VISVESVARAYA TECHNOLOGICAL UNIVERSITY

“JnanaSangama”, Belgaum -590014, Karnataka.

LAB REPORT

On

Database Management Systems (23CS3PCDBM)

Submitted by

DISHA D S

(1BM23CS094)

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 **DISHA D S (1BM23CS094),** who is bonafide student of **B. M. S. College of Engineering.** It is in partial fulfillment for the award of **Bachelor of Engineering in Computer Science and Engineering** of the Visvesvaraya Technological University, Belgaum during the year 2022. The Lab report has been approved as it satisfies the academic requirements in respect of a Database Management Systems (23CS3PCDBM) work prescribed for the said degree.

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

Index

1. INSURANCE DATABASE

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)

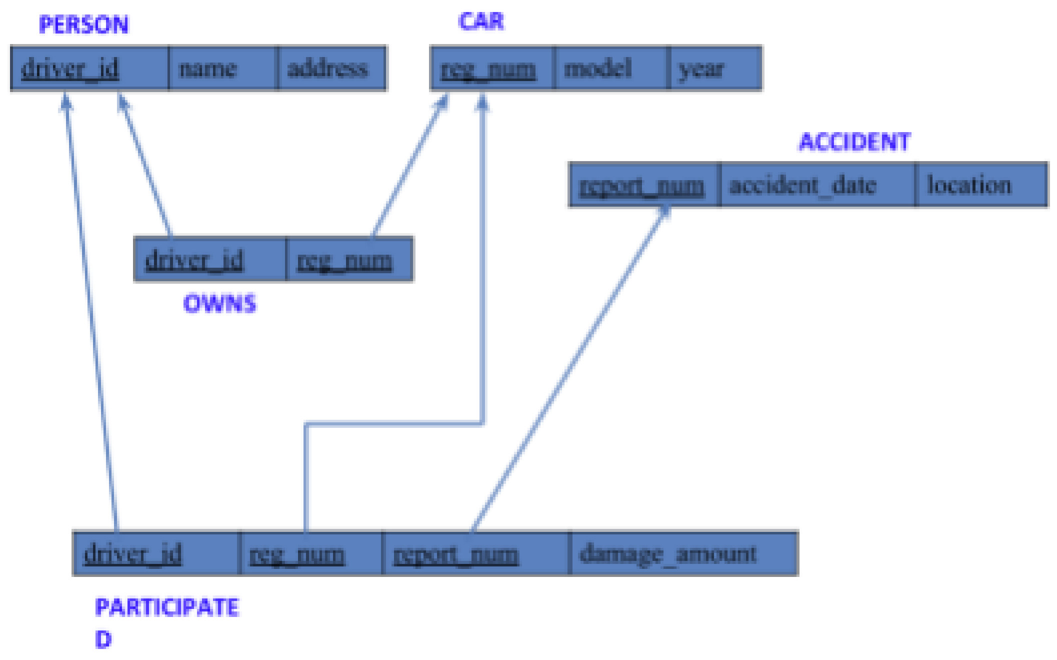
PARTICIPATED (driver_id: String,reg_num: String, report_num: int, damage_amount: int)

- 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:

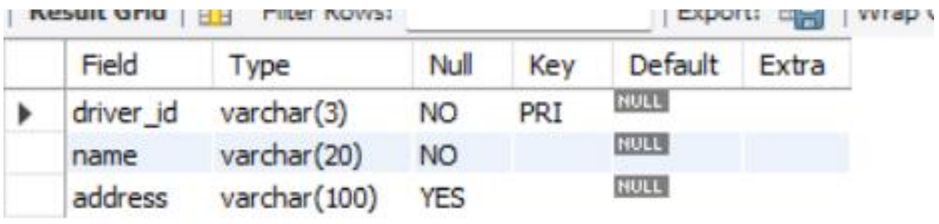
```
create database insurance_094;

use insurance_094;
```

Create table:

```
create database insurance_094;

use insurance_094;

create table person_094(

driver_id varchar(3) primary key,

name varchar(20) not null,

address varchar(100)

);

create table car_094(

reg_no char(8) primary key,

model varchar(20),

year int(4) not null

);

create table accident_094(

report_no int(4) primary key,

accident_date date,

location varchar(100)

);

create table owns_094(

driver_id varchar(3),

reg_no char(8),

foreign key(driver_id) references person_094(driver_id),

foreign key(reg_no) references car_094(reg_no)

);

create table participated_094(

driver_id varchar(3),

reg_no char(8),

report_no int(4),
damage_amt int,
foreign key(driver_id) references person_094(driver_id),
foreign key(reg_no) references car_094(reg_no),
foreign key (report_no) references accident_094(report_no)
);
```

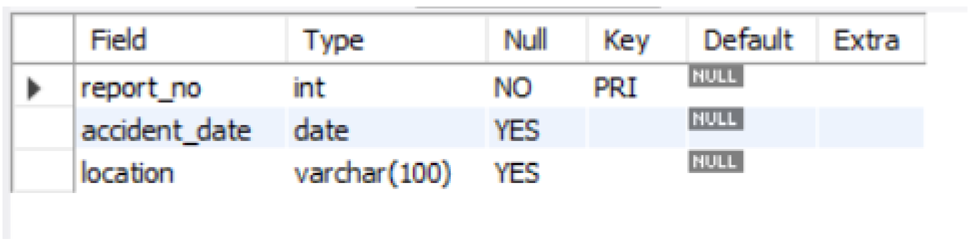
Structure of table:

desc person_094;


Result Grid
Filter Rows:
Export:
Wrap

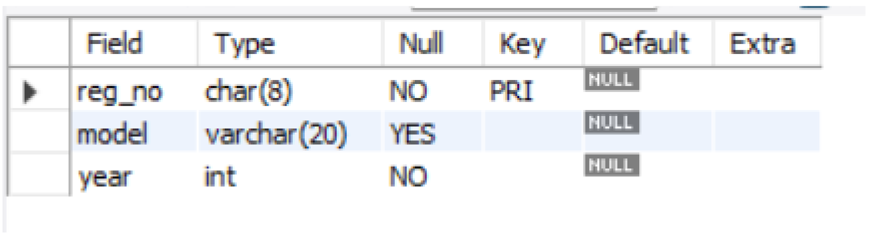
	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(3)	NO	PRI	NULL	
	name	varchar(20)	NO		NULL	
	address	varchar(100)	YES		NULL	

desc accident_094;



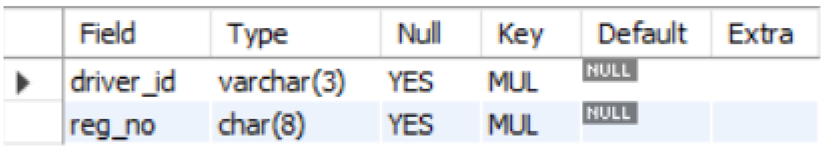
	Field	Type	Null	Key	Default	Extra
▶	report_no	int	NO	PRI	NULL	
	accident_date	date	YES		NULL	
	location	varchar(100)	YES		NULL	

desc car_094;



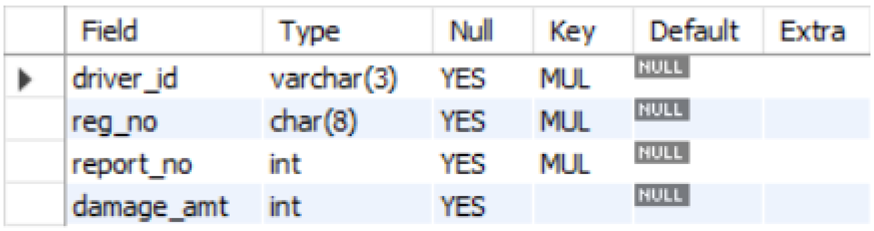
	Field	Type	Null	Key	Default	Extra
▶	reg_no	char(8)	NO	PRI	NULL	
	model	varchar(20)	YES		NULL	
	year	int	NO		NULL	

Desc owns_094;



	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(3)	YES	MUL	NULL	
	reg_no	char(8)	YES	MUL	NULL	

Desc participated_094;



	Field	Type	Null	Key	Default	Extra
▶	driver_id	varchar(3)	YES	MUL	NULL	
	reg_no	char(8)	YES	MUL	NULL	
	report_no	int	YES	MUL	NULL	
	damage_amt	int	YES		NULL	

Inserting Values into the table

insert into person_094 **values**

(**"A01"**, **"Richard"**, **"Sri Nagar"**),

(**"A02"**, **"Pradeep"**, **"Raj Nagar"**),

(**"A03"**, **"Smith"**, **"Ashok Nagar"**),

(**"A04"**, **"Venu"**, **"N R Colony"**),

(**"A05"**, **"John"**, **"Hanu Nagar"**);

insert into car_094 **values**

(**"KA052250"**, **"Indica"**, 1990),

(**"KA031181"**, **"Lancer"**, 1957),

(**"KA095477"**, **"Toyota"**, 1998),

(**"KA053408"**, **"Honda"**, 2008),

(**"KA041702"**, **"Audi"**, 2005);

insert into owns_094 **values**

(**"A01"**, **"KA052250"**);

insert into owns_094 **values**

(**"A02"**, **"KA031181"**);

insert into owns_094 **values**

(**"A03"**, **"KA095477"**);

insert into owns_094 **values**

(**"A04"**, **"KA053408"**);

insert into owns_094 **values**

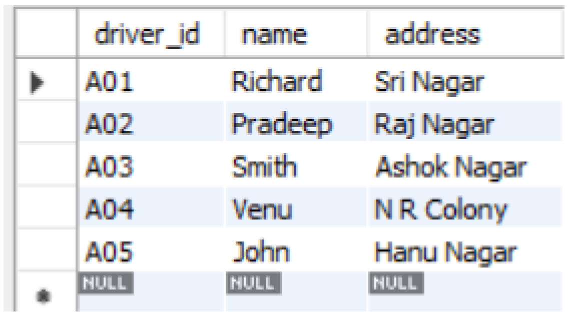
(**"A05"**, **"KA041702"**);

insert into accident_094 **values**

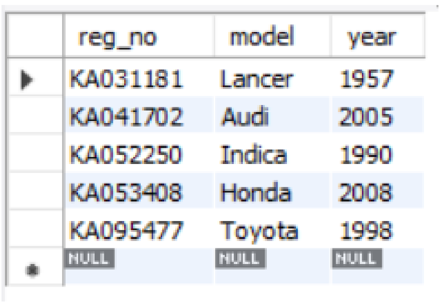
```
(11, "01-01-03", "Mysore Rd"),
(12, "02-02-04", "SE Circle"),
(13, "21-01-03", "Bull Temple Rd"),
(14, "17-02-08", "Mysore Rd"),
(15, "04-03-05", "KR Puram");
insert into participated_094 values
("A01", "KA052250", 11, 10000), ("A02", "KA031181", 12, 50000),
("A03", "KA053408", 13, 25000),
("A04", "KA095477", 14, 3000),
("A05", "KA041702", 15, 5000);
```

5

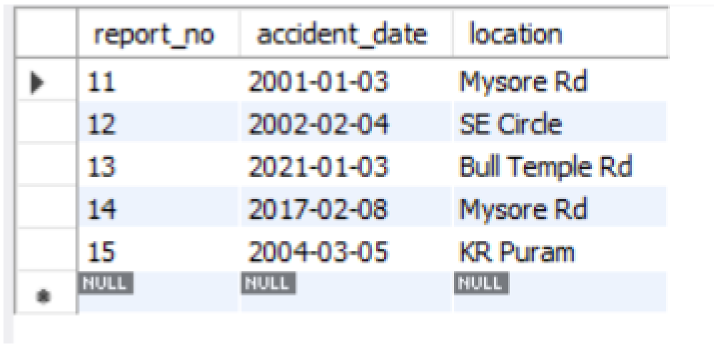
```
select * from person_094;
select * from car_094;
select * from accident_094;
select * from owns_094;
select * from participated_094;
```



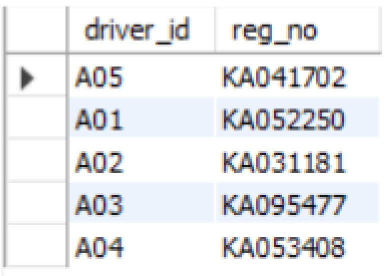
driver_id	name	address
A01	Richard	Sri Nagar
A02	Pradeep	Raj Nagar
A03	Smith	Ashok Nagar
A04	Venu	N R Colony
A05	John	Hanu Nagar
NULL	NULL	NULL



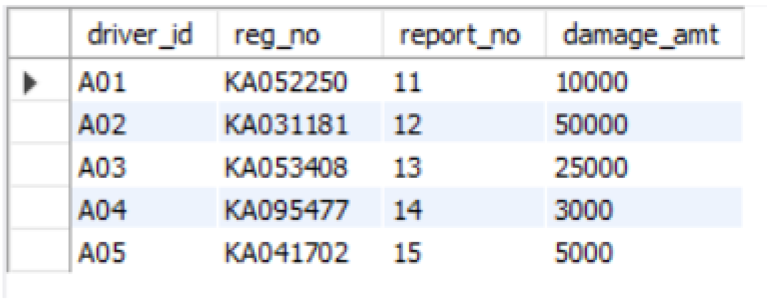
reg_no	model	year
KA031181	Lancer	1957
KA041702	Audi	2005
KA052250	Indica	1990
KA053408	Honda	2008
KA095477	Toyota	1998
NULL	NULL	NULL



report_no	accident_date	location
11	2001-01-03	Mysore Rd
12	2002-02-04	SE Circle
13	2021-01-03	Bull Temple Rd
14	2017-02-08	Mysore Rd
15	2004-03-05	KR Puram
NULL	NULL	NULL



driver_id	reg_no
A05	KA041702
A01	KA052250
A02	KA031181
A03	KA095477
A04	KA053408

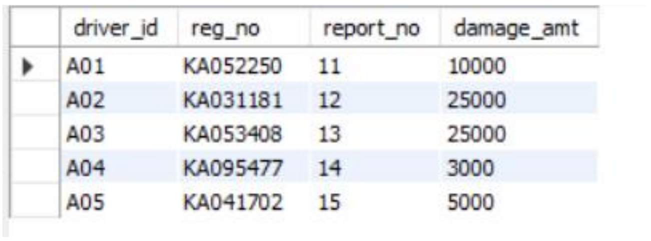


driver_id	reg_no	report_no	damage_amt
A01	KA052250	11	10000
A02	KA031181	12	50000
A03	KA053408	13	25000
A04	KA095477	14	3000
A05	KA041702	15	5000

Queries

Update the damage amount to 25000 for the car with a specific reg-num (example 'KA031181') for which the accident report number was 12.

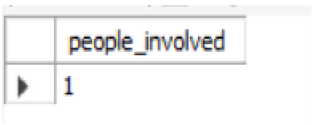
update participated_094 **set** damage_amt = 25000 **where** reg_no = **"KA031181" and** report_no = 12;



driver_id	reg_no	report_no	damage_amt
A01	KA052250	11	10000
A02	KA031181	12	25000
A03	KA053408	13	25000
A04	KA095477	14	3000
A05	KA041702	15	5000

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

select count(driver_id) people_involved **from** participated_094, accident_094 **where** participated_094.report_no = accident_094.report_no **and** accident_094.accident_date **like "%-08"**;

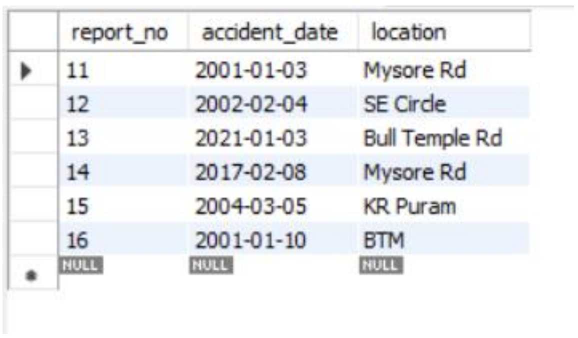


people_involved
1

Add a new accident to the database.

insert into accident_094 **values** (16, "01-01-10", "BTM");

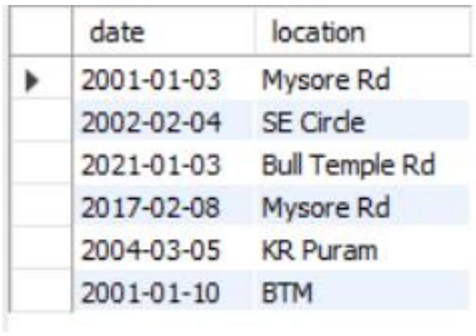
select * from accident_094;



report_no	accident_date	location
11	2001-01-03	Mysore Rd
12	2002-02-04	SE Circle
13	2021-01-03	Bull Temple Rd
14	2017-02-08	Mysore Rd
15	2004-03-05	KR Puram
16	2001-01-10	BTM
NULL	NULL	NULL

To Do:
Display accident date and location

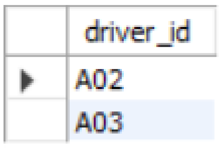
select accident_date as date, location from accident_094;



date	location
2001-01-03	Mysore Rd
2002-02-04	SE Circle
2021-01-03	Bull Temple Rd
2017-02-08	Mysore Rd
2004-03-05	KR Puram
2001-01-10	BTM

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

select participated_094.driver_id as driver_id from accident_094, participated_094 where accident_094.report_no = participated_094.report_no and participated_094.damage_amt >= 25000;



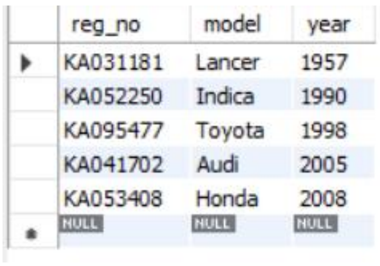
driver_id
A02
A03

2. More Queries on Insurance Database

Queries:

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

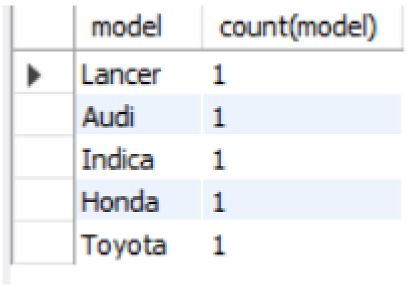
select * from car_094 order by year asc;



	reg_no	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 model, **count**(**model**) **from** participated_094, car_094 **where** participated_094.reg_no = car_094.reg_no **group by model**;



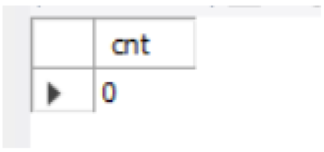
	model	count(model)
▶	Lancer	1
	Audi	1
	Indica	1
	Honda	1
	Toyota	1

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

Select count(distinct driver_id) cnt

From participated_094 a, accident-094 b

Where a.report_no=b.report_no and b.accident_date like ‘2008%’;

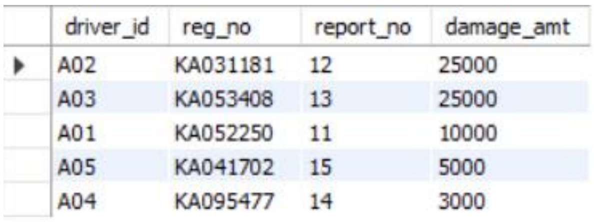


	cnt
▶	0

To Do:

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

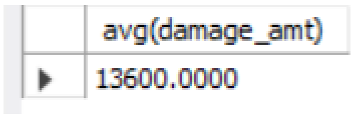
Select * from participated_094 order by damage_amt desc;



	driver_id	reg_no	report_no	damage_amt
▶	A02	KA031181	12	25000
	A03	KA053408	13	25000
	A01	KA052250	11	10000
	A05	KA041702	15	5000
	A04	KA095477	14	3000

Find the average damage amount

Select avg(damage_amt) from participated_094;

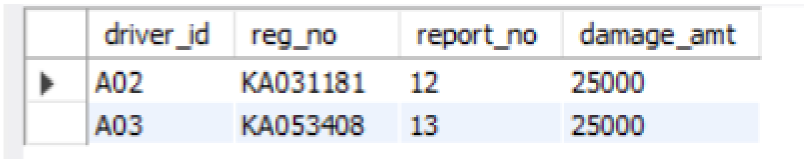


```
avg(damage_amt)
13600.0000
```

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

delete from participated_094 where damage_amt < (select avgd from(select avg(damage_amt) as avgd from participated_094) as subquery)

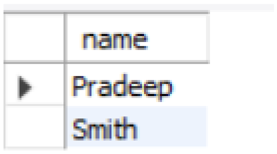
LIMIT 100;



driver_id	reg_no	report_no	damage_amt
A02	KA031181	12	25000
A03	KA053408	13	25000

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

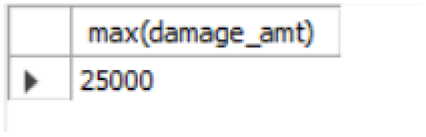
select name from person_094 a, participated_094 b where a.driver_id = b.driver_id and damage_amt > (select avg(damage_amt) from participated_094);



```
name
Pradeep
Smith
```

Find the maximum damage amount

select max(damage_amt) from participated_094;



```
max(damage_amt)
25000
```

3. Bank Database

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

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

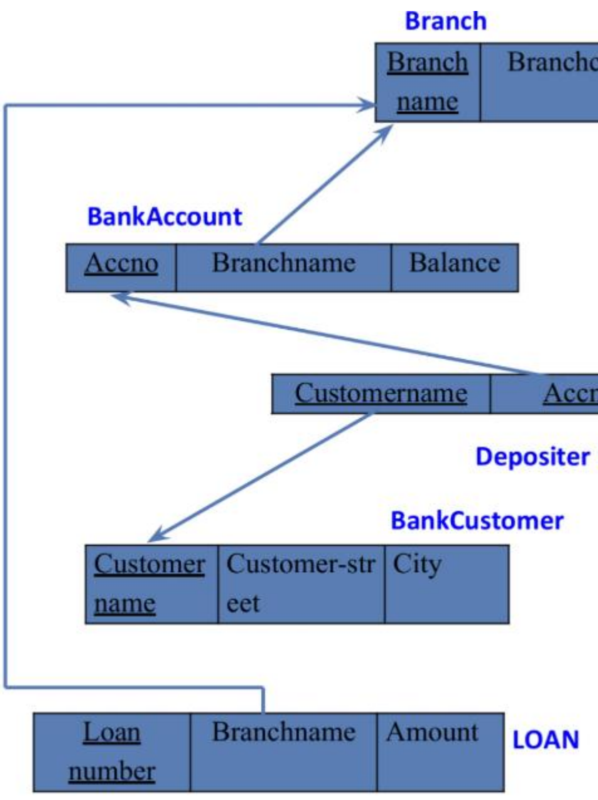
- BankCustomer (customer-name: String, customer-street: String, customer-city: String) - Depositer(customer-name: String, accno: int)
- LOAN (loan-number: int, branch-name: String, amount: real)

- 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

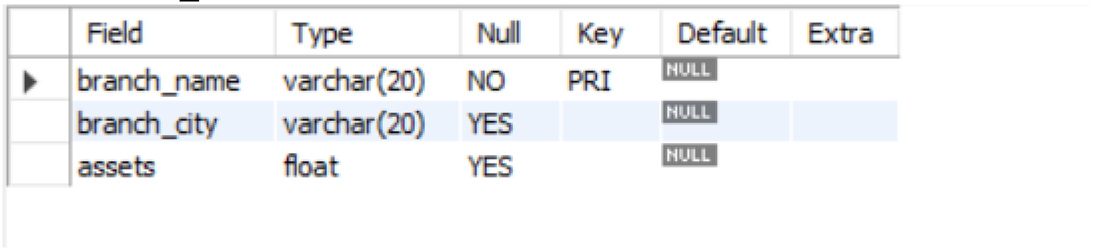
```
create database bank_094;
use bank_094;
```

Create table

```
create table branch_094(
branch_name varchar(20) primary key,
branch_city varchar(20),
assets float
);
create table bank_account_094(
acc_no int primary key,
branch_name varchar(20),
balance float,
foreign key(branch_name) references branch_094(branch_name)
);
create table deposits_094(
customer_name varchar(20),
acc_no int,
foreign key(acc_no) references bank_account_094(acc_no),
foreign key(customer_name) references bank_customer_094(customer_name)
);
create table bank_customer_094(
customer_name varchar(20) primary key,
customer_street varchar(50),
city varchar(15)
);
create table loans_094(
loan_no int primary key,
branch_name varchar(20),
amt float,
foreign key(branch_name) references branch_094(branch_name)
);
```

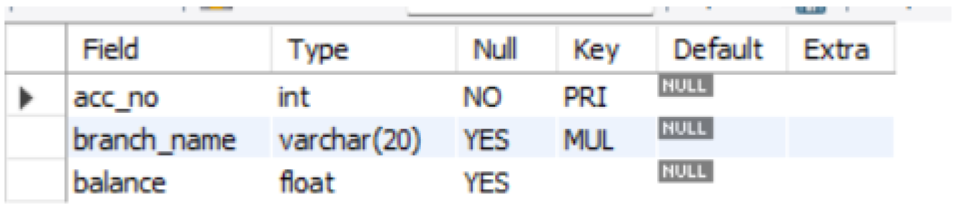
Structure of table

desc branch_094;



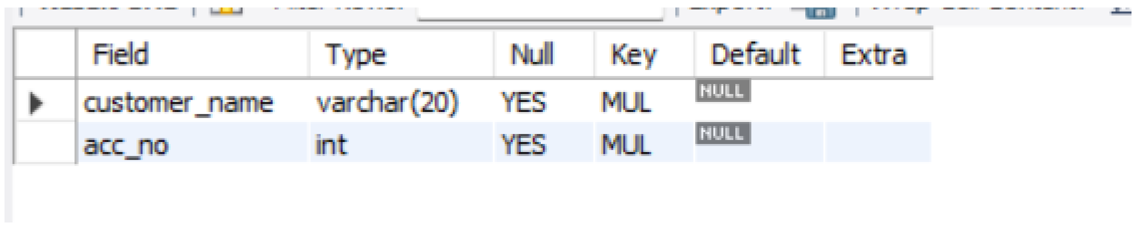
	Field	Type	Null	Key	Default	Extra
▶	branch_name	varchar(20)	NO	PRI	NULL	
	branch_city	varchar(20)	YES		NULL	
	assets	float	YES		NULL	

desc bank_account_094;



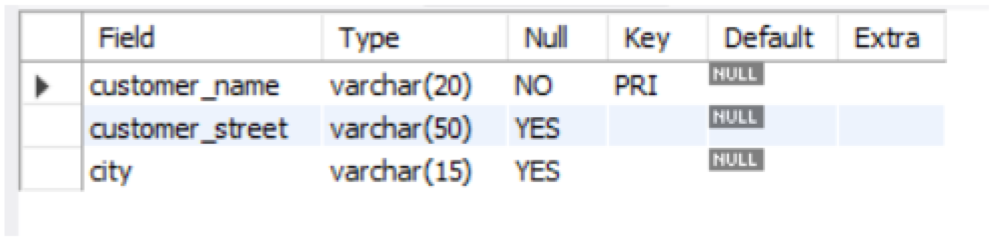
	Field	Type	Null	Key	Default	Extra
▶	acc_no	int	NO	PRI	NULL	
	branch_name	varchar(20)	YES	MUL	NULL	
	balance	float	YES		NULL	

Desc deposits_094;



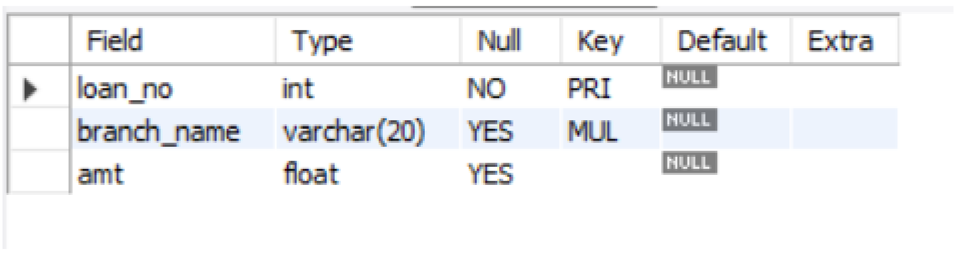
	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(20)	YES	MUL	NULL	
	acc_no	int	YES	MUL	NULL	

Desc bank_customer_094;



	Field	Type	Null	Key	Default	Extra
▶	customer_name	varchar(20)	NO	PRI	NULL	
	customer_street	varchar(50)	YES		NULL	
	city	varchar(15)	YES		NULL	

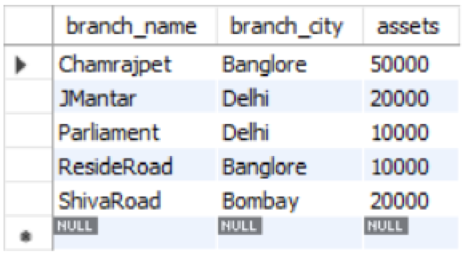
Desc loans_094;



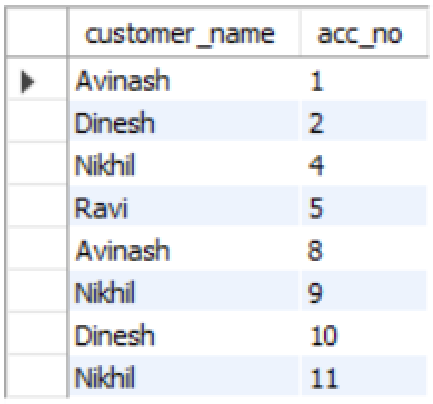
Field	Type	Null	Key	Default	Extra
loan_no	int	NO	PRI	NULL	
branch_name	varchar(20)	YES	MUL	NULL	
amt	float	YES		NULL	

Inserting values to the table:

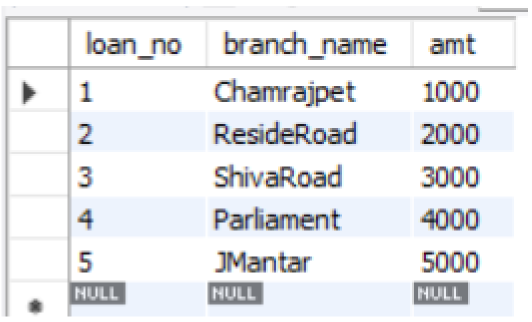
```
insert into branch_094 values
("Chamrajpet","Banglore",50000),
("ResideRoad","Banglore",10000),
("ShivaRoad","Bombay",20000),
("Parliament","Delhi",10000),
("JMantar","Delhi",20000);
insert into bank_account_094 values
(1,"Chamrajpet",2000),
(2,"ResideRoad",5000),
(3,"ShivaRoad",6000),
(4,"Parliament",9000),
(5,"JMantar",8000),
(6,"ShivaRoad",4000),
(8,"ResideRoad",4000),
(9,"Parliament",3000),
(10,"ResideRoad",5000),
(11,"JMantar",2000);
insert into bank_customer_094 values
("Avinash","BulTemple","Banglore"),
("Dinesh","Banrgutta","Banglore"),
("Mohan","Nationalcollege","Banglore"),
("Nikhil","Akbarroad","Delhi"),
("Ravi","Prithvirajroad","Delhi");
insert into deposits_094 values
("Avinash",1),
("Dinesh",2),
("Nikhil",4),
("Ravi",5),
("Avinash",8),
("Nikhil",9),
("Dinesh",10),
("Nikhil",11);
insert into loans_094 values
(1,"Chamrajpet",1000),
(2,"ResideRoad",2000),
(3,"ShivaRoad",3000),
(4,"Parliament",4000),
(5,"JMantar",5000);

select * from branch_094;
select * from deposits_094;
select * from loans_094;
select * from bank_customer_094;
select * from bank_account_094;
```



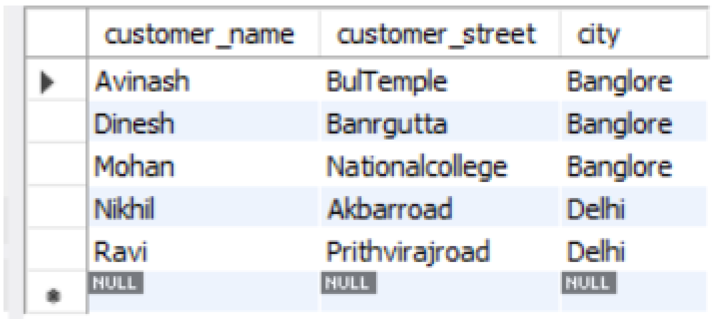
branch_name	branch_city	assets
Chamrajpet	Banglore	50000
JMantar	Delhi	20000
Parliament	Delhi	10000
ResideRoad	Banglore	10000
ShivaRoad	Bombay	20000
NULL	NULL	NULL



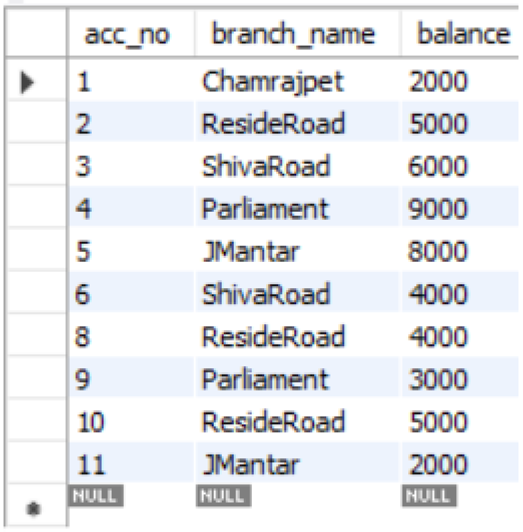
customer_name	acc_no
Avinash	1
Dinesh	2
Nikhil	4
Ravi	5
Avinash	8
Nikhil	9
Dinesh	10
Nikhil	11



loan_no	branch_name	amt
1	Chamrajpet	1000
2	ResideRoad	2000
3	ShivaRoad	3000
4	Parliament	4000
5	JMantar	5000
NULL	NULL	NULL



customer_name	customer_street	city
Avinash	BulTemple	Banglore
Dinesh	Banrgutta	Banglore
Mohan	Nationalcollege	Banglore
Nikhil	Akbarroad	Delhi
Ravi	Prithvirajroad	Delhi
NULL	NULL	NULL



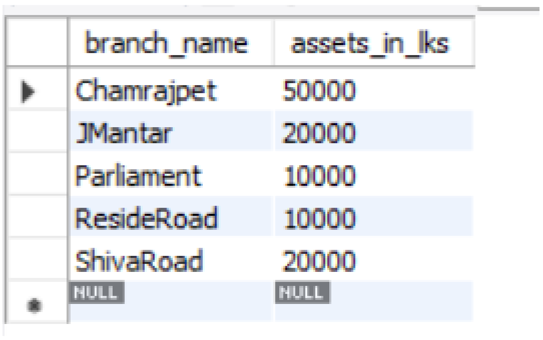
acc_no	branch_name	balance
1	Chamrajpet	2000
2	ResideRoad	5000
3	ShivaRoad	6000
4	Parliament	9000
5	JMantar	8000
6	ShivaRoad	4000
8	ResideRoad	4000
9	Parliament	3000
10	ResideRoad	5000
11	JMantar	2000
NULL	NULL	NULL

Queries

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

alter table branch_094 **rename column** assets **to** assets_in_lks;

select branch_name, assets_in_lks **from** branch_094;

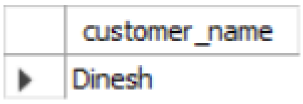


branch_name	assets_in_lks
Chamrajpet	50000
JMantar	20000
Parliament	10000
ResideRoad	10000
ShivaRoad	20000
NULL	NULL

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

select d.customer_name **from** deposits_094 d, bank_account_094 b **where**

b.branch_name=**'ResideRoad' and** d.acc_no=b.acc_no **group by** d.customer_name **having count**(d.acc_no)>=2;



customer_name
Dinesh

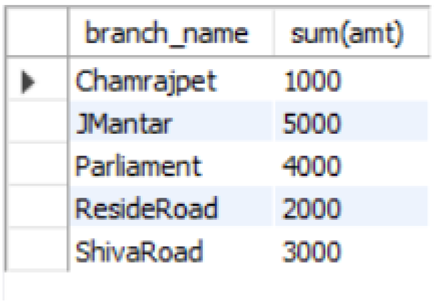
● Create a view which gives each branch the sum of the amount of all the loans at the branch.

create view loansum **as** (

select branch_name, **sum**(amt) **from** loans_094 **group by** branch_name

);

select * from loansum;



branch_name	sum(amt)
Chamrajpet	1000
JMantar	5000
Parliament	4000
ResideRoad	2000
ShivaRoad	3000

4. More Queries on Bank Database

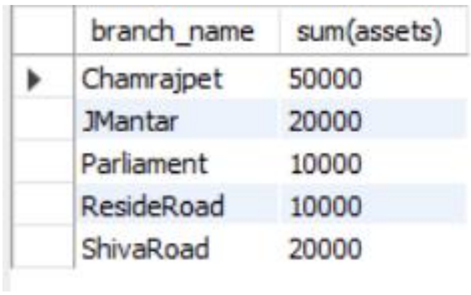
Queries:

Retrieve all branches and their respective total assets

select branch_name, sum(assets)

from branch_094

group by branch_name;



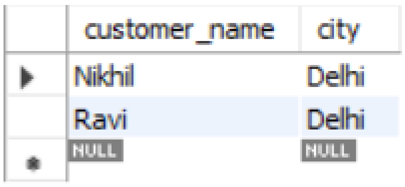
branch_name	sum(assets)
Chamrajpet	50000
JMantar	20000
Parliament	10000
ResideRoad	10000
ShivaRoad	20000

List all customers who live in a particular city(Delhi)

select customer_name , city

from bank_customer_094

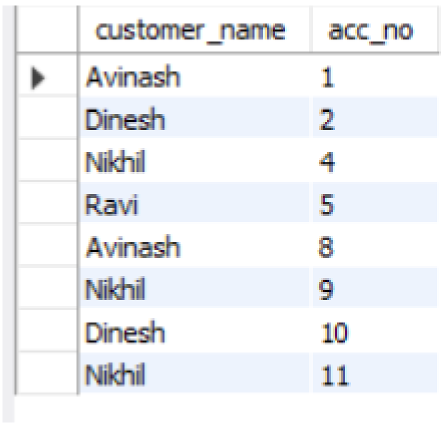
where city = 'Delhi';



customer_name	city
Nikhil	Delhi
Ravi	Delhi
NULL	NULL

List all customers with their account numbers

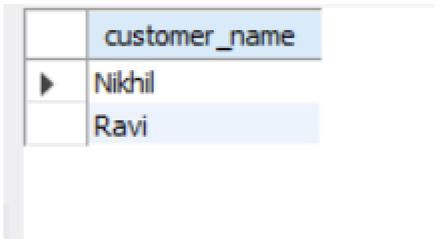
Select * from deposits_094;



customer_name	acc_no
Avinash	1
Dinesh	2
Nikhil	4
Ravi	5
Avinash	8
Nikhil	9
Dinesh	10
Nikhil	11

Find all the customers who have accounts with a balance greater than a specified amount (6000)

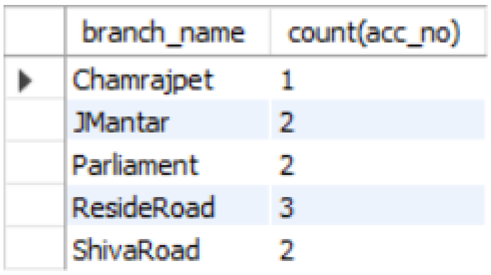
```
SELECT distinct bc.customer_name
FROM bank_customer_094 bc
JOIN deposits_094 ba
    ON bc.customer_name = ba.customer_name
WHERE bc.customer_name IN (
    SELECT d.customer_name
    FROM deposits_094 d
    JOIN bank_account_094 b ON d.acc_no = b.acc_no
    WHERE b.balance > 6000
);
```



customer_name
Nikhil
Ravi

Get the number of accounts held at each branch

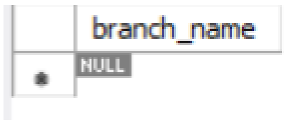
```
select branch_name , count(acc_no) from bank_account_094 group by
branch_name;
```



branch_name	count(acc_no)
Chamrajpet	1
JMantar	2
Parliament	2
ResideRoad	3
ShivaRoad	2

Find all branches that have no loans issued

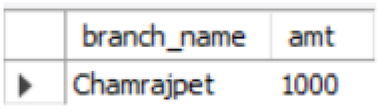
```
select b.branch_name from branch_094 b where b.branch_name not in (select
l.branch_name from loans_094 l where l.branch_name= b.branch_name);
```



	branch_name
*	NULL

Retrive the branch with the smallest total loan amount

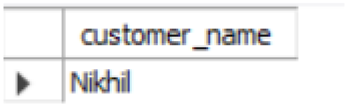
select l.branch_name, l.amt from loans_094 l where l.amt = (select min(b.amt) from loans_094 b); employee



	branch_name	amt
▶	Chamrajpet	1000

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

```
SELECT d.customer_name
FROM deposits_094 d
JOIN bank_account_094 ba ON d.acc_no = ba.acc_no
JOIN branch_094 b ON ba.branch_name = b.branch_name
WHERE b.branch_city = 'Delhi'
GROUP BY d.customer_name
HAVING COUNT(DISTINCT b.branch_name) = (
   SELECT COUNT(*)
   FROM branch_094
   WHERE branch_city = 'Delhi'
);
```



	customer_name
▶	Nikhil

5. Employee Database

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,

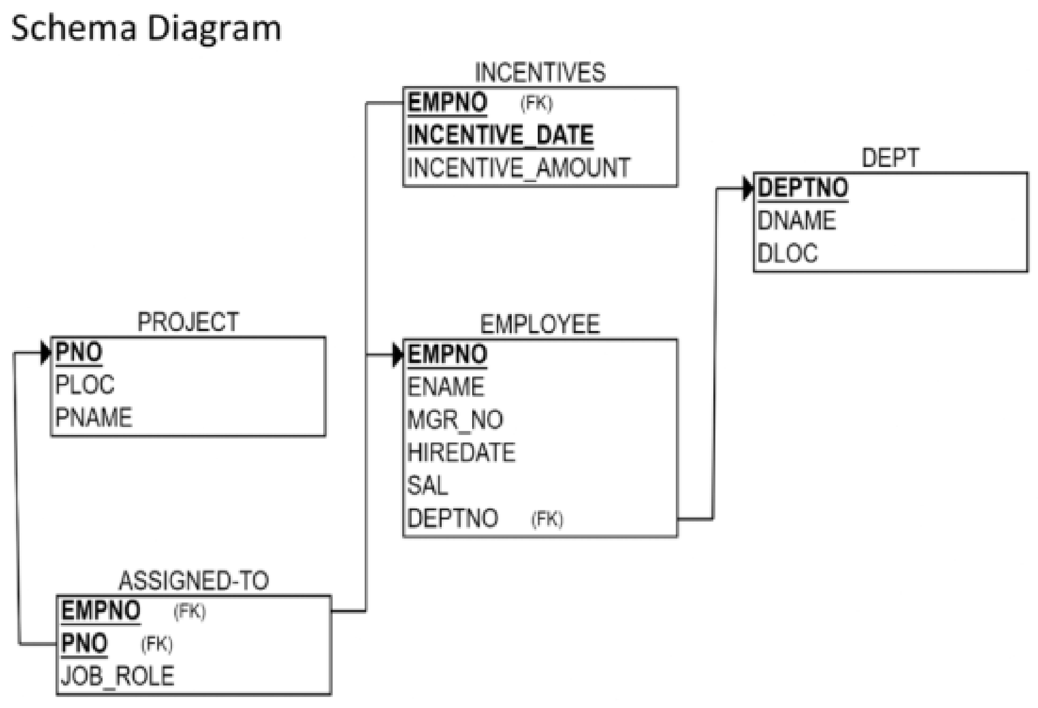
Hyderabad, or Mysuru

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

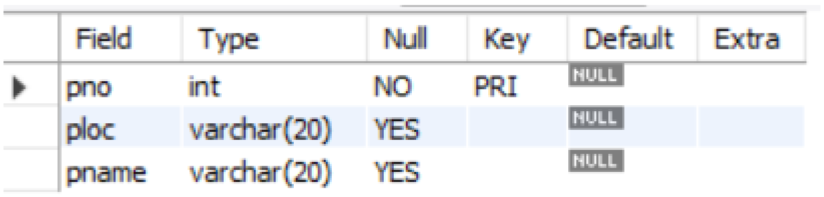
```
create database employee_database_094;
use employee_database_094;
```

Create table

```
create table project_094(
pno int primary key,
ploc varchar(20),
pname varchar(20)
);
create table dept_094(
deptno int primary key,
dname varchar(30),
dloc varchar(30)
);
create table employee_094(
empno int primary key,
ename varchar(20),
mgr_no int,
hiredate date,
sal double,
deptno int,
foreign key(deptno) references dept_094(deptno)
);
create table assigned_to_094(
empno int primary key,
pno int,
job_role varchar(20),
foreign key(empno) references employee_094(empno),
foreign key(pno) references project_094(pno)
);
create table incentives_094(
empno int,
incentive_date date primary key,
incentive_amount double,
foreign key(empno) references employee_094(empno)
);
```

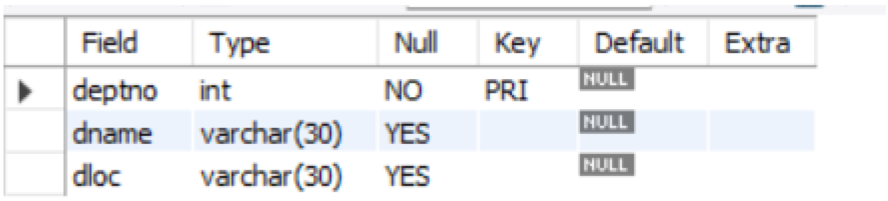
Structure of tables:

desc project_094;



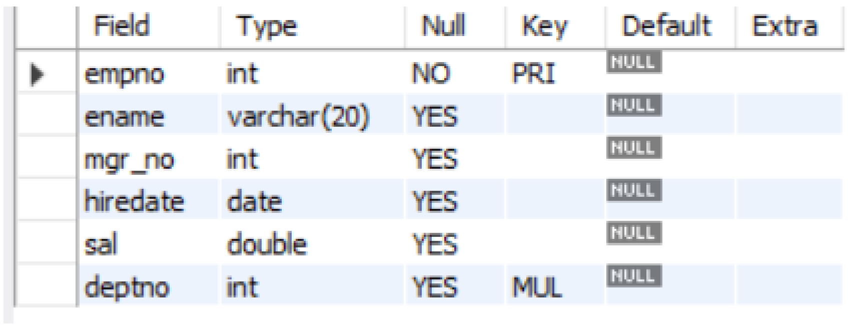
Field	Type	Null	Key	Default	Extra
pno	int	NO	PRI	NULL	
ploc	varchar(20)	YES		NULL	
pname	varchar(20)	YES		NULL	

desc dept_094;



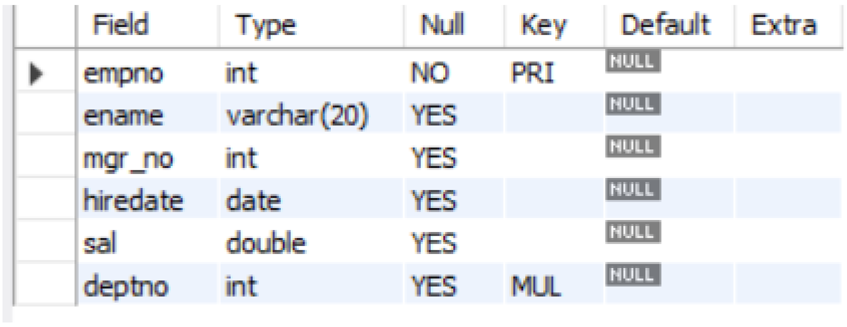
Field	Type	Null	Key	Default	Extra
deptno	int	NO	PRI	NULL	
dname	varchar(30)	YES		NULL	
dloc	varchar(30)	YES		NULL	

desc employee_094;



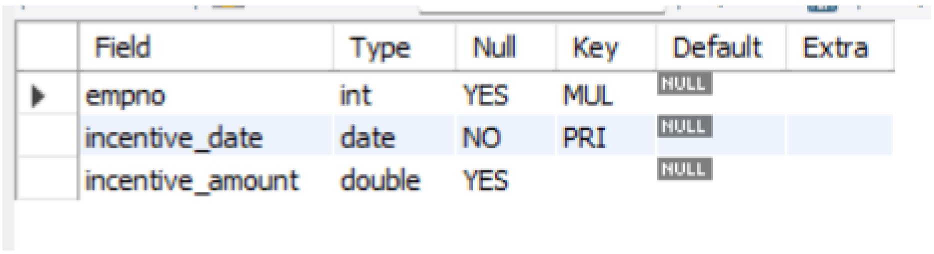
Field	Type	Null	Key	Default	Extra
empno	int	NO	PRI	NULL	
ename	varchar(20)	YES		NULL	
mgr_no	int	YES		NULL	
hiredate	date	YES		NULL	
sal	double	YES		NULL	
deptno	int	YES	MUL	NULL	

desc assigned_to_094;



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 incentives_094;



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_094 **values**

(1,**"bengaluru"**,**"abcd"**),

(2,**"hyderabad"**,**"bcda"**),

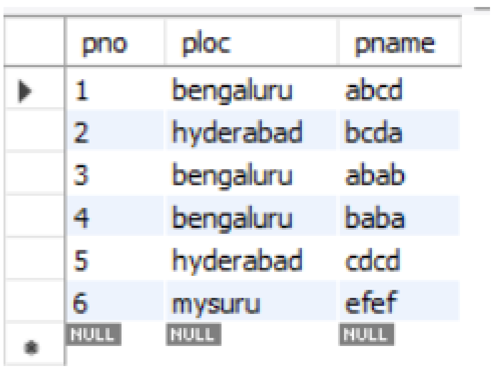
(3,**"bengaluru"**,**"abab"**),

(4,**"bengaluru"**,**"baba"**),

(5,**"hyderabad"**,**"cdcd"**),

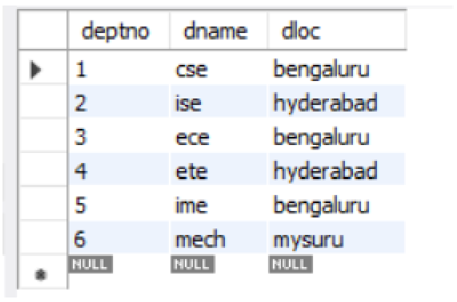
(6, **"mysuru"**, **"efef"**);

select * from project_094;



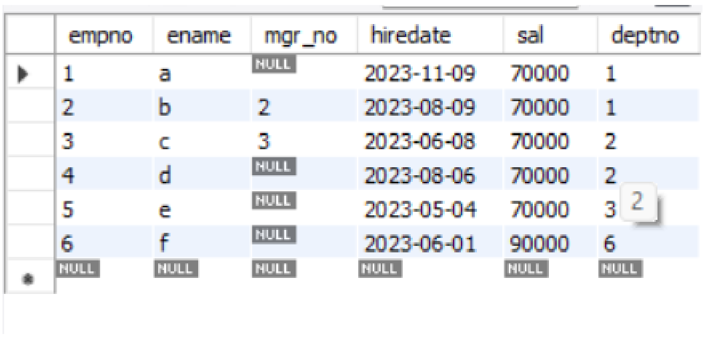
pno	ploc	pname
1	bengaluru	abcd
2	hyderabad	bcda
3	bengaluru	abab
4	bengaluru	baba
5	hyderabad	cdcd
6	mysuru	efef
NULL	NULL	NULL

```
insert into dept_094 values
(1,"cse","bengaluru"),
(2,"ise","hyderabad"),
(3,"ece","bengaluru"),
(4,"ete","hyderabad"),
(5,"ime","bengaluru"),
(6, "mech", "mysuru");
select * from dept_094;
```



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

```
insert into employee_094 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_094;
```



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 assigned_to_094 **values**

(1,1, **"employee"**),

(2,1, **"manager"**),

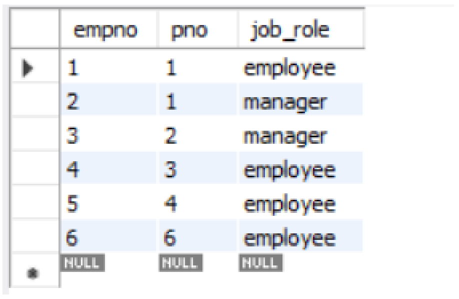
(3,2, **"manager"**),

(4,3, **"employee"**),

(5,4, **"employee"**),

(6, 6, **"employee"**);

select * **from** assigned_to_094;



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

insert into incentives_094 **values**

(1,**"2023-12-9"**,10000),

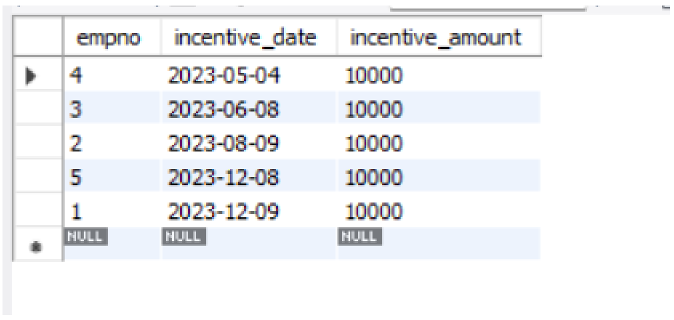
(2,**"2023-8-9"**,10000),

(3,**"2023-6-8"**,10000),

(4,**"2023-5-4"**,10000),

(5,**"2023-12-8"**,10000);

select * **from** incentives_094;



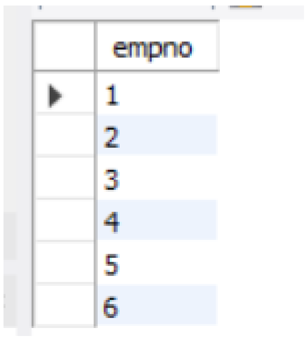
empno	incentive_date	incentive_amount
4	2023-05-04	10000
3	2023-06-08	10000
2	2023-08-09	10000
5	2023-12-08	10000
1	2023-12-09	10000
NULL	NULL	NULL

QUERIES:

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

select assigned_to_094.empno **from** assigned_to_094, project_094

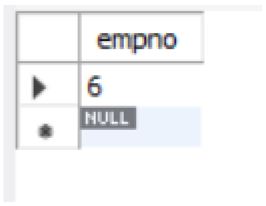
where assigned_to_094.pno = project_094.pno **and** project_094.ploc **in** (**"bengaluru"**, **"mysuru"**, **"hyderabad"**);



```
empno
1
2
3
4
5
6
```

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

select empno from employee_094 where empno not in (select empno from incentives_094);



```
empno
6
NULL
```

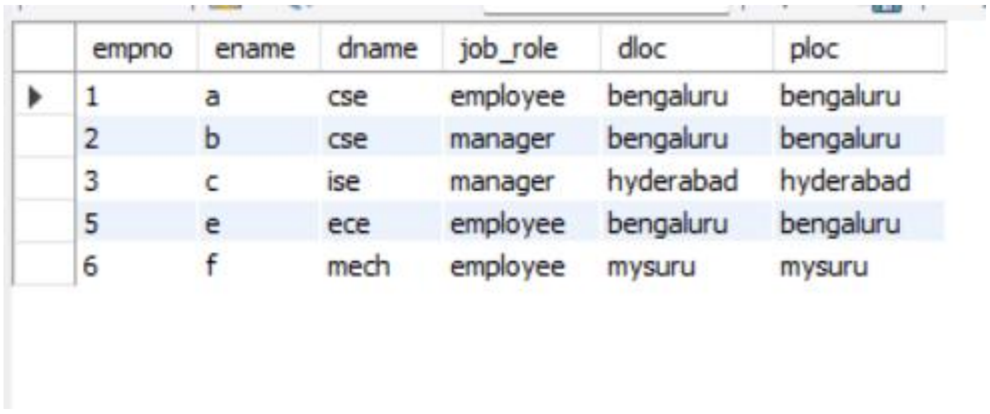
Write a SQL query to find the employees name, number, dept, job_role, department location and project location who are working for a project location same as his/her department location.

select employee_094.empno, ename, dname, job_role, dloc, ploc

from employee_094, assigned_to_094, project_094, dept_094

where ploc = dloc **and** assigned_to_094.empno = employee_094.empno

and employee_094.deptno = dept_094.deptno **and** project_094.pno = assigned_to_094.pno;

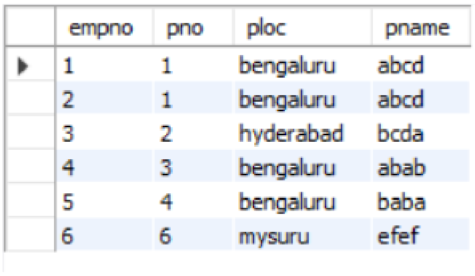


empno	ename	dname	job_role	dloc	ploc
1	a	cse	employee	bengaluru	bengaluru
2	b	cse	manager	bengaluru	bengaluru
3	c	ise	manager	hyderabad	hyderabad
5	e	ece	employee	bengaluru	bengaluru
6	f	mech	employee	mysuru	mysuru

6. More Queries on Employee Database

1.List all the employees with their project details

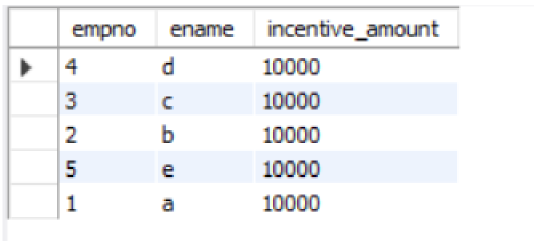
Select a.empno, p.pno, p.ploc, pname from assigned_to_094 a join project_094 p on p.pno = a.pno;



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

2. Find all employees who received incentives along with the total incentives amount

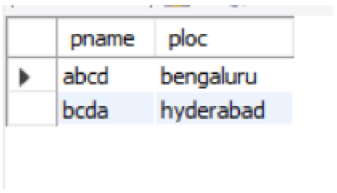
Select i.empno, e.ename, i.incentive_amount from employee_094 e join incentives_094 i on i.empno = e.empno;



empno	ename	incentive_amount
4	d	10000
3	c	10000
2	b	10000
5	e	10000
1	a	10000

3. Retrieve the project names and locations with employees assigned as managers.

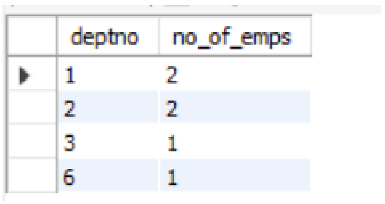
Select p.pname, p.ploc from project_094 p join assigned_to_094 a on a.pno = p.pno where a.job_role = 'manager';



pname	ploc
abcd	bengaluru
bcda	hyderabad

4. List departments along with the number of employees in each department

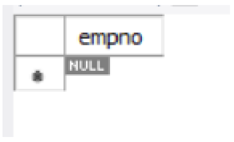
Select deptno, count(deptno) as no_of_emps from employee_094 group by deptno;



deptno	no_of_emps
1	2
2	2
3	1
6	1

5. Find employees who have not been assigned to any project

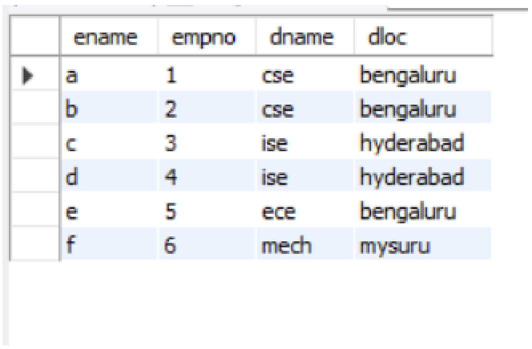
Select e.empno from employee_094 e where e.empno not in (select a.empno from assigned_to_094 a);



empno
NULL

6.List all employees along with their department name & location

Select e.ename, e.empno, d.dname, d.dloc from employee_094 e, dept_094 d where e.deptno = d.deptno;



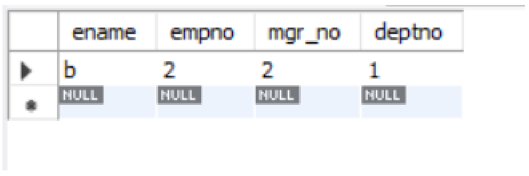
ename	empno	dname	dloc
a	1	cse	bengaluru
b	2	cse	bengaluru
c	3	ise	hyderabad
d	4	ise	hyderabad
e	5	ece	bengaluru
f	6	mech	mysuru

7.Retrive the details of employees who work under a specific manager(eg: manager with empno= 1)

Select ename, empno, mgr_no, deptno

From employee_094 e

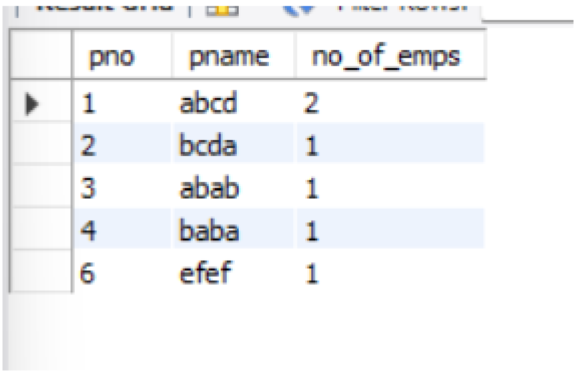
Where mgr_no = 2;



	ename	empno	mgr_no	deptno
▶	b	2	2	1
*	NULL	NULL	NULL	NULL

8.List all projects that have employees assigned and the number of employees on each project

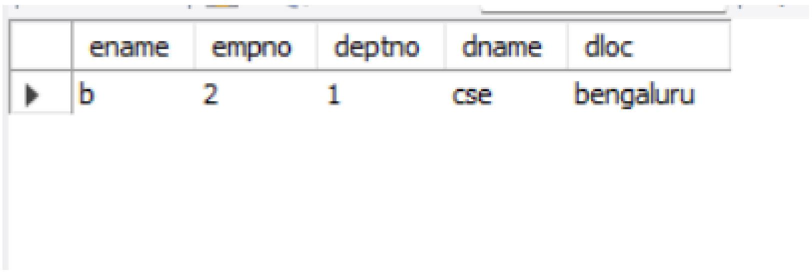
Select a.pno, p.pname, count(a.empno) as no_of_emps from assigned_to_094 a , project_094 p where p.pno = a.pno group by pno;



	pno	pname	no_of_emps
▶	1	abcd	2
	2	bcda	1
	3	abab	1
	4	baba	1
	6	efef	1

9. Find employees with the same manager and list their dept. details

Select e.ename, e.empno, d.deptno, d.dname, d.dloc from employee_094 e , dept_094 d where e.deptno= d.deptno and mgr_no= 2;

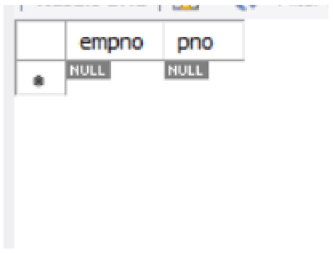


	ename	empno	deptno	dname	dloc
▶	b	2	1	cse	bengaluru

10. Retrieve all employees who have the role of 'developer' on any project

Select a.empno, a.pno

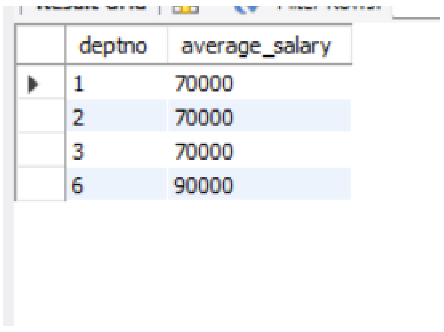
From assigned_to_094 a where job_role = "developer";



	empno	pno
*	NULL	NULL

11. Display the department wise average salary of employees

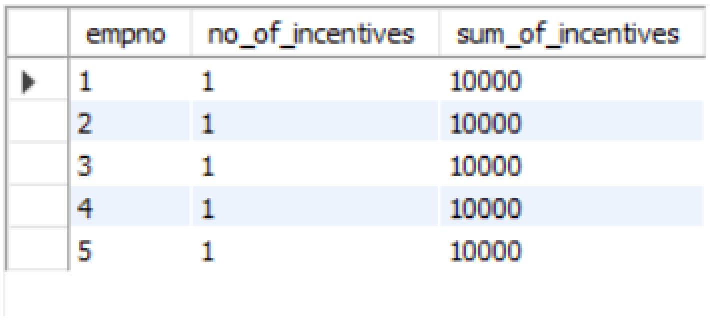
Select deptno, avg(sal) as average_salary from employee_094 group by deptno;



	deptno	average_salary
▶	1	70000
	2	70000
	3	70000
	6	90000

12. List the total number of incentives given to each employee and the sum of incentives for each

Select empno, count(incentive_amount) as no_of_incentives, sum(incentive_amount) as sum_of_incentives from incentives_094 group by empno;

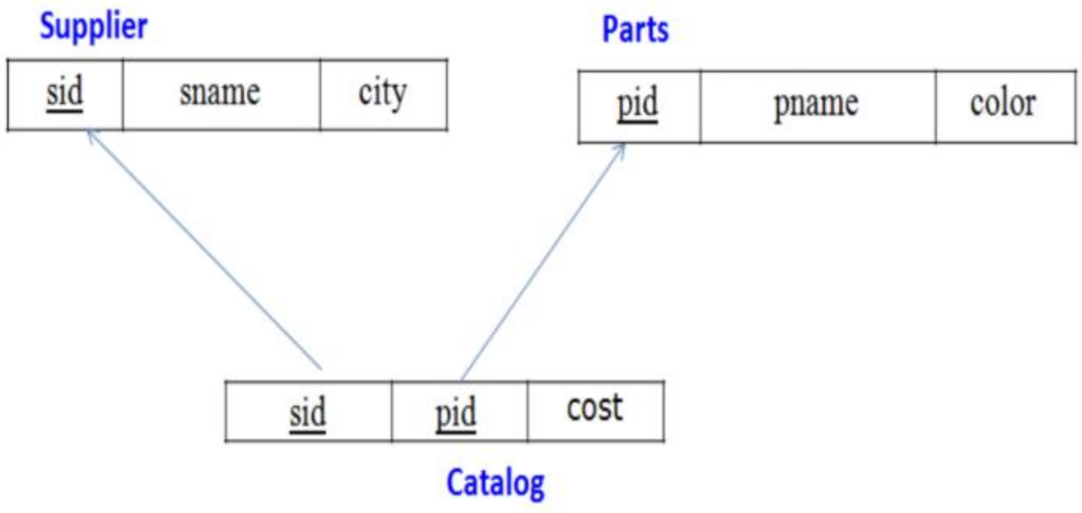


	empno	no_of_incentives	sum_of_incentives
▶	1	1	10000
	2	1	10000
	3	1	10000
	4	1	10000
	5	1	10000

7. Supplier Database

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_204;

use supply_204;

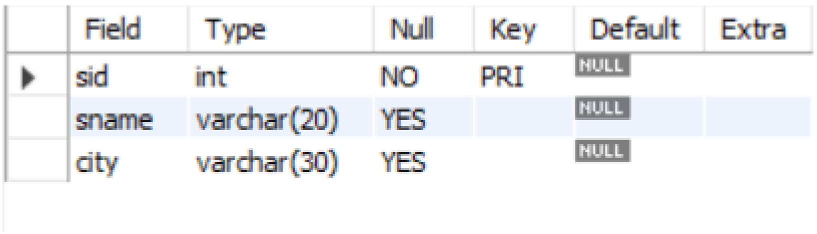
Create table

```
create table supplier_204(
sid int primary key,
sname varchar(20),
city varchar(30)
);
```

```
create table parts_204(
pid int primary key,
pname varchar(20),
color varchar(20)
);
create table catalog_204(
sid int, pid int,
cost int,
foreign key(sid) references supplier_204(sid),
foreign key(pid) references parts_204(pid)
);
```

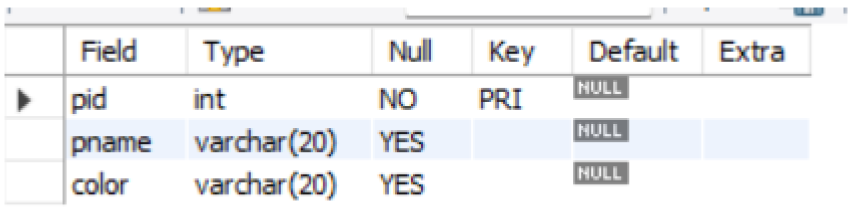
Structure of tables:

desc supplier_204;



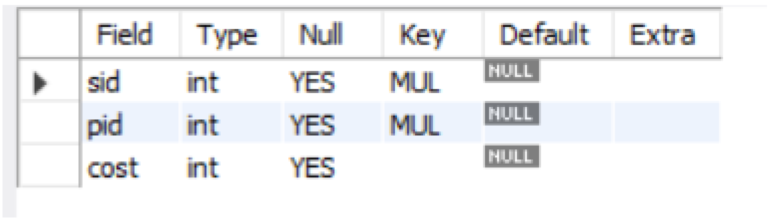
	Field	Type	Null	Key	Default	Extra
▶	sid	int	NO	PRI	NULL	
	sname	varchar(20)	YES		NULL	
	city	varchar(30)	YES		NULL	

desc parts_204;



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

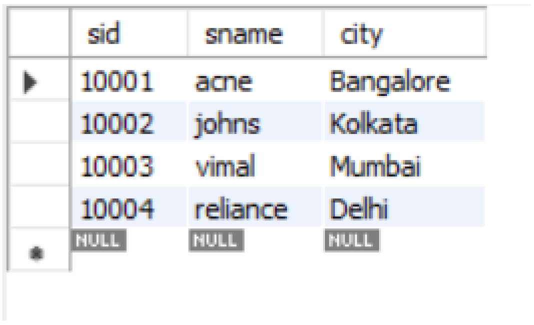
desc catalog_204;



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

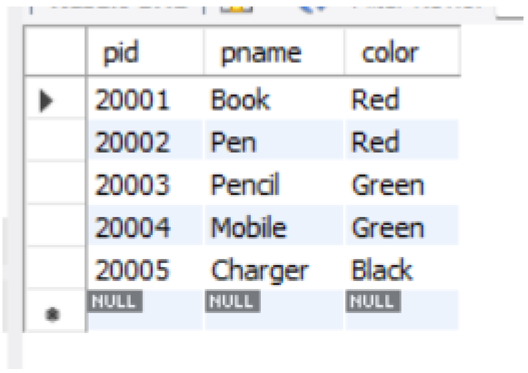
Inserting Values to the table

```
insert into supplier_204 values
(10001, "acne", "Bangalore"),
(10002, "johns", "Kolkata"),
(10003, "vimal", "Mumbai"),
(10004, "reliance", "Delhi");
select * from supplier_204;
```



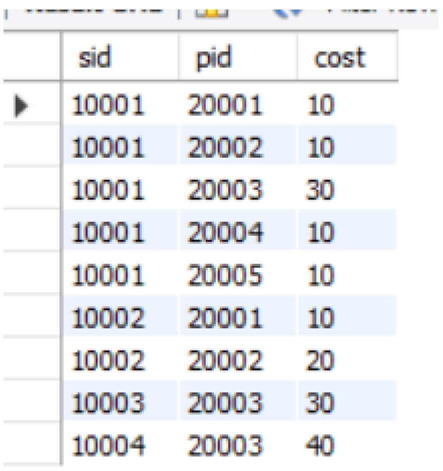
sid	sname	city
10001	acne	Bangalore
10002	johns	Kolkata
10003	vimal	Mumbai
10004	reliance	Delhi
NULL	NULL	NULL

```
insert into parts_204 values
(20001,"Book","Red"),
(20002,"Pen","Red"),
(20003,"Pencil","Green"),
(20004,"Mobile","Green"),
(20005,"Charger","Black");
Select * from parts_204;
```



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

```
Insert into catalog_204 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_204;
```

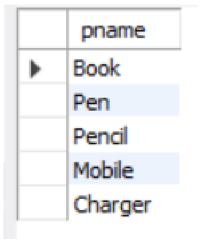


sid	pid	cost
10001	20001	10
10001	20002	10
10001	20003	30
10001	20004	10
10001	20005	10
10002	20001	10
10002	20002	20
10003	20003	30
10004	20003	40

Queries

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

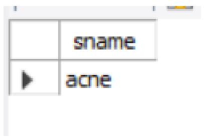
select pname **from** parts_204 **where** pid **in** (**select** pid **from** catalog_204);



pname
Book
Pen
Pencil
Mobile
Charger

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

select sname **from** supplier_204 **where sid in** (**select sid from** catalog_204 **group by sid having count**(**distinct** pid) = (**select count**(**distinct** pid) **from** parts_204));

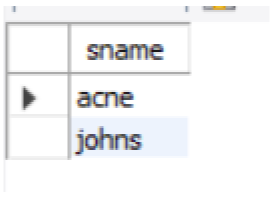


	sname
▶	acne

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

select distinct sname **from** supplier_204, parts_204, catalog_204

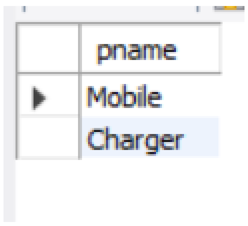
where supplier_204.sid = catalog_204.sid **and** parts_204.pid = catalog_204.pid **and** parts_204.color=**"Red"**;



	sname
▶	acne
	johns

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

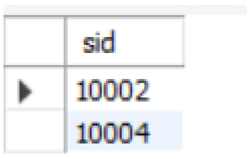
select pname **from** parts_204 **where** pid **not in** (**select** pid **from** catalog_204 **where sid in** (**select sid from** supplier_204 **where** sname != **"acne"**));



	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).**

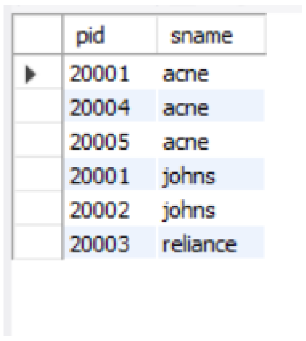
select sid from catalog_204 a **where** a.cost > (**select avg**(b.cost) **from** catalog_204 b **where** a.pid = b.pid **group by** b.pid);



	sid
▶	10002
	10004

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

select pid, sname **from** catalog_204 a, supplier_204 **where** a.cost = (**select max**(b.cost) **from** catalog_204 b **where** a.pid = b.pid **group by** b.pid) **and** supplier_204.sid = a.sid;



pid	sname
20001	acne
20004	acne
20005	acne
20001	johns
20002	johns
20003	reliance

8. NoSQL Lab 1

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.

Create database

db.createCollection("Student");

Create table & Inserting Values to the table

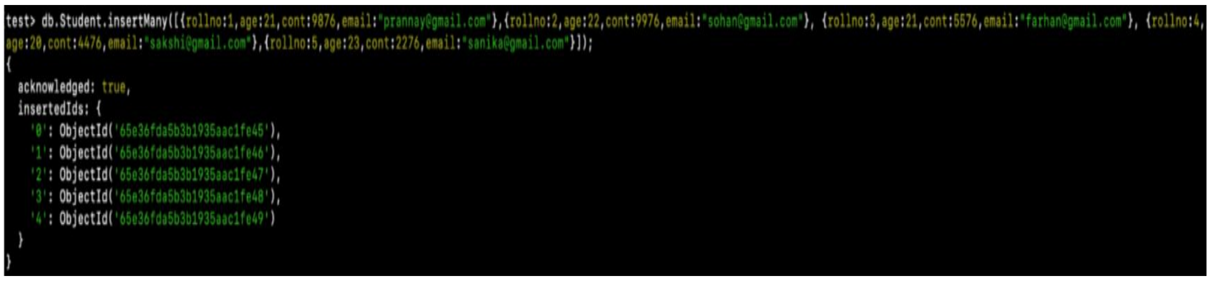
db.Student.insertMany([{rollno:1,age:21,cont:9876,email:"prannay@gmail.com"},{rollno:2,a

ge:22,cont:9976,email:"sohan@gmail.com"},

{rollno:3,age:21,cont:5576,email:"farhan@gmail.com"},

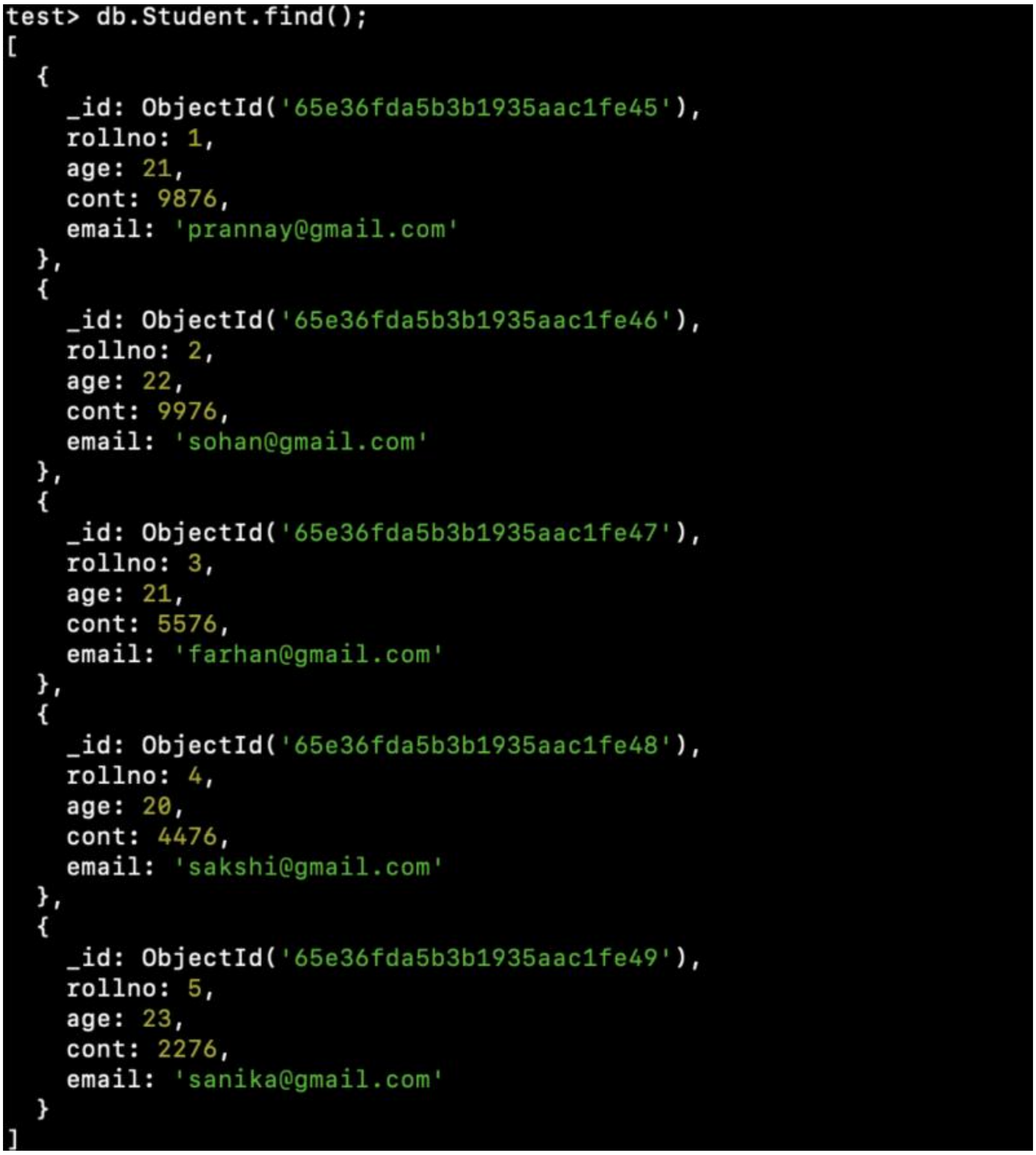
{rollno:4,age:20,cont:4476,email:"sakshi@gmail.com"},{rollno:5,age:23,cont:2276,email:"sa

nika@gmail.com"}]);



```
test> db.Student.insertMany([{rollno:1,age:21,cont:9876,email:"prannay@gmail.com"},{rollno:2,age:22,cont:9976,email:"sohan@gmail.com"}, {rollno:3,age:21,cont:5576,email:"farhan@gmail.com"}, {rollno:4,
age:20,cont:4476,email:"sakshi@gmail.com"},{rollno:5,age:23,cont:2276,email:"sanika@gmail.com"}]);
{
  acknowledged: true,
  insertedIds: {
    '0': ObjectId('65e36fda5b3b1935aac1fe45'),
    '1': ObjectId('65e36fda5b3b1935aac1fe46'),
    '2': ObjectId('65e36fda5b3b1935aac1fe47'),
    '3': ObjectId('65e36fda5b3b1935aac1fe48'),
    '4': ObjectId('65e36fda5b3b1935aac1fe49')
  }
}
```

db.Student.find();

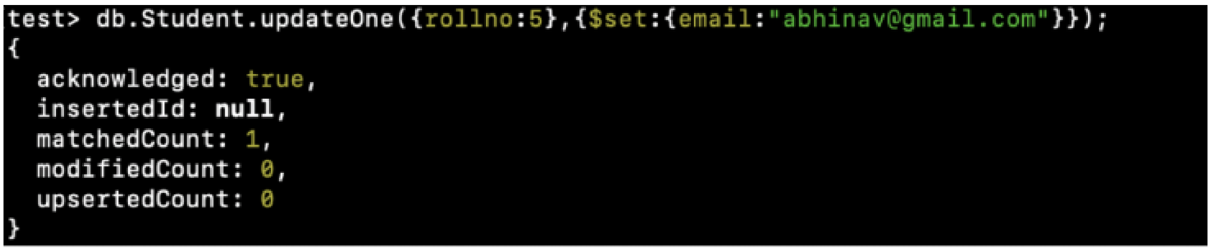


```
test> db.Student.find();
[
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe45'),
    rollno: 1,
    age: 21,
    cont: 9876,
    email: 'prannay@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe46'),
    rollno: 2,
    age: 22,
    cont: 9976,
    email: 'sohan@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe47'),
    rollno: 3,
    age: 21,
    cont: 5576,
    email: 'farhan@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe48'),
    rollno: 4,
    age: 20,
    cont: 4476,
    email: 'sakshi@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe49'),
    rollno: 5,
    age: 23,
    cont: 2276,
    email: 'sanika@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"}});

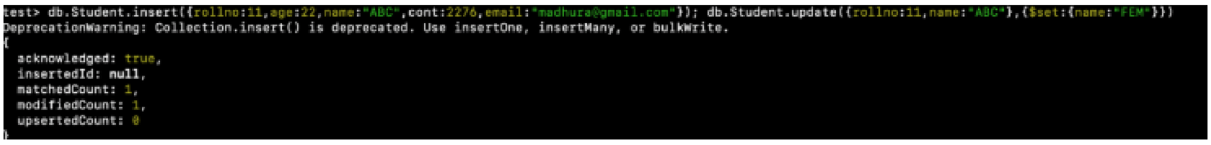


```
test> db.Student.updateOne({rollno:5},{$set:{email:"abhinav@gmail.com"}});
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 0,
  upsertedCount: 0
}
```

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

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

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



```
test> 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"}})
DeprecationWarning: Collection.insert() is deprecated. Use insertOne, insertMany, or bulkWrite.
{
  acknowledged: true,
  insertedId: null,
  matchedCount: 1,
  modifiedCount: 1,
  upsertedCount: 0
}
```

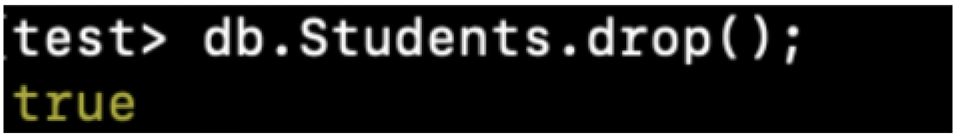
● Export the created table into local file system

mongoexport mongodb+srv://204:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Student -- out C:\Users\DISHADS\Documents\test.Students.json

● Drop the table

db.Student.drop();



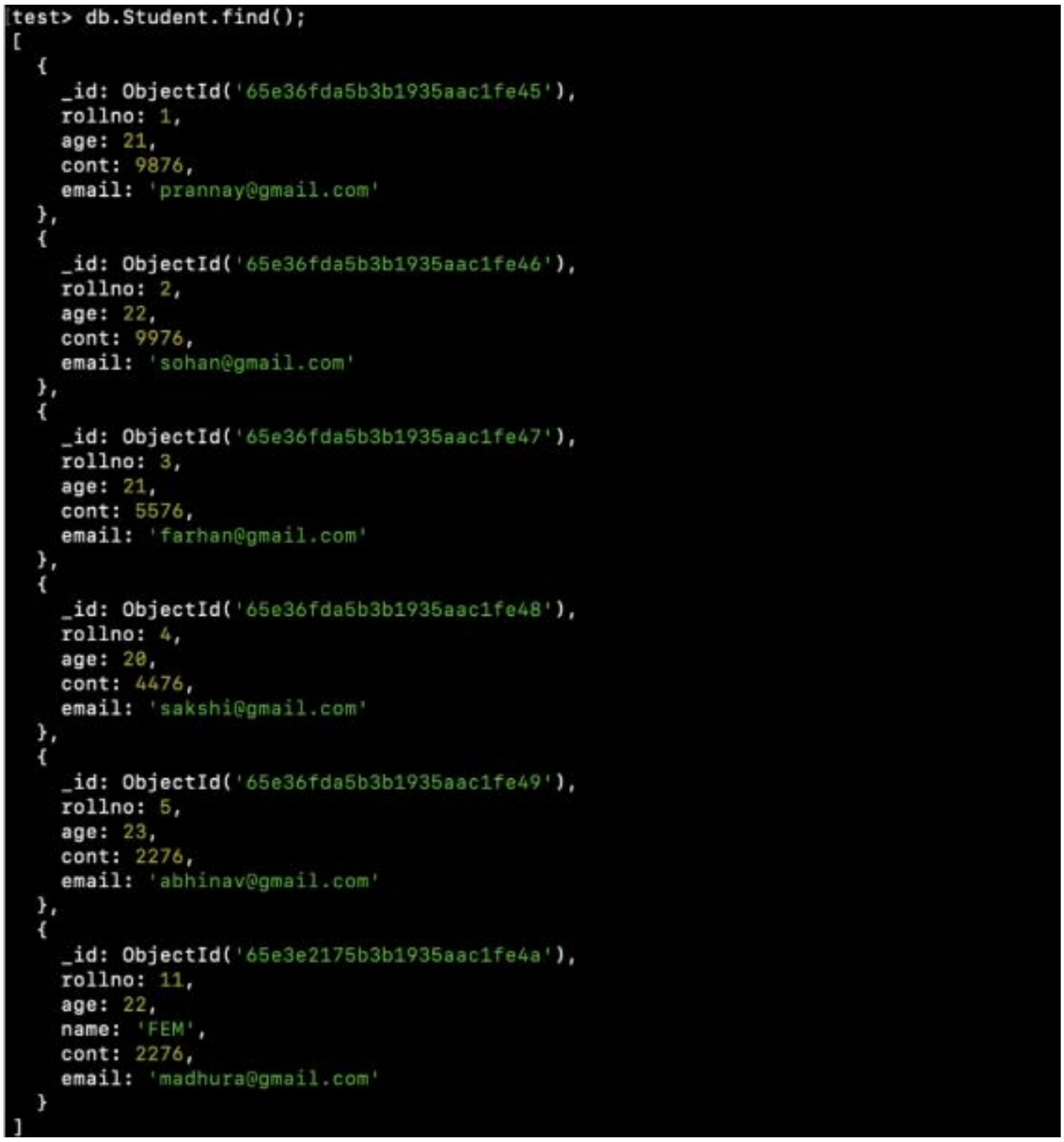
```
test> db.Students.drop();
true
```

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

mongoimport mongodb+srv://204:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Student -- type json -file C:\Users\DISHADS\Documents\test.Students.json

db.Student.find();



```
test> db.Student.find();
[
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe45'),
    rollno: 1,
    age: 21,
    cont: 9876,
    email: 'prannay@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe46'),
    rollno: 2,
    age: 22,
    cont: 9976,
    email: 'sohan@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe47'),
    rollno: 3,
    age: 21,
    cont: 5576,
    email: 'farhan@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe48'),
    rollno: 4,
    age: 20,
    cont: 4476,
    email: 'sakshi@gmail.com'
  },
  {
    _id: ObjectId('65e36fda5b3b1935aac1fe49'),
    rollno: 5,
    age: 23,
    cont: 2276,
    email: 'abhinav@gmail.com'
  },
  {
    _id: ObjectId('65e3e2175b3b1935aac1fe4a'),
    rollno: 11,
    age: 22,
    name: 'FEM',
    cont: 2276,
    email: 'madhura@gmail.com'
  }
]
```

9. NoSQL LAB 2

Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes.

Cust_id, Acc_Bal, Acc_Type

2. Insert at least 5 values into the table

3. Write a query to display those records whose total account balance

is greater than 1200 of account type ‘Checking’ for each customer_id.

4. Determine Minimum and Maximum account balance for each

customer_id.

5. Export the created collection into local file system

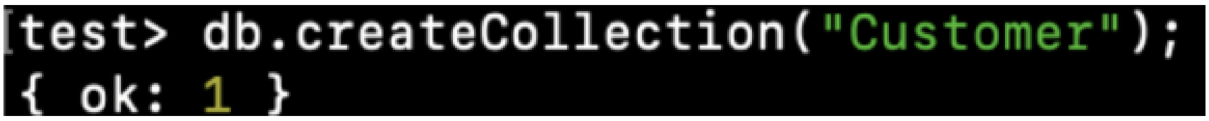
6. Drop the table

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

collection.

Create Table:

db.createCollection("Customer");



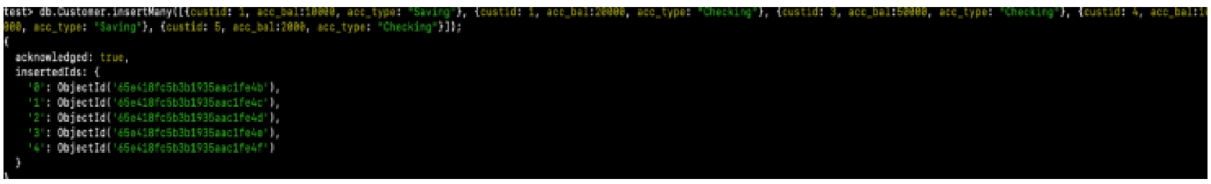
```
test> db.createCollection("Customer");
{ ok: 1 }
```

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"}]);

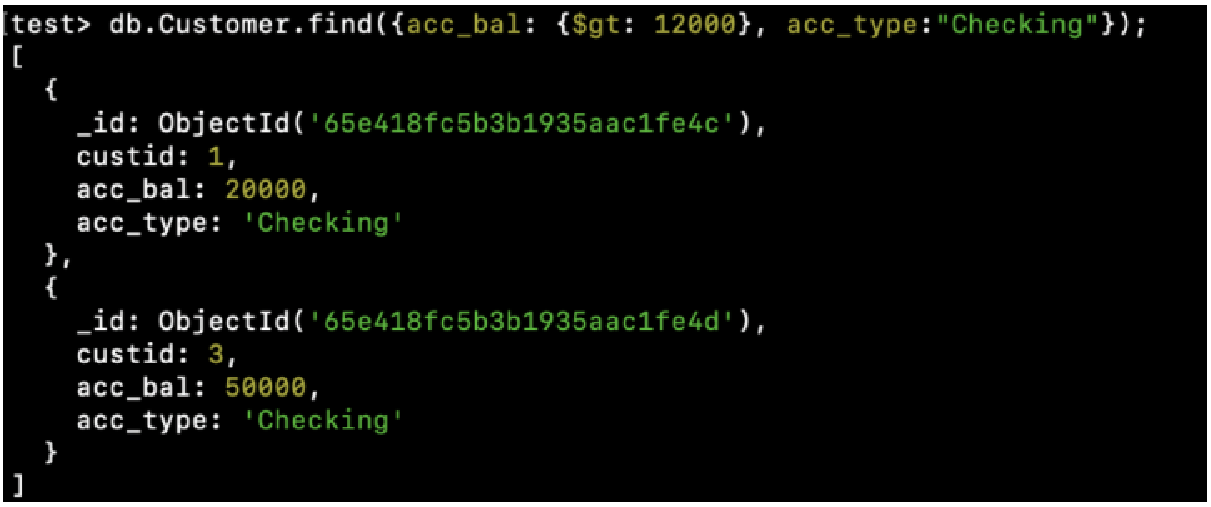


```
test> db.Customer.insertMany([{custid: 1, acc_bal:10000, acc_type: "Saving"}, {custid: 1, acc_bal:20000, acc_type: "Checking"}, {custid: 3, acc_bal:50000, acc_type: "Checking"}, {custid: 4, acc_bal:10
000, acc_type: "Saving"}, {custid: 5, acc_bal:2000, acc_type: "Checking"}]);
{
  acknowledged: true,
  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"});

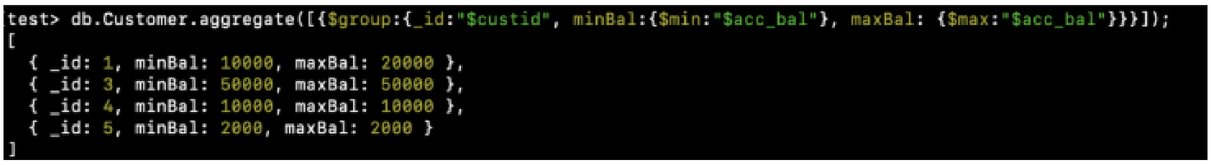


```
test> db.Customer.find({acc_bal: {$gt: 12000}, acc_type:"Checking"});
[
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4c'),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4d'),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'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

● 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 }
]
```

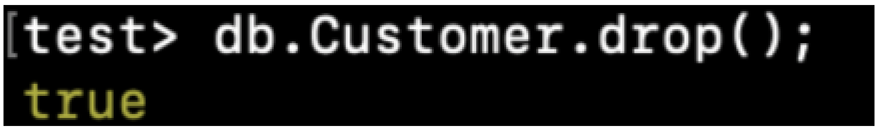
● Exporting the collection to a json file

mongoexport
mongodb+srv://204:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Customer -- out
C:\Users\DISHADS\Documents\test.Customer.json

● Dropping collection "Customer"

db.Customer.drop();



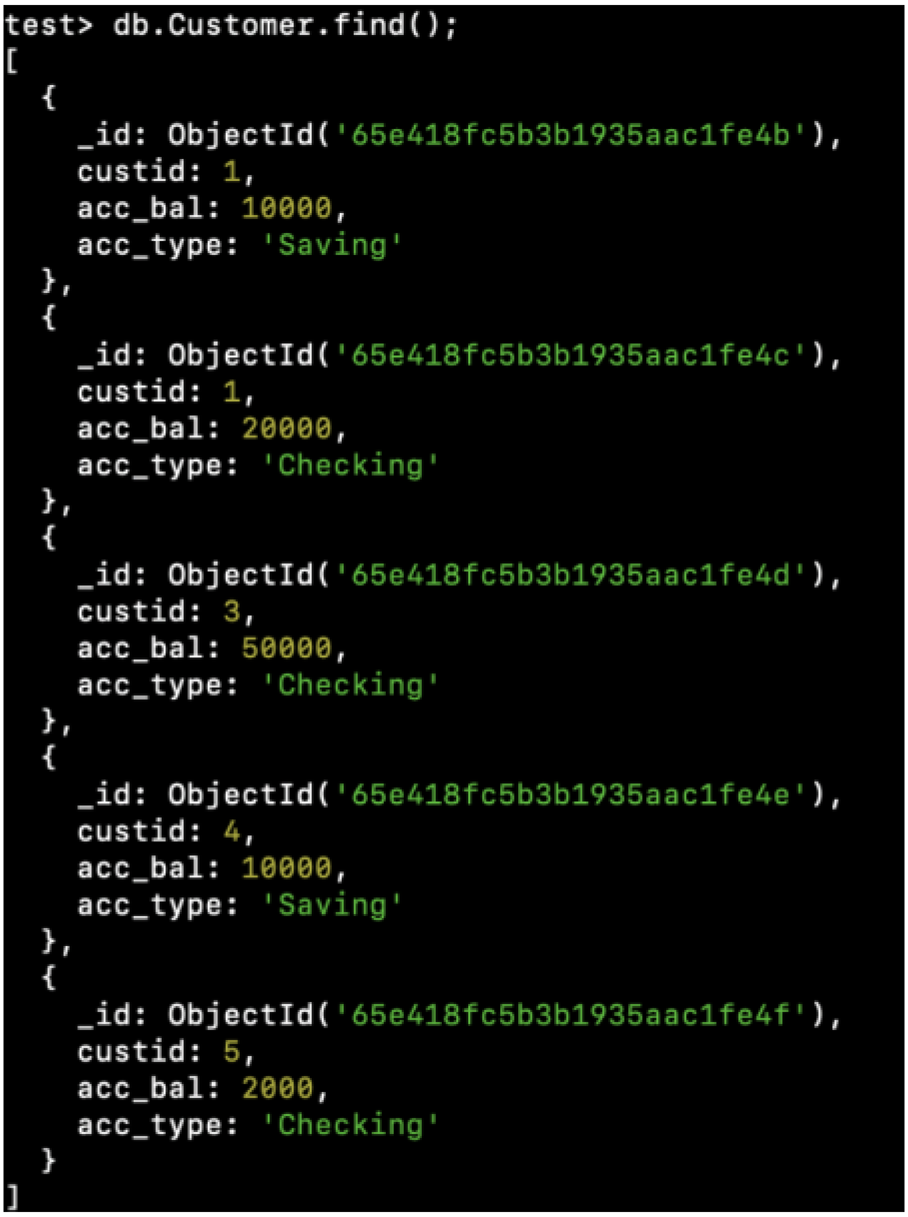
```
[test> db.Customer.drop();
 true
```

● Exporting from a json file to the collection

mongoimport
mongodb+srv://204:<password>@cluster0.xbmgopf.mongodb.net/test

--collection=Customer -- type json -file
C:\Users\DISHADS\Documents\test.Customer.json

db.Customer.find();



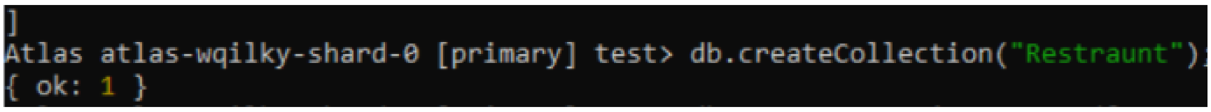
```
test> db.Customer.find();
[
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4b'),
    custid: 1,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4c'),
    custid: 1,
    acc_bal: 20000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4d'),
    custid: 3,
    acc_bal: 50000,
    acc_type: 'Checking'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4e'),
    custid: 4,
    acc_bal: 10000,
    acc_type: 'Saving'
  },
  {
    _id: ObjectId('65e418fc5b3b1935aac1fe4f'),
    custid: 5,
    acc_bal: 2000,
    acc_type: 'Checking'
  }
]
```

10.NoSql LAB 3

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");



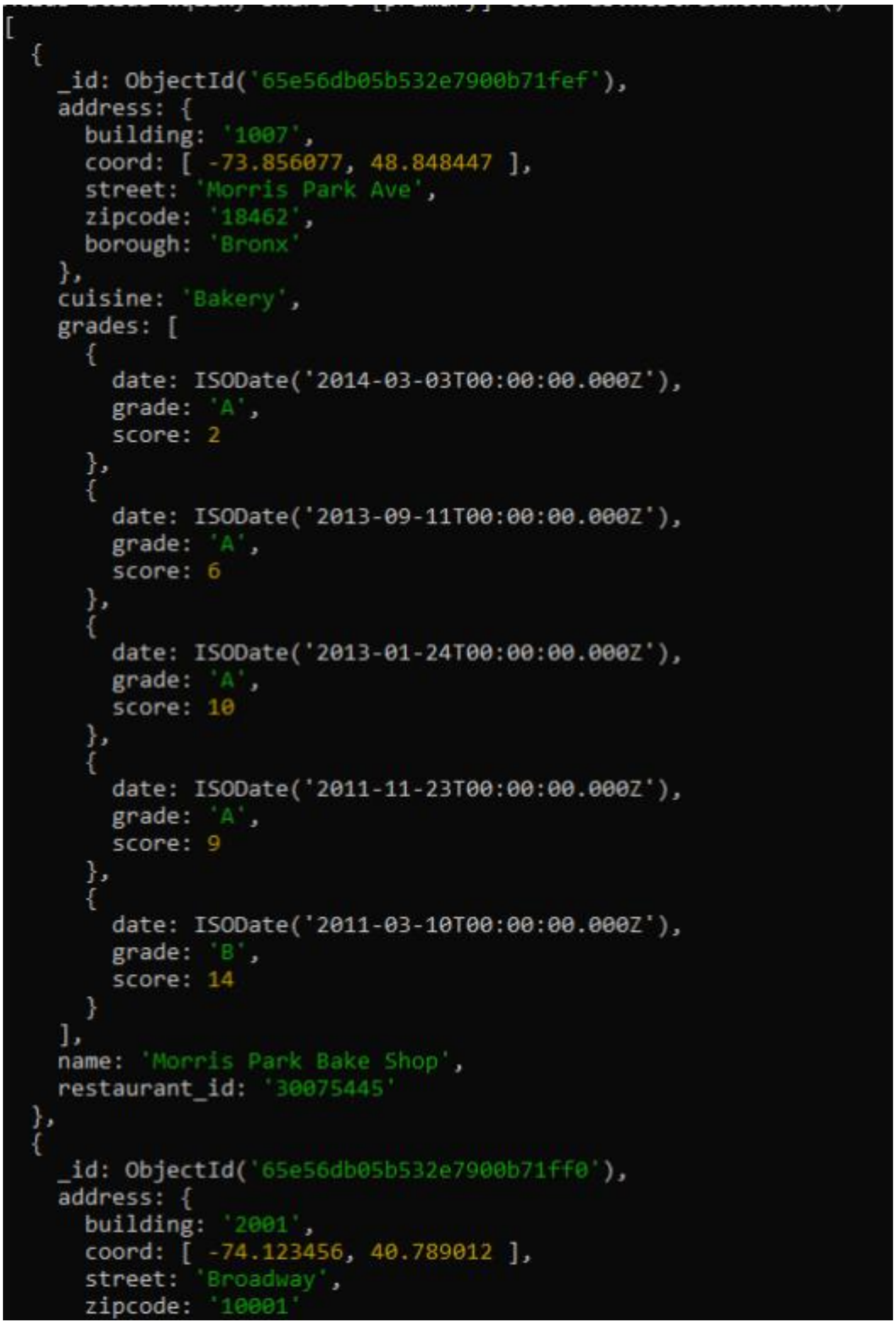
```
]
Atlas atlas-wqilky-shard-0 [primary] test> db.createCollection("Restraunt");
{ ok: 1 }
```

Inserting Values:

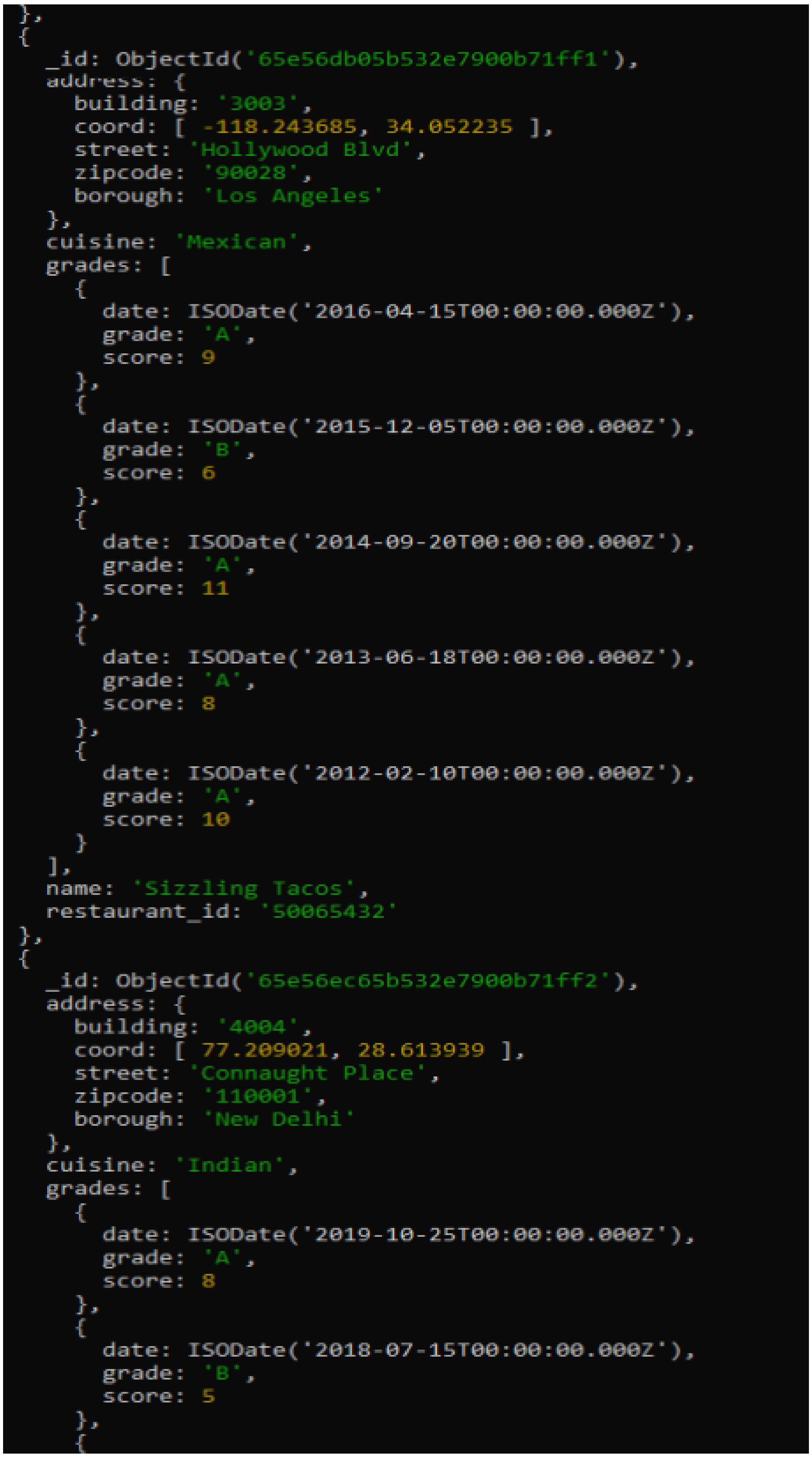
```
db.Restraunt.insertMany([
{
"address": {
"building": "1007",
"coord": [-73.856077, 48.848447],
"street": "Morris Park Ave",
"zipcode": "18462",
"borough": "Bronx"
},
"cuisine": "Bakery",
"grades": [
{"date": new Date("2014-03-03"), "grade": "A", "score": 2},
{"date": new Date("2013-09-11"), "grade": "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"
}]);
```

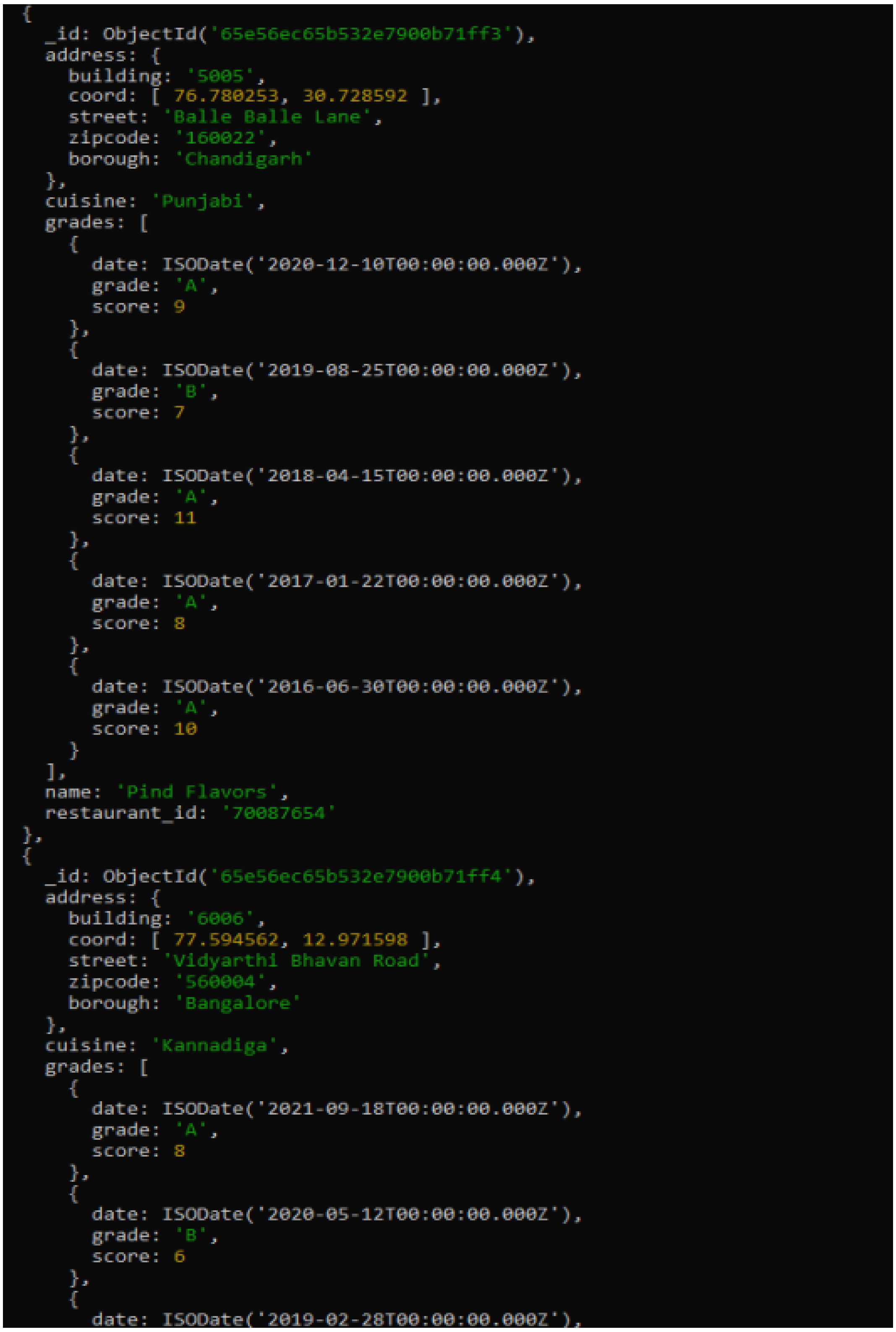
Queries:
1.db. Restraunt.find();



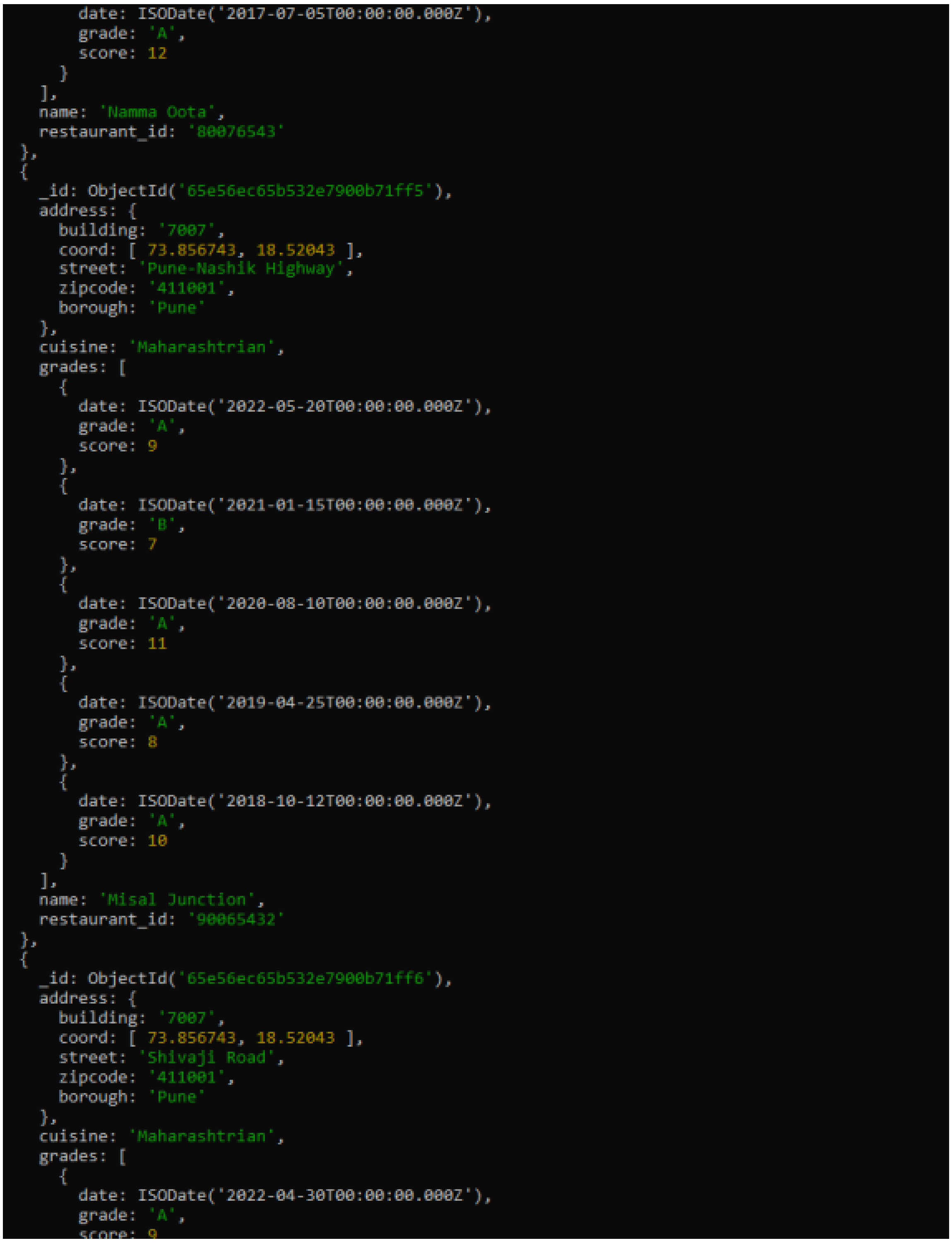
```
[
  {
    _id: ObjectId('65e56db05b532e7900b71fef'),
    address: {
      building: '1007',
      coord: [ -73.856077, 48.848447 ],
      street: 'Morris Park Ave',
      zipcode: '18462',
      borough: 'Bronx'
    },
    cuisine: 'Bakery',
    grades: [
      {
        date: ISODate('2014-03-03T00:00:00.000Z'),
        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'
```



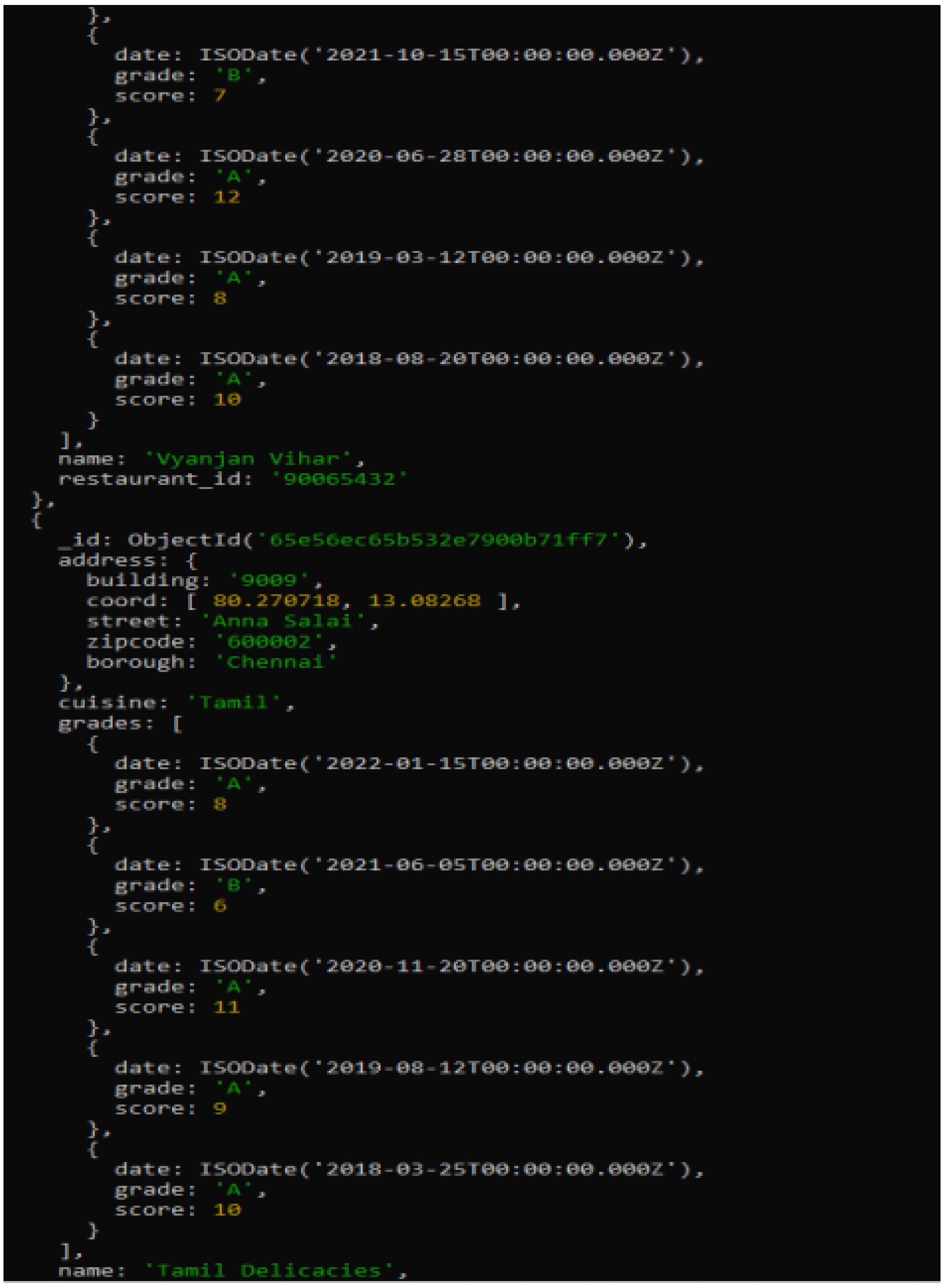
```
},
{
  _id: ObjectId('65e56db05b532e7900b71ff1'),
  address: {
    building: '3003',
    coord: [ -118.243685, 34.052235 ],
    street: 'Hollywood Blvd',
    zipcode: '90028',
    borough: 'Los Angeles'
  },
  cuisine: 'Mexican',
  grades: [
    {
      date: ISODate('2016-04-15T00:00:00.000Z'),
      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'),
```

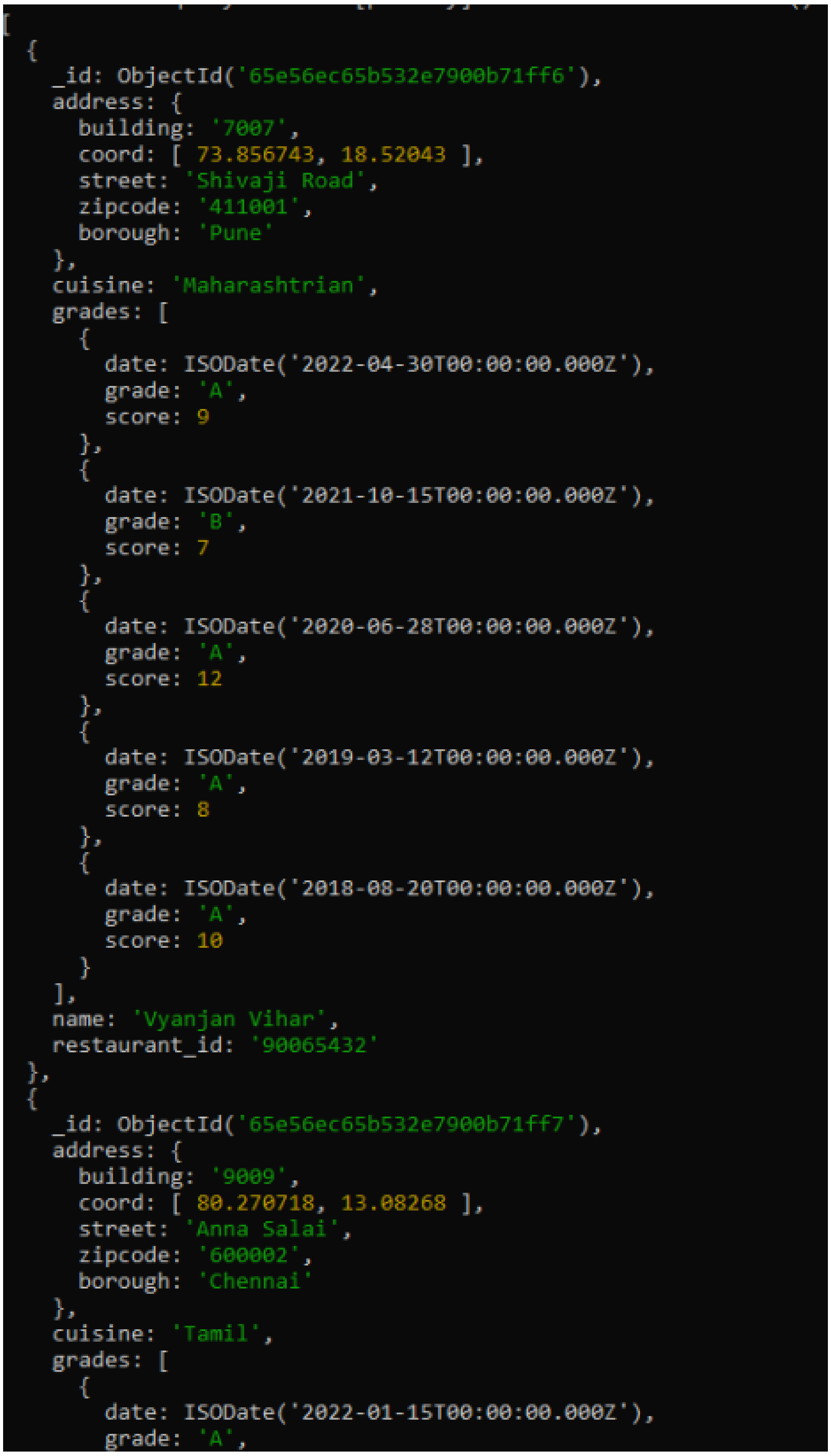


```
      date: ISODate('2017-07-05T00:00:00.000Z'),
      grade: 'A',
      score: 12
    }
  ],
  name: 'Namma Oota',
  restaurant_id: '80076543'
},
{
  _id: ObjectId('65e56ec65b532e7900b71ff5'),
  address: {
    building: '7007',
    coord: [ 73.856743, 18.52043 ],
    street: 'Pune-Nashik Highway',
    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
```

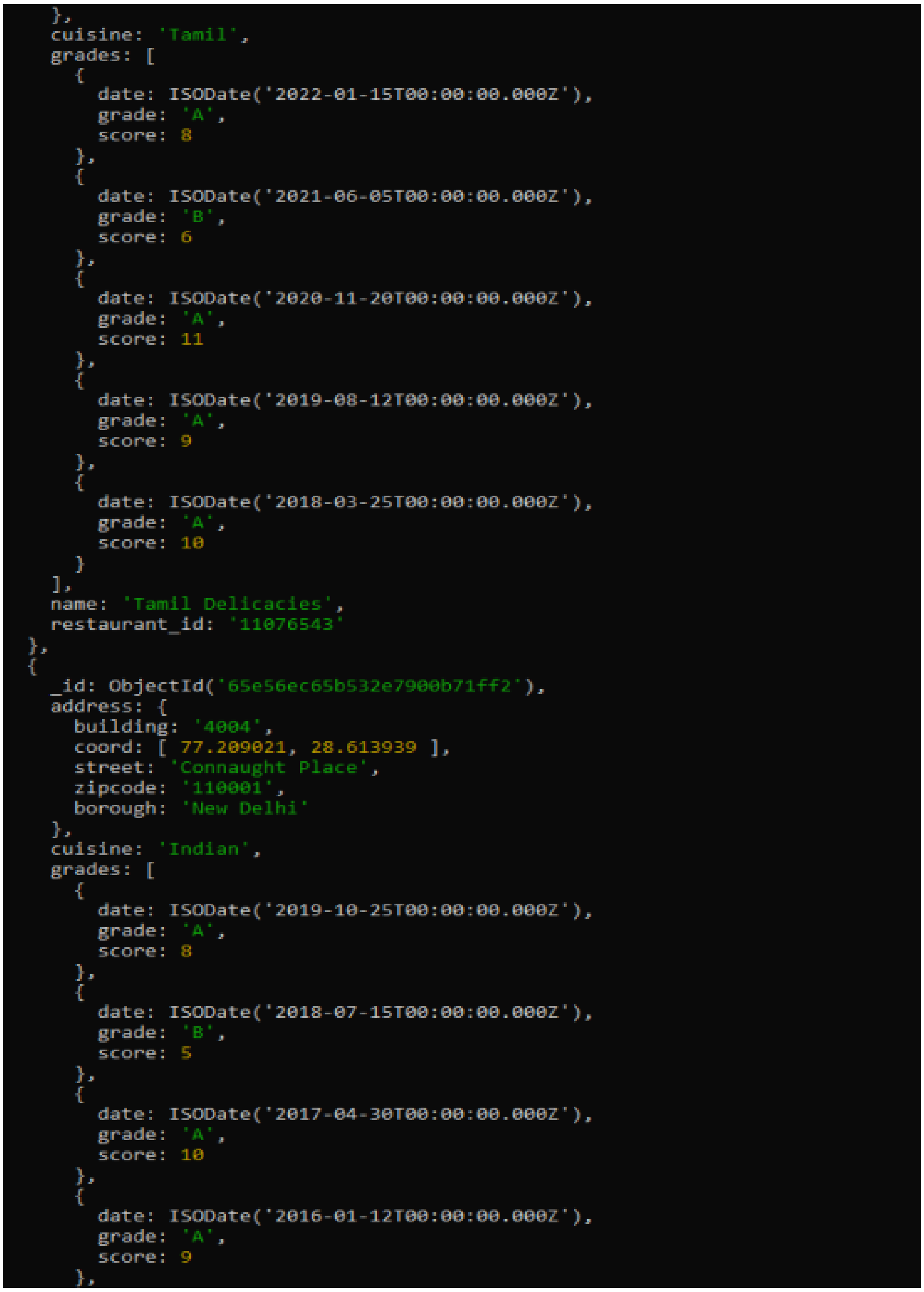


```
      },
      {
        date: ISODate('2021-10-15T00:00:00.000Z'),
        grade: 'B',
        score: 7
      },
      {
        date: ISODate('2020-06-28T00:00:00.000Z'),
        grade: 'A',
        score: 12
      },
      {
        date: ISODate('2019-03-12T00:00:00.000Z'),
        grade: 'A',
        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',
```

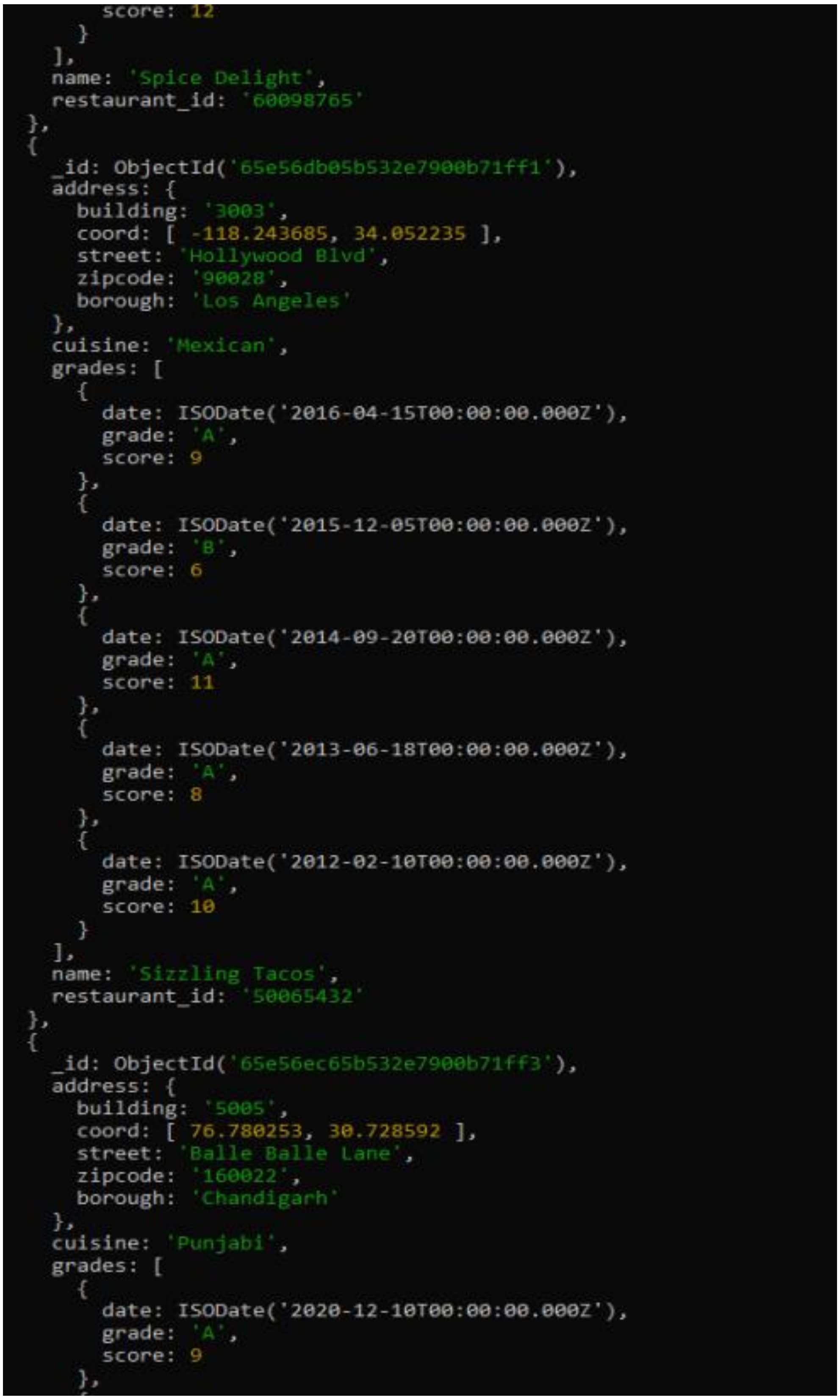
2) db.Restraunt.find().sort({ "name": -1 });



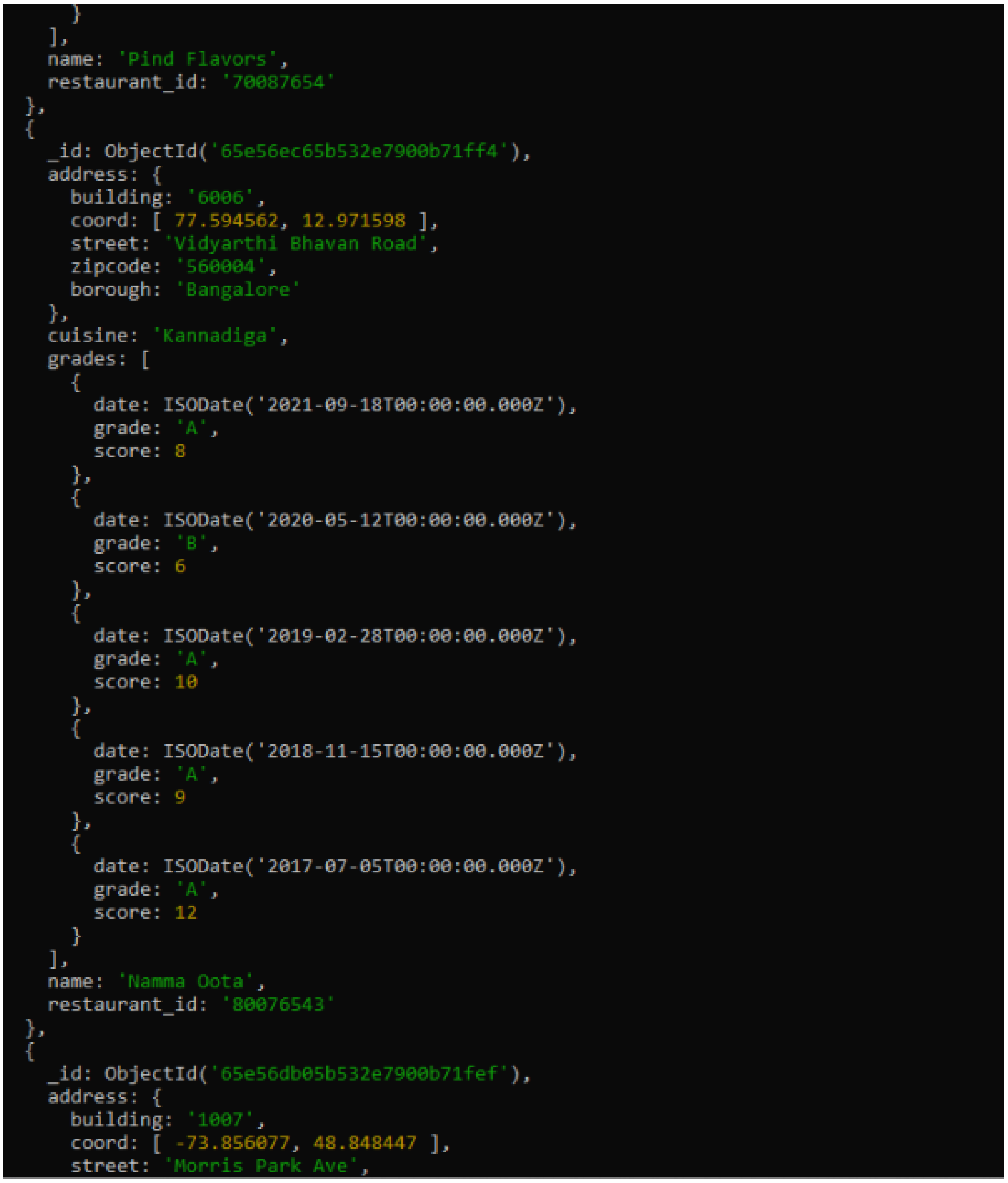
```
[
  {
    _id: ObjectId('65e56ec65b532e7900b71ff6'),
    address: {
      building: '7007',
      coord: [ 73.856743, 18.52043 ],
      street: 'Shivaji Road',
      zipcode: '411001',
      borough: 'Pune'
    },
    cuisine: 'Maharashtrian',
    grades: [
      {
        date: ISODate('2022-04-30T00:00:00.000Z'),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



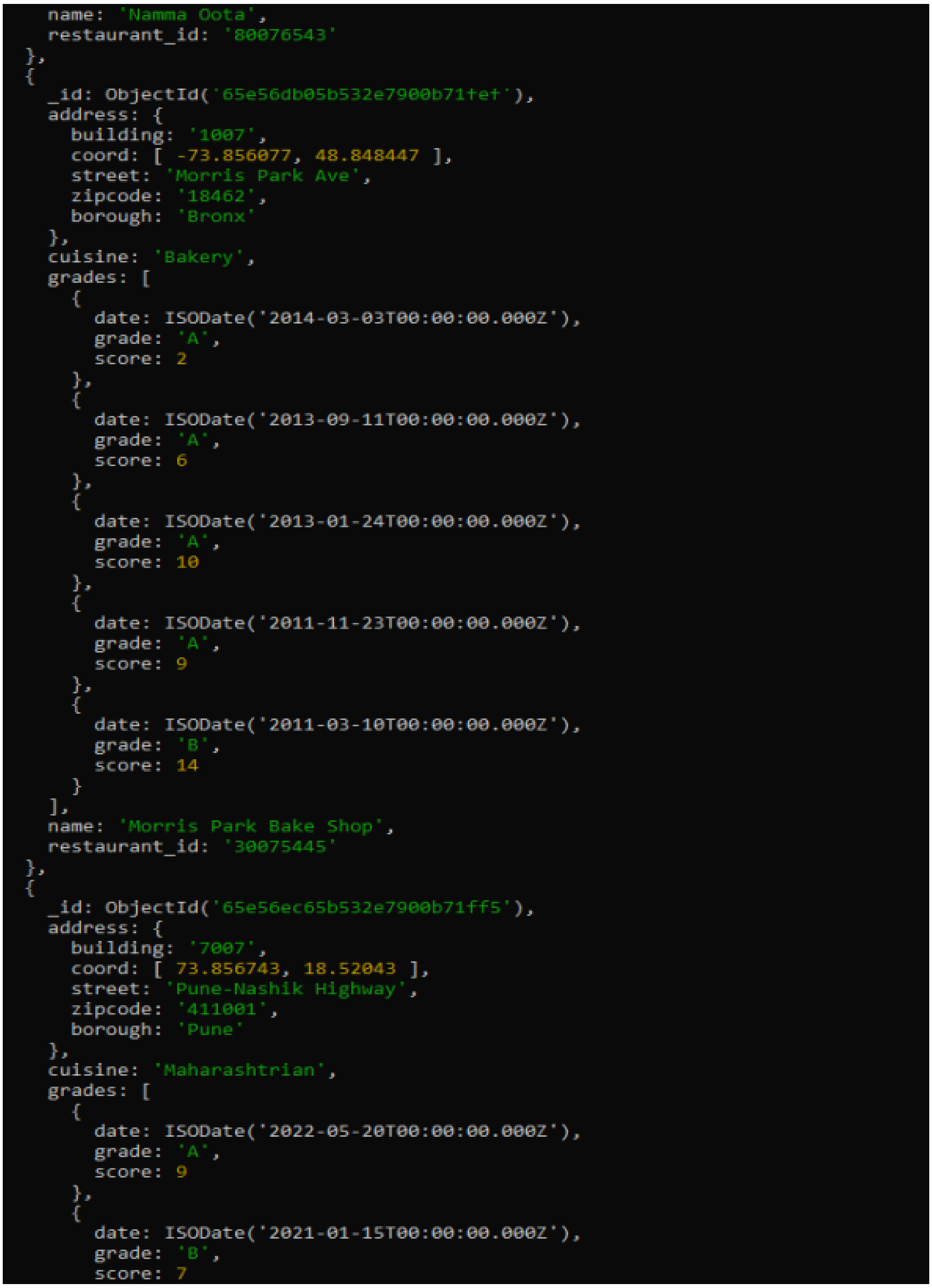
```
  },
  cuisine: 'Tamil',
  grades: [
    {
      date: ISODate('2022-01-15T00:00:00.000Z'),
      grade: 'A',
      score: 8
    },
    {
      date: ISODate('2021-06-05T00:00:00.000Z'),
      grade: 'B',
      score: 6
    },
    {
      date: ISODate('2020-11-20T00:00:00.000Z'),
      grade: 'A',
      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
    },
```



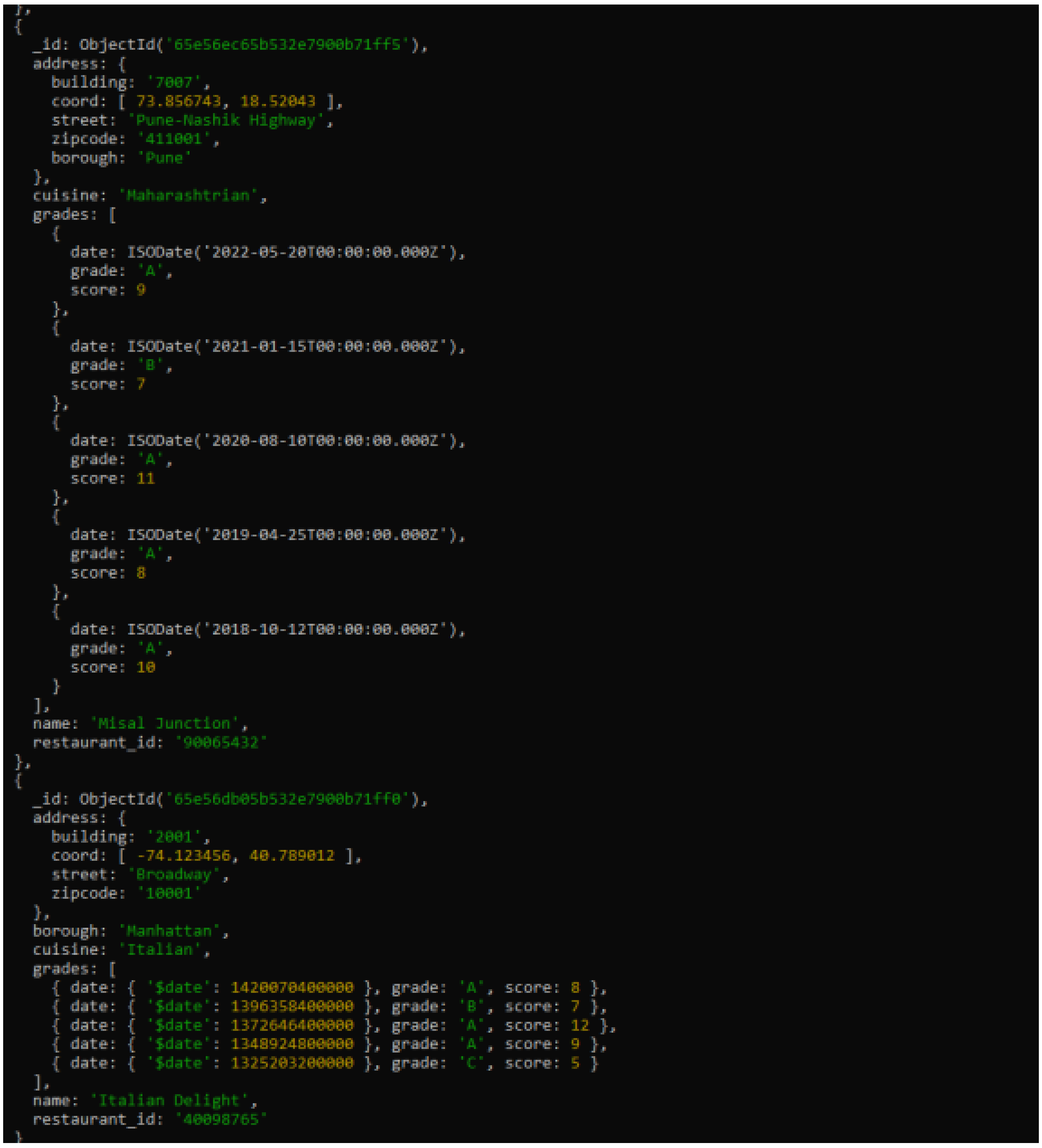
```
        score: 12
      }
    ],
    name: 'Spice Delight',
    restaurant_id: '60098765'
  },
  {
    _id: ObjectId('65e56db05b532e7900b71ff1'),
    address: {
      building: '3003',
      coord: [ -118.243685, 34.052235 ],
      street: 'Hollywood Blvd',
      zipcode: '90028',
      borough: '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
      },
```



```
    }
  ],
  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, 40.848447 ],
    street: 'Morris Park Ave',
```

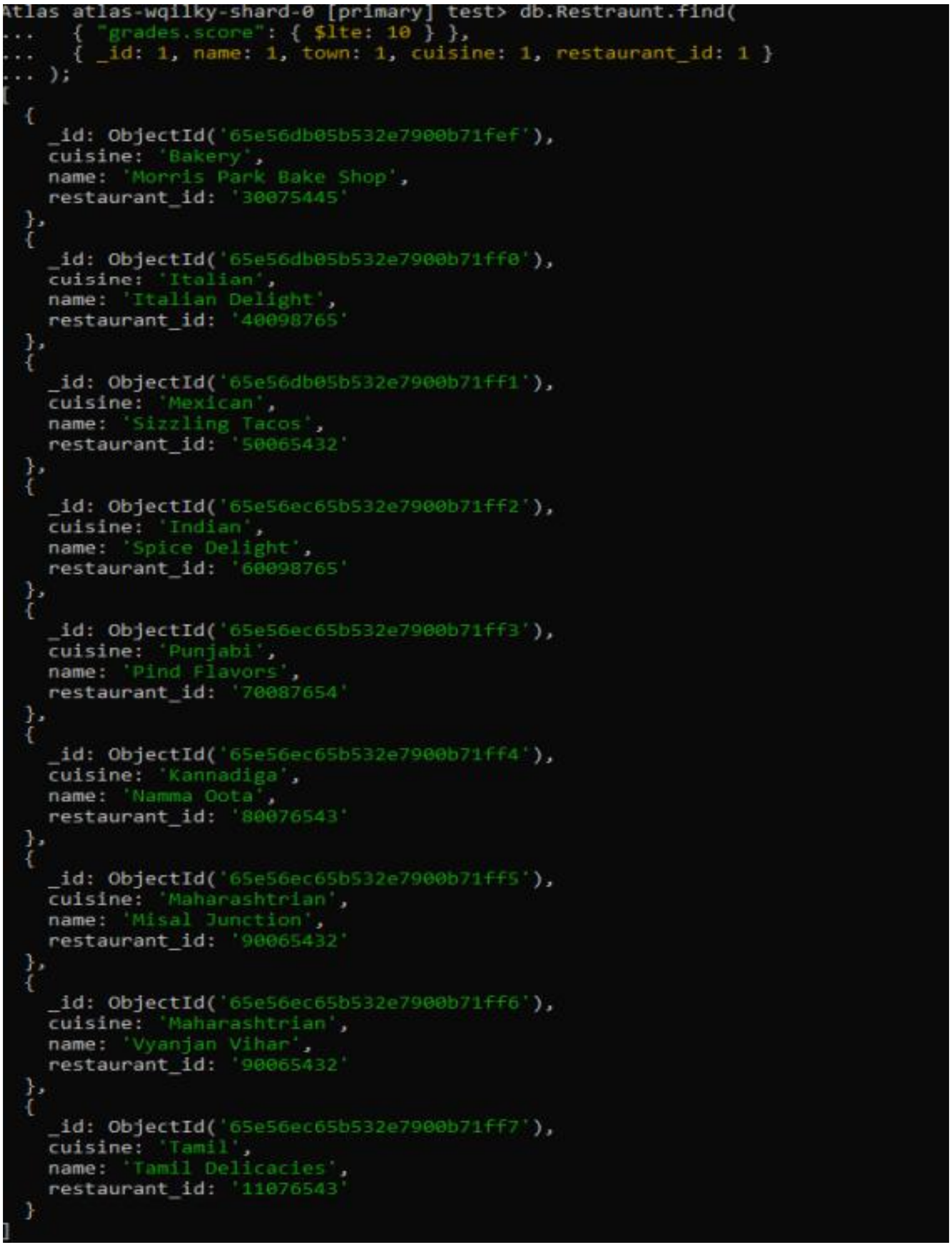


```
    name: 'Namma Oota',
    restaurant_id: '80076543'
  },
  {
    _id: ObjectId('65e56db05b532e7900b71fef'),
    address: {
      building: '1007',
      coord: [ -73.856077, 40.848447 ],
      street: 'Morris Park Ave',
      zipcode: '10462',
      borough: 'Bronx'
    },
    cuisine: 'Bakery',
    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'
  },
  {
    _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'
}
```

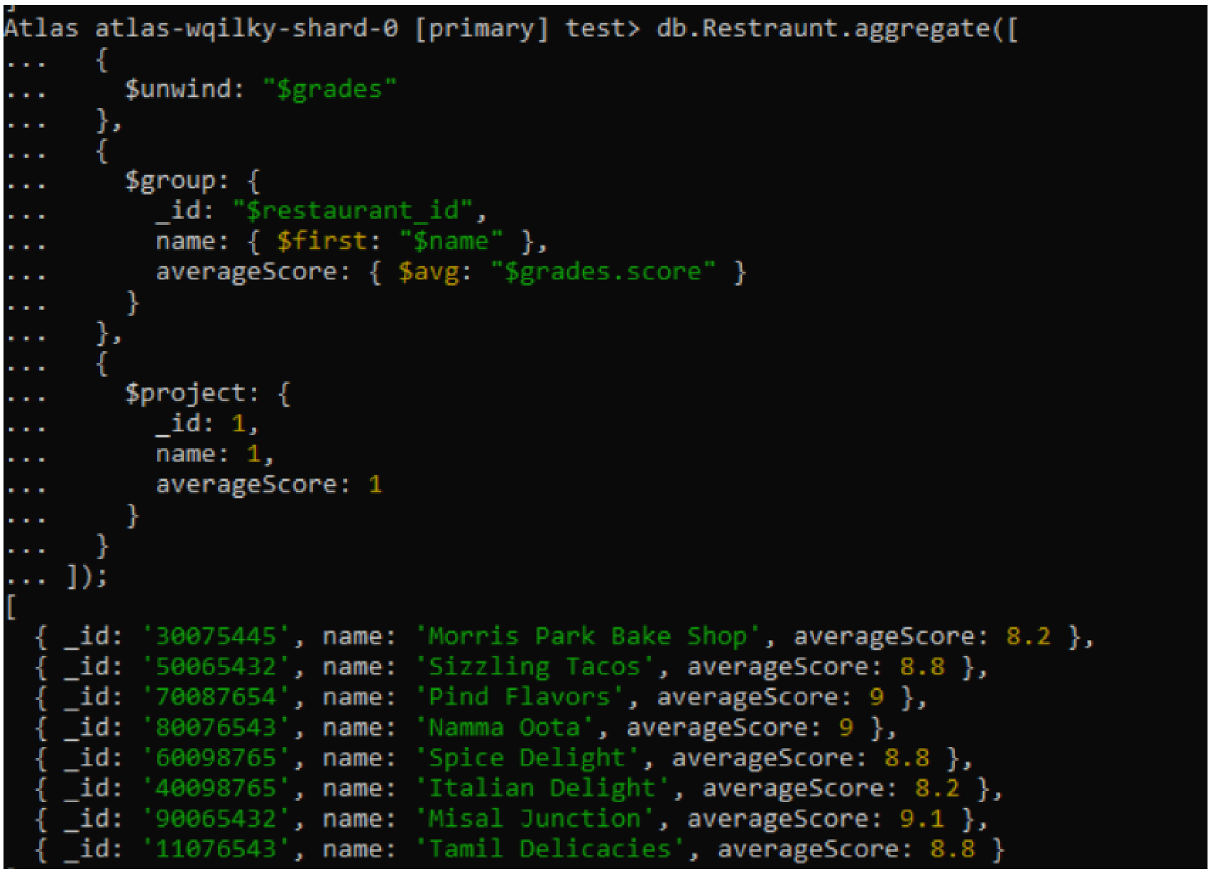
3) db.Restraunt.find({ "grades.score": { $lte: 10 } }, { _id: 1, name: 1, town: 1, cuisine: 1, restaurant_id: 1 });



```
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'
  }
]
```

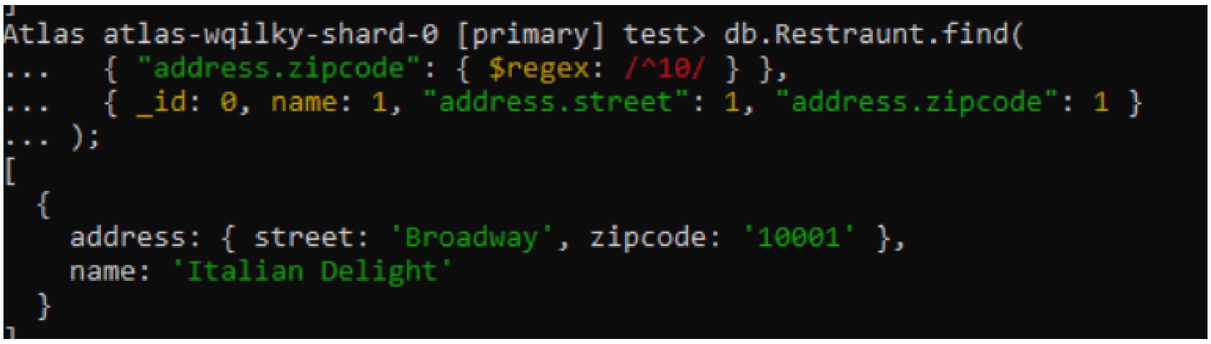
4) db.Restraunt.aggregate ([{ $ unwind : " $ grades " } ,

{ $group: { _id: "$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'
  }
```