VISVESVARAYA TECHNOLOGICAL UNIVERSITY

"JnanaSangama", Belgaum -590014, Karnataka.

LAB REPORT
on

DATABASE MANAGEMENT SYSTEMS

(23CS3PCDBM)

Submitted by

SANJANA SURESH(1BM22CS239)

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
December-2023 to March-2024

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" carried out by **SANJANA SURESH(1BM22CS239),** 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.

Name of the Lab-In charge : Vikranth BM
Assistant Professor
Department of CSE
BMSCE, Bengaluru
`

Dr. Jyothi S Nayak
Professor and Head
Department of CSE
BMSCE, Bengaluru

INDEX

1. Insurance Database

PROGRAM 1: INSURANCE DATABASE

Consider the Insurance database given below:

PERSON (driver_id: String, name: String, address: String)

CAR (reg_num: String, model: String, year: int)

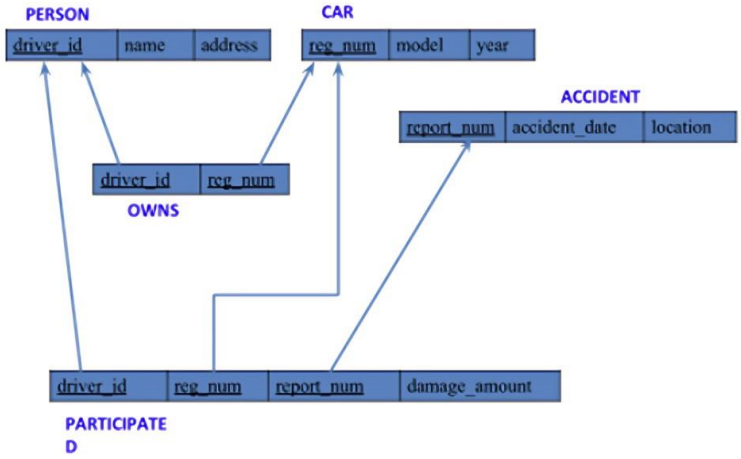
ACCIDENT (report_num: int, accident_date: date, location: String)

OWNS (driver_id: String, reg_num: String)

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

i. Create the above tables by properly specifying the primary keys and the foreign keys.
ii. Enter at least five tuples for each relation.
iii. Display Accident date and location.
iv. Update the damage amount to 25000 for the car with a specific reg_num (example 'KA053408') for which the accident report number was 12.
v. Add a new accident to the database.
vi. Display driver id who did accident with damage amount greater than or equal to Rs.25000.

Schema Diagram:



Creating Database and Table:

```
create database insurance_239;
use insurance_239;

Create table person(
driver_id varchar(20),
name varchar(30),
address varchar(50),
PRIMARY KEY(driver_id) );
Create table car(
reg_num varchar(15),
model varchar(10),
year int,
PRIMARY KEY(reg_num)
```

```
);
Create table owns(
driver_id varchar(20),
reg_num varchar(10),
PRIMARY KEY(driver_id, reg_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num)
);
Create table accident(
report_num int,
accident_date date,
location varchar(50),
PRIMARY KEY(report_num)
);
Create table participated(
driver_id varchar(20),
reg_num varchar(10),
report_num int,
damage_amount int,
PRIMARY KEY(driver_id,reg_num,report_num),
FOREIGN KEY(driver_id) REFERENCES person(driver_id),
FOREIGN KEY(reg_num) REFERENCES car(reg_num),
FOREIGN KEY(report_num) REFERENCES accident(report_num)
);
```

Inserting Values to the table :

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

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

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

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

```
insert into owns values("A01","KA052250");
insert into owns values("A02","KA031181");
insert into owns values("A03","KA095477");
insert into owns values("A04","KA053408");
insert into owns values("A05","KA041702");
select * from owns;
```

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

```
insert into accident values(11,'2003-01-01',"Mysore Road");
insert into accident values(12,'2004-02-02',"South end Circle");
insert into accident values(13,'2003-01-21',"Bull temple Road");
insert into accident values(14,'2008-02-17',"Mysore Road");
insert into accident values(15,'2004-03-05',"Kanakpura Road");
select * from accident;
```

report_num	accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road
NULL	NULL	NULL

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

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

Queries :

iii. Display accident date and location .

```
select accident_date, location from accident;
```

accident_date	location
2003-01-01	Mysore Road
2004-02-02	South end Circle
2003-01-21	Bull temple Road
2008-02-17	Mysore Road
2004-03-05	Kanakpura Road

iv. Update the damage amount to 25000 for the car with a specific reg-num (example 'KA053408') for which the accident report number was 12.

```
update participated
set damage_amount=25000
where reg_num='KA053408' and report_num=12;
select * from participated where reg_num='KA053408' and report_num=12;
```

driver_id	reg_num	report_num	damage_amount
A02	KA053408	12	25000
NULL	NULL	NULL	NULL

v. Add a new accident to the database.

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

report_num	accident_date	location
11	2003-01-01	Mysore Road
12	2004-02-02	South end Circle
13	2003-01-21	Bull temple Road
14	2008-02-17	Mysore Road
15	2004-03-05	Kanakpura Road
16	2008-03-08	Domlur
NULL	NULL	NULL

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

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

driver_id
A02
A03

2. More Queries on Insurance Database

PROGRAM 2. More Queries on 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 as done in "Program 1"week's lab and Enter at least five tuples for each relation.

i. Display the entire CAR relation in the ascending order of manufacturing year.
ii. Find the number of accidents in which cars belonging to a specific model (example 'Lancer') were involved.
iii. Find the total number of people who owned cars that involved in accidents in 2008.
iv. List the Entire Participated Relation in the Descending Order of Damage Amount. Find the Average Damage Amount.
v. Delete the Tuple Whose Damage Amount is below the Average Damage Amount.
vi. List the Name of Drivers Whose Damage is Greater than The Average Damage Amount.
vii. Find Maximum Damage Amount.

Creating database and table:

Database insurance_239 and tables as per schema were created in the previous lab and it is as shown in the previous experiment.

Queries :

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

```
select * from car 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

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

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

count(report_num)
1

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

```
select count(distinct driver_id) CNT
from participated a, accident b
where a.report_num=b.report_num and b.accident_date like '__08%';
```

CNT
1

iv. List the entire participated relation in the descending order of damage amount.

```
select * from participated order by damage_amount desc;
```

Find the average damage amount .

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

AVG(damage_amount)
13600.0000

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

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

```
delete from participated  where damage_amount < (select
p.damage_amount from(select AVG(damage_amount) as
damage_amount FROM participated )p);
select * from participated;
```

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

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

```
select name from person p, participated part where p.driver_id=part.driver_id and
damage_amount>(select AVG(damage_amount) FROM participated);
```

name

vii. Find maximum damage amount.

```
select MAX(damage_amount) from participated;
```

MAX(damage_amount)
25000

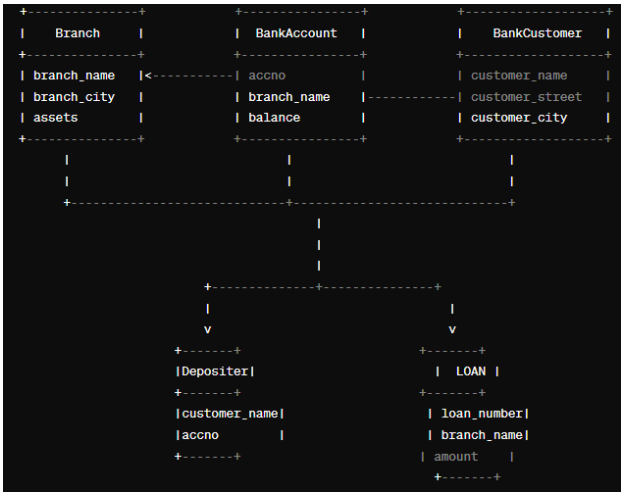
3. Bank Database

PROGRAM 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)

i. Create the above tables by properly specifying the primary keys and the foreign keys.
ii. Enter at least five tuples for each relation.
iii. Display the branch name and assets from all branches in lakhs of rupees and rename the assets column to 'assets in lakhs'.
iv. Find all the customers who have at least two accounts at the same branch (ex. SBI_ResidencyRoad).
v. Create A View Which Gives Each Branch the Sum of The Amount of All The Loans At The Branch.

Schema Diagram :



Creating Database and Table:

```
create database bank_239;
use bank_239;

Create table branch(
Branch_name varchar(30),
Branch_city varchar(25),
assets int,
PRIMARY KEY (Branch_name)
);
Create table BankAccount(
Accno int,
Branch_name varchar(30),
Balance int,
PRIMARY KEY(Accno),
foreign key (Branch_name) references branch(Branch_name)
);
Create table BankCustomer(
Customername varchar(20),
Customer_street varchar(30),
CustomerCity varchar (35),
```

```
PRIMARY KEY(Customername)
);
Create table Depositer(
Customername varchar(20),
Accno int,
PRIMARY KEY(Customername,Accno),
foreign key (Accno) references BankAccount(Accno),
foreign key (Customername) references BankCustomer(Customername)
);

Create table Loan(
Loan_number int,
Branch_name varchar(30),
Amount int,
PRIMARY KEY(Loan_number),
foreign key (Branch_name) references branch(Branch_name)
);
```

Inserting Values to the table :

```
insert into branch values("SBI_Chamrajpet","Bangalore",50000);
insert into branch values("SBI_ResidencyRoad","Bangalore",10000);
insert into branch values("SBI_ShivajiRoad","Bombay",20000);
insert into branch values("SBI_ParlimentRoad","Delhi",10000);
insert into branch values("SBI_Jantarmantar","Delhi",20000);
select * from branch;
```

Branch_name	Branch_city	assets
SBI_Chamrajpet	Bangalore	50000
SBI_Jantarmantar	Delhi	20000
SBI_ParlimentRoad	Delhi	10000
SBI_ResidencyRoad	Bangalore	10000
SBI_ShivajiRoad	Bombay	20000
NULL	NULL	NULL

```
insert into BankAccount values(1,"SBI_Chamrajpet",2000);
insert into BankAccount values(2,"SBI_ResidencyRoad",5000);
insert into BankAccount values(3,"SBI_ShivajiRoad",6000);
insert into BankAccount values(4,"SBI_ParlimentRoad",9000);
insert into BankAccount values(5,"SBI_Jantarmantar",8000);
insert into BankAccount values(6,"SBI_ShivajiRoad",4000);
insert into BankAccount values(8,"SBI_ResidencyRoad",4000);
insert into BankAccount values(9,"SBI_ParlimentRoad",3000);
insert into BankAccount values(10,"SBI_ResidencyRoad",5000);
insert into BankAccount values(11,"SBI_Jantarmantar",2000);
select * from BankAccount;
```

Accno	Branch_name	Balance
1	SBI_Chamrajpet	2000
2	SBI_ResidencyRoad	5000
3	SBI_ShivajiRoad	6000
4	SBI_ParlimentRoad	9000
5	SBI_Jantarmantar	8000
6	SBI_ShivajiRoad	4000
8	SBI_ResidencyRoad	4000
9	SBI_ParlimentRoad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000
NULL	NULL	NULL

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

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

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

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

```
insert into Loan values(1,"SBI_Chamrajpet",1000);
insert into Loan values(2,"SBI_ResidencyRoad",2000);
insert into Loan values(3,"SBI_ShivajiRoad",3000);
insert into Loan values(4,"SBI_ParlimentRoad",4000);
insert into Loan values(5,"SBI_Jantarmantar",5000);
select * from Loan;
```

Loan_number	Branch_name	Amount
1	SBI_Chamrajpet	1000
2	SBI_ResidencyRoad	2000
3	SBI_ShivajiRoad	3000
4	SBI_ParlimentRoad	4000
5	SBI_Jantarmantar	5000
NULL	NULL	NULL

Queries :

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

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

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

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

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

Customername
Dinesh

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

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

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

4. More Queries on Bank Database

PROGRAM 4: More Queries on 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)

i. Find all the customers who have an account at all the branches located in a specific city (Ex. Delhi).
ii. Find all customers who have a loan at the bank but do not have an account.
iii. Find all customers who have both an account and a loan at the Bangalore branch .
iv. Find the names of all branches that have greater assets than all branches located in Bangalore.
v. Demonstrate how you delete all account tuples at every branch located in a specific city (Ex. Bombay).
vi. Update the Balance of all accounts by 5%

Queries：

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

```
SELECT customer_name FROM BankCustomer WHERE customer_city = 'Delhi' AND NOT EXISTS ( SELECT
branch_name FROM Branch WHERE branch_city = 'Delhi' AND NOT EXISTS ( SELECT * FROM
BankAccount WHERE BankAccount.branch_name = Branch.branch_name AND
BankCustomer.customer_name = Depositer.customer_name ) );
```

customername
Nikil
Ravi

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

```
SELECT customer_name FROM BankCustomer WHERE EXISTS ( SELECT * FROM Loan WHERE
Loan.branch_name = Branch.branch_name AND NOT EXISTS ( SELECT * FROM BankAccount
WHERE BankAccount.branch_name = Branch.branch_name AND
BankCustomer.customer_name = Depositer.customer_name ) );
```

customername
Mohan

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

```
SELECT DISTINCT customer_name FROM BankCustomer WHERE EXISTS ( SELECT * FROM
BankAccount WHERE BankAccount.branch_name = 'SBI_ResidencyRoad' AND
BankCustomer.customer_name = Depositer.customer_name ) AND EXISTS ( SELECT * FROM
Loan WHERE Loan.branch_name = 'SBI_ResidencyRoad' AND BankCustomer.customer_name
= Depositer.customer_name );
```

customername
Avinash
Dinesh
Nikil
Ravi

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

```
SELECT branch_name FROM Branch WHERE assets > ALL ( SELECT assets FROM Branch
WHERE branch_city = 'Bangalore' );
```

branch_name

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

```
DELETE FROM BankAccount WHERE branch_name IN ( SELECT branch_name FROM
Branch WHERE branch_city = 'Bombay' );
select * from BankAccount;
```

Accno	Branch_name	Balance
1	SBI_Chamrajpet	2000
2	SBI_ResidencyRoad	5000
4	SBI_ParlimentRoad	9000
5	SBI_Jantarmantar	8000
8	SBI_ResidencyRoad	4000
9	SBI_ParlimentRoad	3000
10	SBI_ResidencyRoad	5000
11	SBI_Jantarmantar	2000
NULL	NULL	NULL

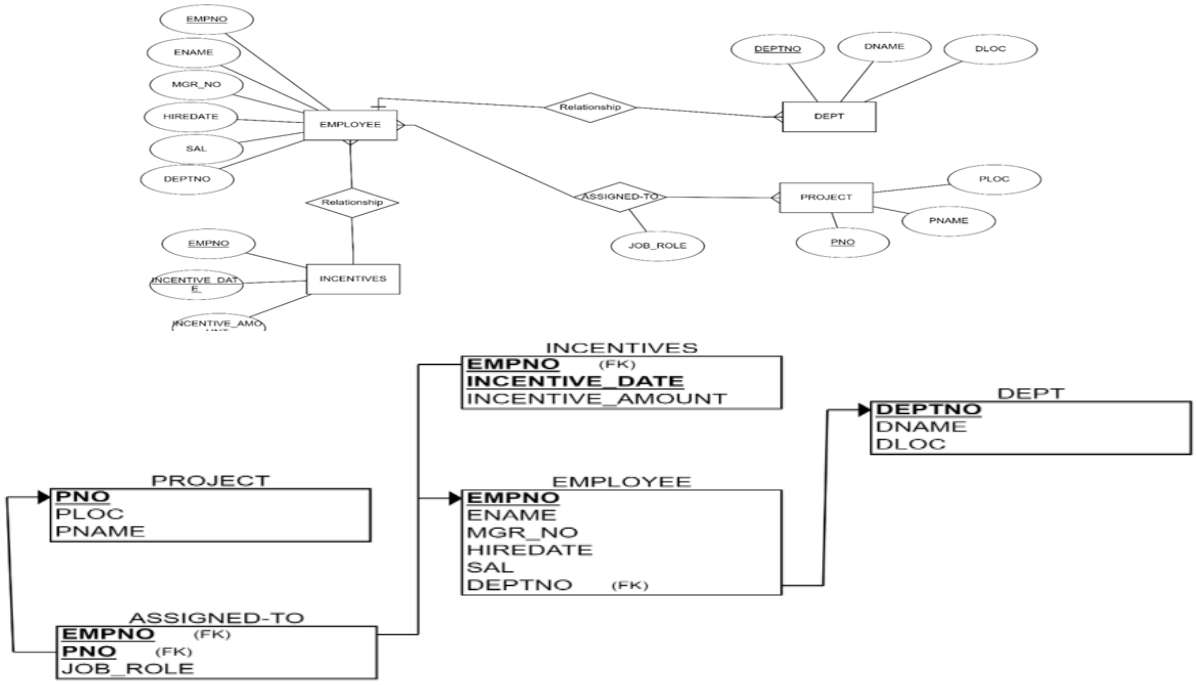
vi. Update the Balance of all accounts by 5%

```
UPDATE BankAccount set Balance=(Balance + (Balance*0.05));
```

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

5. Employee Database

PROGRAM 5: Employee Database



i. Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys.
ii. Enter greater than five tuples for each table.
iii. Retrieve the employee numbers of all employees who work on project located in Bengaluru, Hyderabad, or Mysuru.
iv. Get Employee IDs of those employees who didn't receive incentives.
v. 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.

Creating of database and tables:

```
create database employee_239;
use employee_239;

create table project(
pno int,
ploc varchar(40),
pname varchar(40),
PRIMARY KEY(pno)
);
create table dept(
deptno int,
dname varchar(40),
dloc varchar(40),
PRIMARY KEY(deptno)
);
create table employee(
empno int,
ename varchar(40),
mgr_no int,
hiredate date,
```

```
sal int,
deptno int,
primary key (empno),
foreign key (deptno) references dept(deptno)
);
create table incentives(
empno int,
incentive_date date,
incentive_amount int,
primary key(incentive_date),
foreign key (empno) references employee(empno)
);
create table assigned_to(
empno int,
pno int,
job_role varchar(50),
foreign key (pno) references project(pno),
foreign key (empno) references employee(empno)
);
```

Inserting values into the tables:

```
insert into project values(1,"Bengaluru","Syntax");
insert into project values(2,"Gujurat","Rolex");
insert into project values(3,"Mysuru","Hybrid");
insert into project values(4,"Hyderabad,","Synergy");
insert into project values(5,"Mumbai","Mercury");
select * from project;
```

pno	ploc	pname
1	Bengaluru	Syntax
2	Gujurat	Rolex
3	Mysuru	Hybrid
4	Hyderabad,	Synergy
5	Mumbai	Mercury
NULL	NULL	NULL

```
insert into dept values(10,"Sales","Bengaluru");
insert into dept values(20,"Finance","West Bengal");
insert into dept values(30,"Marketing","Bihar");
insert into dept values(40,"Purchase","Mumbai");
insert into dept values(50,"Research & Develeopment","Hyderabad");
select * from dept;
```

deptno	dname	dloc
10	Sales	Bengaluru
20	Finance	West Bengal
30	Marketing	Bihar
40	Purchase	Mumbai
50	Research & Develeopment	Hyderabad
NULL	NULL	NULL

```
insert into employee values(100,"Prannay",400,'2003-01-01',100000,10);
insert into employee values(200,"Farhaan",500,'2004-02-02',100500,50);
insert into employee values(300,"Sanika",100,'2003-01-21',200500,30);
insert into employee values(400,"Sakshi", NULL ,'2008-02-17',300500,40);
insert into employee values(500,"Nishith",300,'2004-03-05',200700,40);
insert into employee values(600,"Sohan",200,'2005-11-01',200000,20);
insert into employee values(700,"Mahima",200,'2005-11-21',200900,20);
select * from employee;
```

empno	ename	mgr_no	hiredate	sal	deptno
100	Prannay	400	2003-01-01	100000	10
200	Farhaan	500	2004-02-02	100500	50
300	Sanika	100	2003-01-21	200500	30
400	Sakshi	NULL	2008-02-17	300500	40
500	Nishith	300	2004-03-05	200700	40
600	Sohan	200	2005-11-01	200000	20
700	Mahima	200	2005-11-21	200900	20
NULL	NULL	NULL	NULL	NULL	NULL

```
insert into incentives values(100,'2012-02-17',6000);
insert into incentives values(200,'2012-05-21',7000);
insert into incentives values(400,'2012-07-25',6500);
insert into incentives values(500,'2013-04-19',7400);
insert into incentives values(600,'2013-08-08',8000);
select * from incentives;
```

empno	incentive_date	incentive_amount
100	2012-02-17	6000
200	2012-05-21	7000
400	2012-07-25	6500
500	2013-04-19	7400
600	2013-08-08	8000
NULL	NULL	NULL

```
insert into assigned_to values(100,1, "Project Manager");
```

```
insert into assigned_to values(200,1, "Resource Manager");
insert into assigned_to values(300,2, "Business Analyst");
insert into assigned_to values(400,3, "Business Analyst");
insert into assigned_to values(500,3, "Project Manager");
insert into assigned_to values(600,5, "Resource Manager");
select * from assigned_to;
```

empno	pno	job_role
100	1	Project Manager
200	1	Resource Manager
300	2	Business Analyst
400	3	Business Analyst
500	3	Project Manager
600	5	Resource Manager

Queries:

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

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

Employee_number
100
200
400
500

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

```
select e.empno from employee e where e.empno NOT IN
        (select i.empno from incentives i);
```

empno
700
300
NULL

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

```
select e.ename Emp_name, e.empno Emp_Number, d.dname Dept, a.job_role Job_Role, d.dloc
Department_Location, p.ploc Project_Location from project p, dept d, employee e, assigned_to a
where e.empno=a.empno and p.pno=a.pno and e.deptno=d.deptno and p.ploc=d.dloc;
```

Emp_name	Emp_Number	Dept	Job_Role	Department_Location	Project_Location
Prannay	100	Sales	Project Manager	Bengaluru	Bengaluru

6. More Queries on Employee Database

PROGRAM 6: More Queries on Employee Database

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

Queries:

iii. List the name of the managers with the maximum employees

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

ename
Farhaan

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

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

ename
Sakshi
Nishith

v. Find the name of the second top level managers of each department.

```
select ename from employee where empno in(select distinct mgr_no
from employee where empno in
(select distinct mgr_no from employee where empno in
(select distinct mgr_no from employee)));
```

ename
Prannay
Sanika
Sakshi

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

```
select * from employee where empno=
(select i.empno from incentives i
where i.incentive_amount= (select max(n.incentive_amount) from incentives n where
n.incentive_amount<(select max(inc.incentive_amount) from incentives inc where inc.incentive_date
between '2019-01-01' and '2019-12-31') and incentive_date between '2019-01-01' and '2019-12-31'));
```

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

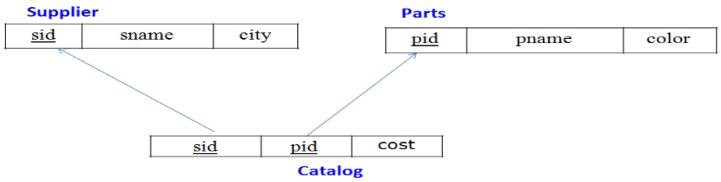
vii. Display those employees who are working in the same department where his manager is working.

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

ename

7. Supplier Database

PROGRAM 7: Supplier Database



i. Using Scheme diagram, create tables by properly specifying the primary keys and the foreign keys.
ii. Insert appropriate records in each table.
iii. Find the pnames of parts for which there is some supplier.
iv. Find the snames of suppliers who supply every part.
v. Find the snames of suppliers who supply every red part.
vi. Find the pnames of parts supplied by Acme Widget Suppliers and by no one else.
vii. 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).
viii. For each part, find the sname of the supplier who charges the most for that part.

Creating database and table:

```
create database supplier_239;
use supplier_239;

create table Supplier
        (sid int primary key,
        sname varchar(35),
        city varchar(35));

create table parts
        (pid int primary key,
        pname varchar(35),
        color varchar(35));

create table catalog
        (sid int,
        pid int,
        cost float,
        primary key(sid,pid),
        foreign key(sid) references Supplier(sid),
        foreign key(pid) references parts(pid));
```

Inserting values to tables:

```
insert into Supplier values
        (10001,"Acme Widget","Bangalore"),
        (10002,"Johns","Kolkata"),
        (10003,"Vimal","Mumbai"),
        (10004,"Reliance","Delhi");
Select * from  Supplier;
```

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

```
insert into parts values
```

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

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

```
insert into catalog values
          (10001,20001,10),
          (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;
```

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

Queries:

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

```
select distinct pname from parts p,catalog c where p.pid=c.pid;
```

pname
Book
Pen
Pencil
Mobile
Charger

ii. Find the snames of suppliers who supply every part.

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

sname
Acme Widget

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

```
select distinct sname from Supplier s,catalog c where s.sid=c.sid and pid in(select pid from
parts where color="red");
```

sname
Acme Widget
Johns

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

```
select pname from parts p,supplier s where pid in(select pid from catalog group by pid
having count(pid)=1) and s.sname="Acme Widget";
```

pname
Mobile
Charger

v. 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).

```
create view c as select c.pid,p.pname,avg(cost) as co from catalog c,parts p where
c.pid=p.pid group by c.pid;
select ca.sid from catalog ca,c where ca.pid=c.pid and ca.cost>c.co and c.pid=ca.pid;
```

sid
10002
10004

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

```
select sname,co.pid,pname,cost from Supplier s,parts po,catalog co where
co.pid=po.pid and s.sid=co.sid and co.cost =(select max(cost) from catalog where
pid=po.pid) ;
```

sname	pid	pname	cost
Acme Widget	20001	Book	10
Acme Widget	20004	Mobile	10
Acme Widget	20005	Charger	10
Johns	20001	Book	10
Johns	20002	Pen	20
Reliance	20003	Pencil	40

8. Flight Database

PROGRAM 8: Flight Database

FLIGHTS(flno: integer, from: string, to: string, distance: integer, departs: time, arrives: time, price: integer)
AIRCRAFT(aid: integer, aname: string, cruising_range: integer)
CERTIFIED(eid: integer, aid: integer)
EMPLOYEES(eid: integer, ename: string, salary: integer)

Note that the Employees relation describes pilots and other kinds of employees as well; Every pilot is certified for some aircraft, and only pilots are certified to fly.

i. Create database table and insert appropriate data.
ii. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.
iii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.
iv. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.
v. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the Average salary of all pilots certified for this aircraft.
vi. Find the names of pilots certified for some Boeing aircraft.
vii. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

Creating database and inserting values:

```
CREATE database Flight_239;
Use Flight_239;

CREATE TABLE FLIGHTS (
   flno INTEGER,
   from1 VARCHAR(50),
   to VARCHAR(50),
   distance INTEGER,
   departs TIME,
   arrives TIME,
   price INTEGER
);

CREATE TABLE AIRCRAFT (
   aid INTEGER,
   aname VARCHAR(50),
   cruising_range INTEGER
);

CREATE TABLE CERTIFIED (
   eid INTEGER,
   aid INTEGER
);

CREATE TABLE EMPLOYEES (
   eid INTEGER,
   ename VARCHAR(50),
   salary INTEGER
);
```

```
INSERT INTO FLIGHTS VALUES
(1, 'Bengaluru', 'Frankfurt', 8000, '08:00:00', '14:00:00', 100000),
(2, 'Bengaluru', 'New Delhi', 2000, '10:00:00', '12:00:00', 30000),
(3, 'New Delhi', 'Bengaluru', 2000, '14:00:00', '16:00:00', 30000);
Select * from flights;
```

fino	from1	to1	distance	departs	arrives	price
1	Bengaluru	Frankfurt	8000	08:00:00	14:00:00	100000
2	Bengaluru	New Delhi	2000	10:00:00	12:00:00	30000
3	New Delhi	Bengaluru	2000	14:00:00	16:00:00	30000

```
INSERT INTO AIRCRAFT VALUES
(1, 'Boeing 747', 12000),
(2, 'Airbus A380', 15000),
(3, 'Boeing 737', 8000);
Select * from aircraft;
```

aid	aname	cruising_range
1	Boeing 747	12000
2	Airbus A380	15000
3	Boeing 737	8000

```
INSERT INTO CERTIFIED VALUES
(1, 1),
(2, 2),
(3, 3);
Select * from certified;
```

eid	aid
1	1
2	2
3	3

```
INSERT INTO EMPLOYEES VALUES
(1, 'John Doe', 90000),
(2, 'Jane Smith', 75000),
(3, 'Alice Johnson', 85000),
(4, 'Bob Brown', 80000),
(5, 'Charlie Davis', 82000);
Select * from employees;
```

eid	ename	salary
1	John Doe	90000
2	Jane Smith	75000
3	Alice Johnson	85000
4	Bob Brown	80000
5	Charlie Davis	82000

Queries:

ii. Find the names of aircraft such that all pilots certified to operate them have salaries more than Rs.80,000.

```
SELECT aname
FROM AIRCRAFT
WHERE aid NOT IN (
   SELECT aid
   FROM CERTIFIED
   JOIN EMPLOYEES ON CERTIFIED.eid = EMPLOYEES.eid
   WHERE salary <= 80000
);
```

aname
Boeing 747
Boeing 737

iii. For each pilot who is certified for more than three aircrafts, find the eid and the maximum cruising range of the aircraft for which she or he is certified.

```
SELECT eid, MAX(cruising_range) AS max_cruising_range
FROM CERTIFIED
JOIN AIRCRAFT ON CERTIFIED.aid = AIRCRAFT.aid
GROUP BY eid
HAVING COUNT(*) > 3;
```

eid	max_cruising_range

iv. Find the names of pilots whose salary is less than the price of the cheapest route from Bengaluru to Frankfurt.

```
SELECT ename
FROM EMPLOYEES
JOIN FLIGHTS ON EMPLOYEES.salary < FLIGHTS.price
WHERE "from" = 'Bengaluru' AND "to" = 'Frankfurt'
ORDER BY salary;
```

ename

v. For all aircraft with cruising range over 1000 Kms, find the name of the aircraft and the Average salary of all pilots certified for this aircraft.

```
SELECT aname, AVG(salary) AS avg_salary
FROM AIRCRAFT
JOIN CERTIFIED ON AIRCRAFT.aid = CERTIFIED.aid
JOIN EMPLOYEES ON CERTIFIED.eid = EMPLOYEES.eid
WHERE cruising_range > 1000
GROUP BY aname;
```

aname	avg_salary
Boeing 747	90000.0000
Airbus A380	75000.0000
Boeing 737	85000.0000

vi. Find the names of pilots certified for some Boeing aircraft.

```
SELECT DISTINCT ename
FROM EMPLOYEES
JOIN CERTIFIED ON EMPLOYEES.eid = CERTIFIED.eid
JOIN AIRCRAFT ON CERTIFIED.aid = AIRCRAFT.aid
WHERE aname LIKE 'Boeing%';
```

ename
John Doe
Alice Johnson

vii. Find the aids of all aircraft that can be used on routes from Bengaluru to New Delhi.

```
SELECT DISTINCT aid
FROM FLIGHTS
WHERE "from" = 'Bengaluru' AND "to" = 'New Delhi';
```

aid

9. NoSQL Lab-1

PROGRAM 9: NoSQL - STUDENT DATABASE

Perform the following DB operations using MongoDB.

1. Create a database “Student” with the following attributesRollno, 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.

Creating database:

```
db.createCollect(“Student”);
```

Queries:

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

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

2. Insert appropriate values .

```
db.Student.insert({rollno:1,age:21,cont:9876,email:"prannay@gmail.com"});
db.Student.insert({rollno:2,age:22,cont:9976,email:"sohan@gmail.com"});
db.Student.insert({rollno:3,age:21,cont:5576,email:"farhaan@gmail.com"});
db.Student.insert({rollno:4,age:20,cont:4476,email:"sakshi@gmail.com"});
db.Student.insert({rollno:5,age:23,cont:2276,email:"sanika@gmail.com"});
```

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

```
db.Students.updateOne(
  { Rollno: 10 },
  { $set: { Email_Id: "newemail@example.com" } }
)
```

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

```
db.Students.updateOne(
  { Rollno: 11 },
  { $set: { Name: "FEM" } }
)
```

5. Export the created table into local file system .

```
mongoexport mongodb+srv://dhiksha:<password>@cluster0.xbmgopf.mongodb.net/Lab_9 --
collection=Student -- out C:\Users\dhiks\Desktop\export\output.json
```

6. Drop the table .

```
db.Student.drop();
```

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

```
mongoimport mongodb+srv://dhiksha:<password>@cluster0.xbmgopf.mongodb.net/Lab_9 --
collection=new_Student – type json –file C:\Users\dhiks\Desktop\export\output.json
```

10. NoSQL Lab-2

PROGRAM 10: NoSQL CUSTOMER DATABASE

Perform the following DB operations using MongoDB.

1. Create a collection by name Customers with the following attributes. Cust_id, Acc_Bal, Acc_Type .
2. Insert at least 5 values into the table
3. Write a query to display those records whose total account balance is greater than 1200 of account type 'Z' for each customer_id.
4. Determine Minimum and Maximum account balance for each customer_id.
5. Export the created collection into local file system.
6. Drop the table.
7. Import a given csv dataset from local file system into mongodb collection.

Queries:

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

```
db.createCollection("Customers", {
  validator: {
    $jsonSchema: {
      bsonType: "object",
      required: ["Cust_id", "Acc_Bal", "Acc_Type"],
      properties: {
        Cust_id: {
          bsonType: "int",
          description: "must be an integer and is required"
        },
        Acc_Bal: {
          bsonType: "int",
          description: "must be an integer and is required"
        },
        Acc_Type: {
          bsonType: "string",
          description: "must be a string and is required"
        }
      }
    }
  }
})
```

11. **Insert at least 5 values into the table.**

```
db.Customers.insertMany([
  { Cust_id: 1, Acc_Bal: 1000, Acc_Type: 'X' },
  { Cust_id: 1, Acc_Bal: 1200, Acc_Type: 'Z' },
  { Cust_id: 2, Acc_Bal: 1500, Acc_Type: 'Z' },
  { Cust_id: 3, Acc_Bal: 800, Acc_Type: 'Y' },
  { Cust_id: 4, Acc_Bal: 1300, Acc_Type: 'Z' },
  { Cust_id: 4, Acc_Bal: 2000, Acc_Type: 'X' }
])
```

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

```
db.Customers.aggregate([
  {
    $match: { Acc_Type: 'Z' }
  },
  {
```

```
    $group: {
      _id: "$Cust_id",
      total_balance: { $sum: "$Acc_Bal" }
    }
  },
  {
    $match: { total_balance: { $gt: 1200 } }
  }
])
```

13. **Determine Minimum and Maximum account balance for each customer_id.**

```
db.Customers.aggregate([
  {
    $group: {
      _id: "$Cust_id",
      min_balance: { $min: "$Acc_Bal" },
      max_balance: { $max: "$Acc_Bal" }
    }
  }
])
```

14. **Export the created collection into local file system.**

```
mongoexport --db your_database --collection Customers --out /path/to/your/exported_file.json
```

15. **Drop the table.**

```
db.Customers.drop()
```

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

```
mongoimport --db your_database --collection Customers --type csv --headerline --file
/path/to/your/csv/file.csv
```