
	10004	20003	40

Queries

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

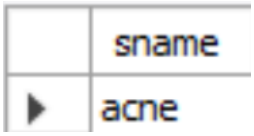
select pname from parts_101 where pid in (select pid from catalog_101);



	pname
▶	Book
	Pen
	Pencil
	Mobile
	Charger

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

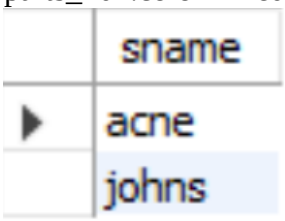
select sname from supplier_101 where sid in
(select sid from catalog_101 group by sid having count(distinct pid) = (select count(distinct pid) from parts_101));



	sname
▶	acne

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

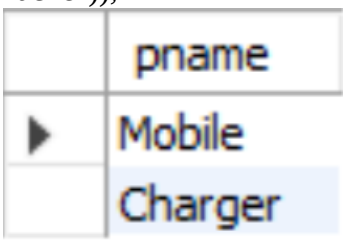
select distinct sname from supplier_101, parts_101, catalog_101
where supplier_101.sid = catalog_101.sid and parts_101.pid = catalog_101.pid and parts_101.color="Red";



	sname
▶	acne
	johns

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

select pname from parts_101 where pid not in
(select pid from catalog_101 where sid in (select sid from supplier_101 where sname != "acne"));



	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_101 a where a.cost > (select avg(b.cost) from catalog_101 b where a.pid = b.pid group by b.pid);

	sid
▶	10002
	10004

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

select pid, sname from catalog_101 a, supplier_101
where a.cost = (select max(b.cost) from catalog_101 b where a.pid = b.pid group by b.pid) and supplier_101.sid = a.sid;

	pid	sname
▶	20001	acne
	20004	acne
	20005	acne
	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.
5. Drop the table

Create database

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

Create table & Inserting Values to the table

```
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"});
```

```
C:\Users\STUDENT>mongosh "mongodb+srv://cluster0.zn7bx.mongodb.net/" --apiVersion 1 --username gopikapushparajan
Enter password: *********
Current Mongosh Log ID: 676511d9cf980bcf29d14a0d
Connecting to:          mongodb+srv://<credentials>@cluster0.zn7bx.mongodb.net/?appName=mongosh+2.2.1
Using MongoDB:          8.0.4 (API Version 1)
Using Mongosh:          2.2.1
mongosh 2.3.7 is available for download: https://www.mongodb.com/try/download/shell

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

Atlas atlas-lfioua-shard-0 [primary] test> use myDB;
switched to db myDB
Atlas atlas-lfioua-shard-0 [primary] myDB> db
myDB
Atlas atlas-lfioua-shard-0 [primary] myDB> db.createCollection("Student");
{ ok: 1 }
Atlas atlas-lfioua-shard-0 [primary] myDB> 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('676512b2cf980bcf29d14a0e') }
}
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.insert({RollNo:2,Age:22,Cont:9976,email:"anushka.de9@gmail.com"});

{
  acknowledged: true,
  insertedIds: { '0': ObjectId('676512ddcf980bcf29d14a0f') }
}
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.insert({RollNo:3,Age:21,Cont:5576,email:"anubhav.de9@gmail.com"});

{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6765136acf980bcf29d14a10') }
}
Atlas atlas-lfioua-shard-0 [primary] myDB>
(To exit, press Ctrl+C again or Ctrl+D or type .exit)
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.insert({RollNo:4,Age:20,Cont:4476,email:"pani.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('6765138bcf980bcf29d14a11') }
}
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.insert({RollNo:10,Age:23,Cont:2276,email:"rekha.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67651396cf980bcf29d14a12') }
}
```

db.Student.find();

```
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.find()
[
  {
    _id: ObjectId('676512b2cf980bcf29d14a0e'),
    RollNo: 1,
    Age: 21,
    Cont: 9876,
    email: 'antara.de9@gmail.com'
  },
  {
    _id: ObjectId('676512ddcf980bcf29d14a0f'),
    RollNo: 2,
    Age: 22,
    Cont: 9976,
    email: 'anushka.de9@gmail.com'
  },
  {
    _id: ObjectId('6765136acf980bcf29d14a10'),
    RollNo: 3,
    Age: 21,
    Cont: 5576,
    email: 'anubhav.de9@gmail.com'
  },
  {
    _id: ObjectId('6765138bcf980bcf29d14a11'),
    RollNo: 4,
    Age: 20,
    Cont: 4476,
    email: 'pani.de9@gmail.com'
  },
  {
    _id: ObjectId('67651396cf980bcf29d14a12'),
    RollNo: 10,
    Age: 23,
    Cont: 2276,
    email: 'rekha.de9@gmail.com'
  }
]
```

Queries

- **Write a query to update the Email-Id of a student with rollno 5.**

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

```
Atlas atlas-lfioua-shard-0 [primary] myDB> 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
}
```

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

db.Student.insert({rollno:11,age:22,name:"ABC",cont:2276,email:"madhura@gmail.com"});
db.Student.update({rollno:11,name:"ABC"},{$set:{name:"FEM"}})

```
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.insert({RollNo:11,Age:22,Name:"ABC",Cont:2276,email:"rea.de9@gmail.com"});
{
  acknowledged: true,
  insertedIds: { '0': ObjectId('67651421cf980bcf29d14a13') }
}
Atlas atlas-lfioua-shard-0 [primary] myDB> db.Student.update({RollNo:11,Name:"ABC"},{$set:{Name:"FEM"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

- **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()
```

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. Drop the table

Create Table:

db.createCollection("Customer");

```
test> db.createCollection("Customer");
{ ok: 1 }
```

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"}]);

```
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:10
000, acc_type: "Saving"}, {custid: 5, acc_bal:2000, acc_type: "Checking"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65e418fc5b3b1935aac1fe4b'),
    '1': ObjectId('65e418fc5b3b1935aac1fe4c'),
    '2': ObjectId('65e418fc5b3b1935aac1fe4d'),
    '3': ObjectId('65e418fc5b3b1935aac1fe4e'),
    '4': ObjectId('65e418fc5b3b1935aac1fe4f')
  }
}
```

Queries:

- **Finding all checking accounts with balance greater than 12000**

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

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

- **Finding the maximum and minimum balance of each customer**

  db.Customer.aggregate([{$group:{_id:"$custid", minBal:{$min:"$acc_bal"}, maxBal: {$max:"$acc_bal"}}}]);

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

- **Dropping collection "Customer"**

  db.Customer.drop();

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

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'.

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"},

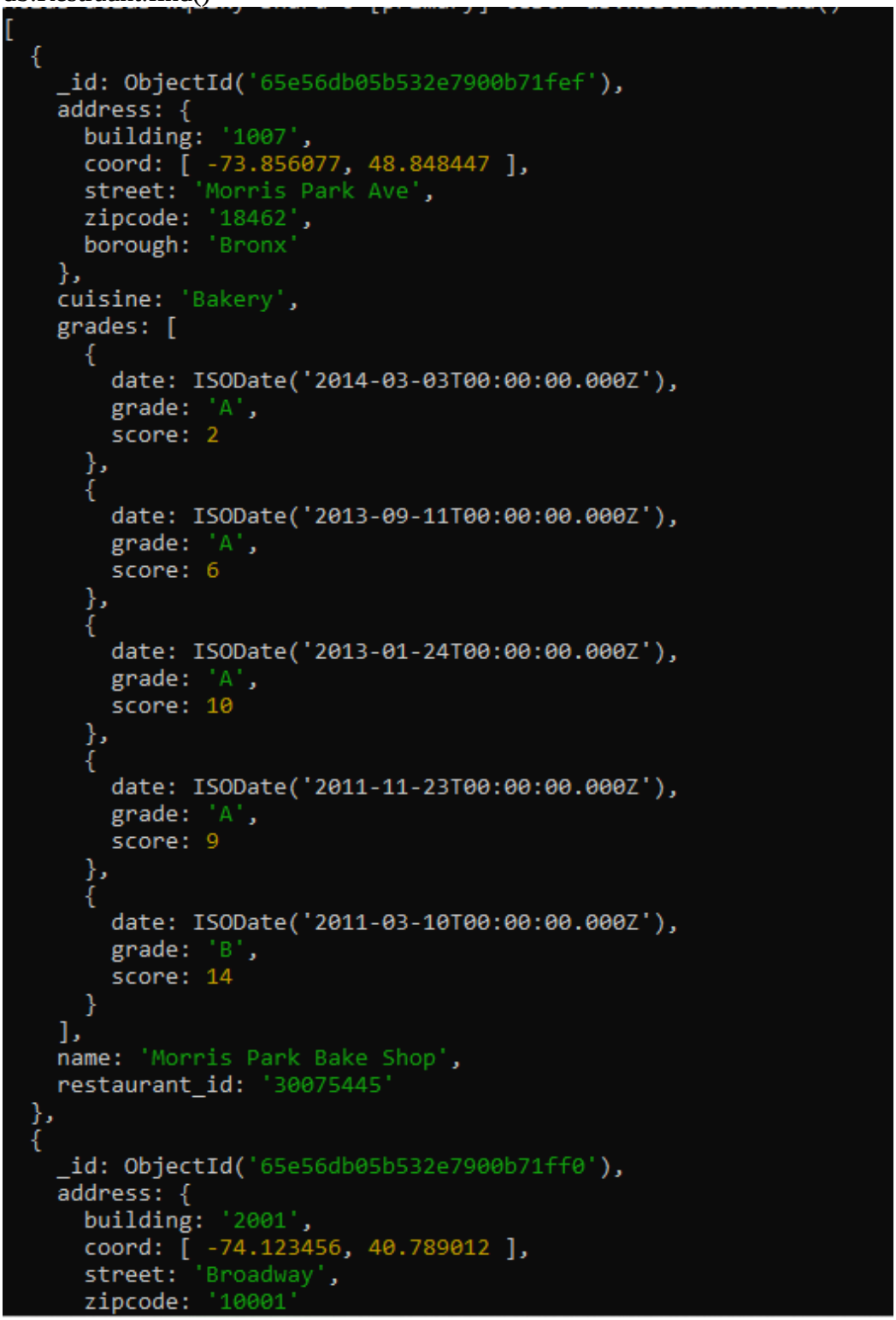
{"address": {"building": "7007","coord": [73.856743, 18.520430],"street": "Shivaji Road", "zipcode": "411001","borough": "Pune"},"cuisine": "Maharashtrian",
"grades": [
{"date": new Date("2022-04-30"), "grade": "A", "score": 9},
{"date": new Date("2021-10-15"), "grade": "B", "score": 7},
{"date": new 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',
  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'
},
{
  _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('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
      }
    ],
    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',
```

```
  },
  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('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
    }
  ],
  name: 'Sizzling Tacos',
  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
    },
    {
```

```
      grade: 'A',
      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
    }
  ],
  name: 'Namma Oota',
  restaurant_id: '80076543'
},
{
  _id: ObjectId('65e56db05b532e7900b71fef'),
  address: {
    building: '1007',
    coord: [ -73.856077, 48.848447 ],
    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: '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('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('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('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", 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'
  }
]
```