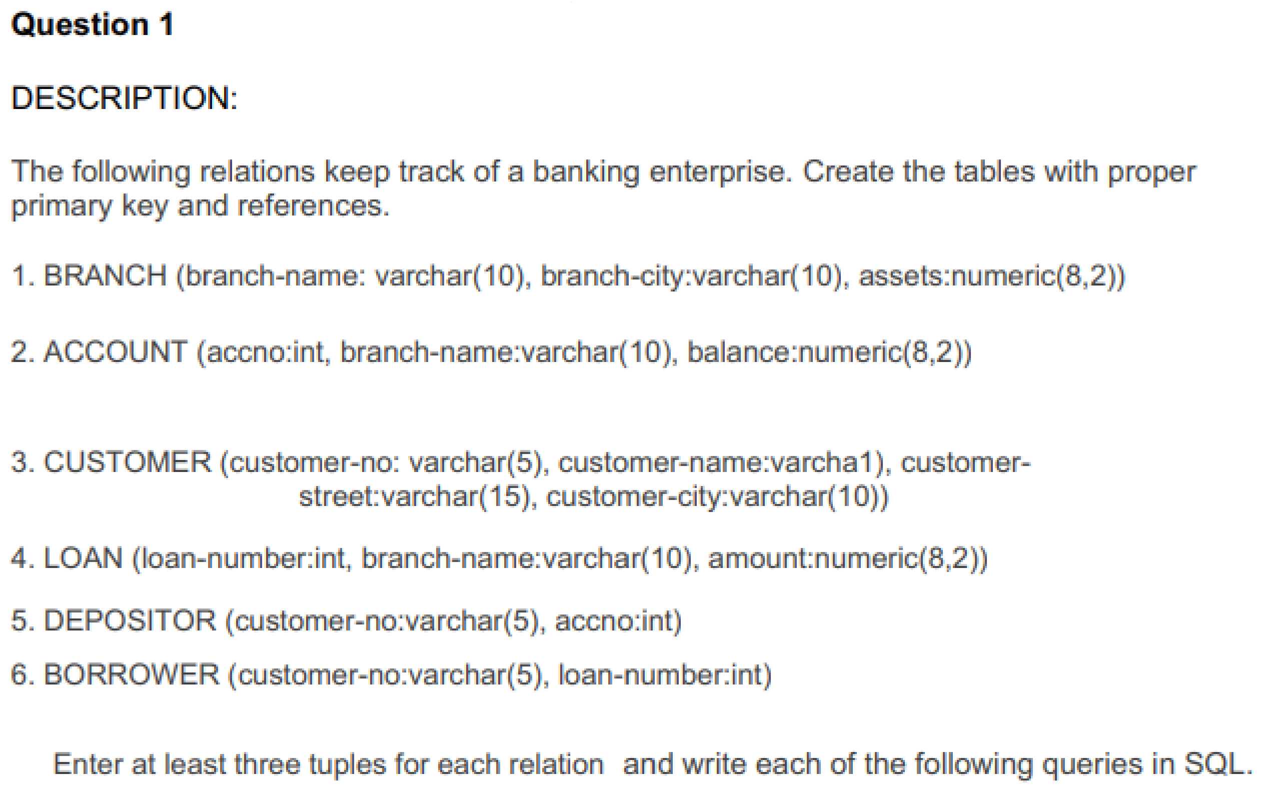
AIE303 Labsheet 4

Name: Aniketh Vijesh

Roll No: AM.EN.U4AIE22009



Question 1

DESCRIPTION:

The following relations keep track of a banking enterprise. Create the tables with proper primary key and references.

1. BRANCH (branch-name: varchar(10), branch-city:varchar(10), assets:numeric(8,2))

2. ACCOUNT (accno:int, branch-name:varchar(10), balance:numeric(8,2))

3. CUSTOMER (customer-no: varchar(5), customer-name:varcha1), customer-street:varchar(15), customer-city:varchar(10))

4. LOAN (loan-number:int, branch-name:varchar(10), amount:numeric(8,2))

5. DEPOSITOR (customer-no:varchar(5), accno:int)

6. BORROWER (customer-no:varchar(5), loan-number:int)

Enter at least three tuples for each relation and write each of the following queries in SQL.

```sql
CREATE TABLE BRANCH (
    branch_name VARCHAR(10) PRIMARY KEY,
    branch_city VARCHAR(10),
    assets NUMERIC(8, 2)
);

CREATE TABLE CUSTOMER (
    customer_no VARCHAR(5) PRIMARY KEY,
    customer_name VARCHAR(50),
    customer_street VARCHAR(15),
    customer_city VARCHAR(10)
);

CREATE TABLE ACCOUNT (
    accno INT PRIMARY KEY,
    customer_no VARCHAR(5),
    branch_name VARCHAR(10),
    balance NUMERIC(8, 2),
    FOREIGN KEY (customer_no) REFERENCES CUSTOMER(customer_no),
    FOREIGN KEY (branch_name) REFERENCES BRANCH(branch_name)
);

CREATE TABLE LOAN (
    loan_number INT PRIMARY KEY,
    branch_name VARCHAR(10),
    amount NUMERIC(8, 2),
    FOREIGN KEY (branch_name) REFERENCES BRANCH(branch_name)
);

CREATE TABLE DEPOSITOR (
    customer_no VARCHAR(5),
    accno INT,
    PRIMARY KEY (customer_no, accno),
    FOREIGN KEY (customer_no) REFERENCES CUSTOMER(customer_no),
    FOREIGN KEY (accno) REFERENCES ACCOUNT(accno)
);

CREATE TABLE BORROWER (
    customer_no VARCHAR(5),
    loan_number INT,
    PRIMARY KEY (customer_no, loan_number),
    FOREIGN KEY (customer_no) REFERENCES CUSTOMER(customer_no),
    FOREIGN KEY (loan_number) REFERENCES LOAN(loan_number)
);
```

```
-- Inserting data into BRANCH
INSERT INTO BRANCH (branch_name, branch_city, assets) VALUES
('Main', 'CityA', 500000.00),
('Kollam', 'CityB', 300000.00),
('Thiruvananthapuram', 'CityC', 250000.00);

-- Inserting data into CUSTOMER
INSERT INTO CUSTOMER (customer_no, customer_name, customer_street,
customer_city) VALUES
('C001', 'Alice Smith', 'Street 1', 'CityA'),
('C002', 'Bob Johnson', 'Street 2', 'CityB'),
('C003', 'Charlie Brown', 'Street 3', 'CityC');

-- Inserting data into ACCOUNT
INSERT INTO ACCOUNT (accno, customer_no, branch_name, balance) VALUES
(1001, 'C001', 'Main', 1500.00),
(1002, 'C001', 'Main', 2000.00),
(1003, 'C002', 'Kollam', 3000.00);
```

```sql
-- Inserting data into LOAN
INSERT INTO LOAN (loan_number, branch_name, amount) VALUES
(5001, 'Main', 10000.00),
(5002, 'Kollam', 20000.00),
(5003, 'Main', 15000.00);

-- Inserting data into DEPOSITOR
INSERT INTO DEPOSITOR (customer_no, accno) VALUES
('C001', 1001),
('C001', 1002),
('C002', 1003);

-- Inserting data into BORROWER
INSERT INTO BORROWER (customer_no, loan_number) VALUES
('C001', 5001),
('C002', 5002),
('C001', 5003);
```

1. Find all the customers who have at least two accounts at the '*Main*' branch.

2. Find the average account balance at the *'Kollam'* branch.

3. Find the number of depositors for each branch.

4. Find the names of all branches where the average account balance is more than RS. 1,2000.

5. Find all customers who have a loan, an account, or both.

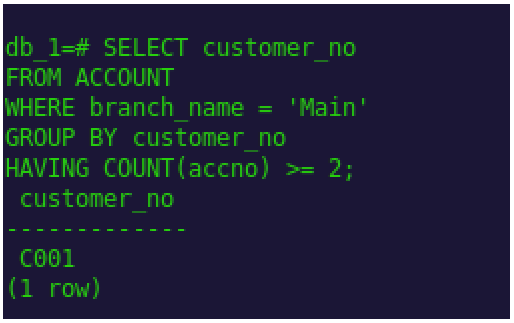
6. Find all customers who have both a loan and an account.

7. Find the number of branches that currently have loans.

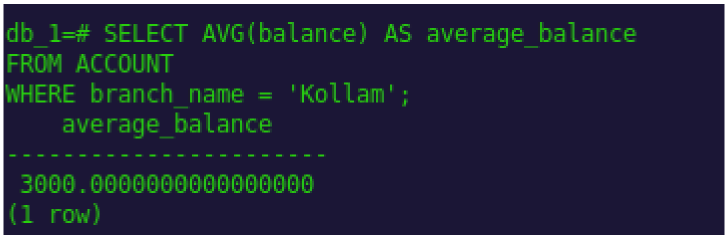
8. Find the average loan amount for each branch.

9. Find all customers with more than one loan.

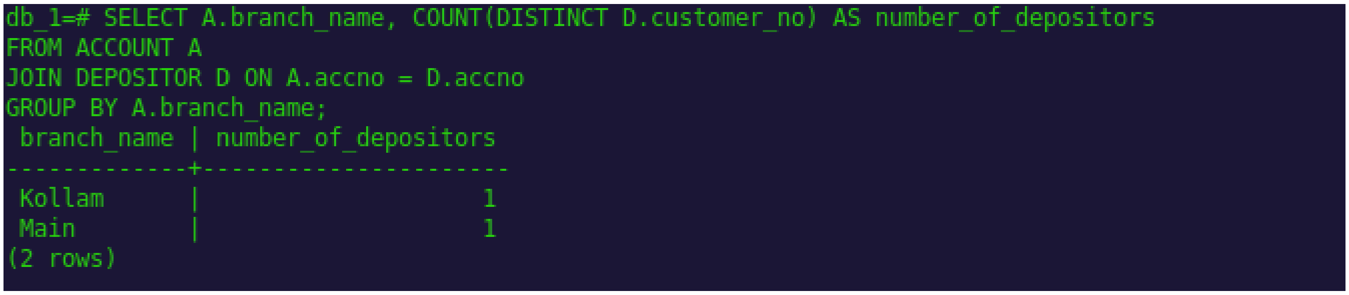
10. Find the total of all loan amounts



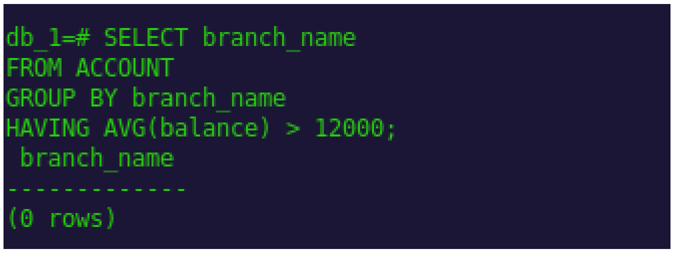
```
db_1=# SELECT customer_no
FROM ACCOUNT
WHERE branch_name = 'Main'
GROUP BY customer_no
HAVING COUNT(accno) >= 2;
 customer_no
-------------
 C001
(1 row)
```



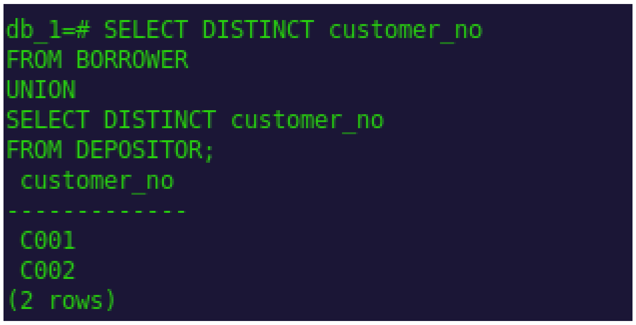
```
db_1=# SELECT AVG(balance) AS average_balance
FROM ACCOUNT
WHERE branch_name = 'Kollam';
    average_balance
-----------------------
 3000.0000000000000000
(1 row)
```



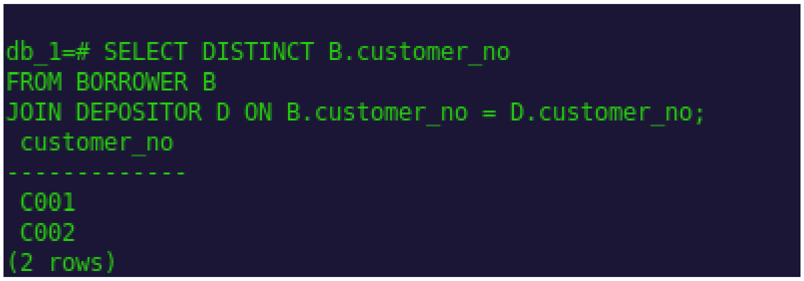
```
db_1=# SELECT A.branch_name, COUNT(DISTINCT D.customer_no) AS number_of_depositors
FROM ACCOUNT A
JOIN DEPOSITOR D ON A.accno = D.accno
GROUP BY A.branch_name;
 branch_name | number_of_depositors
-------------+----------------------
 Kollam      |                    1
 Main        |                    1
(2 rows)
```



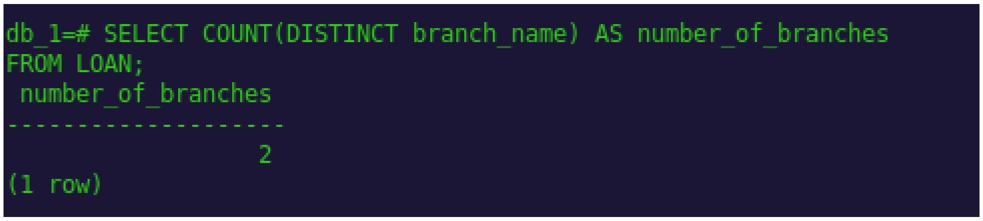
```
db_1=# SELECT branch_name
FROM ACCOUNT
GROUP BY branch_name
HAVING AVG(balance) > 12000;
 branch_name
-------------
(0 rows)
```



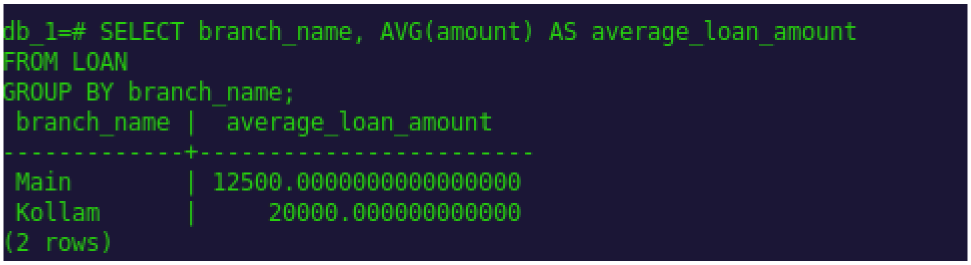
```
db_1=# SELECT DISTINCT customer_no
FROM BORROWER
UNION
SELECT DISTINCT customer_no
FROM DEPOSITOR;
 customer_no
-------------
 C001
 C002
(2 rows)
```



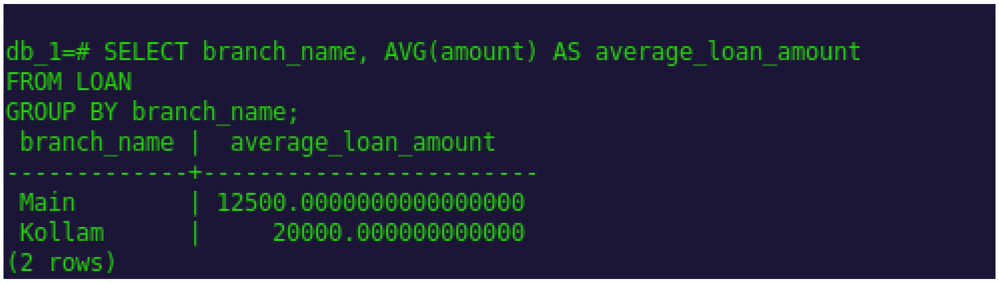
```
db_1=# SELECT DISTINCT B.customer_no
FROM BORROWER B
JOIN DEPOSITOR D ON B.customer_no = D.customer_no;
 customer_no
-------------
 C001
 C002
(2 rows)
```



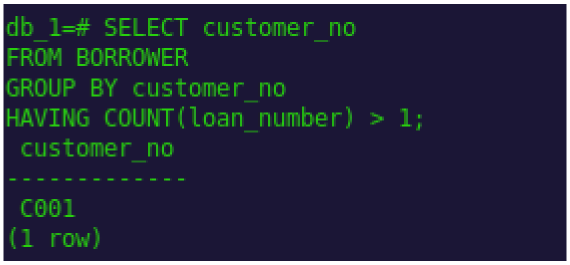
```
db_1=# SELECT COUNT(DISTINCT branch_name) AS number_of_branches
FROM LOAN;
 number_of_branches
--------------------
                  2
(1 row)
```



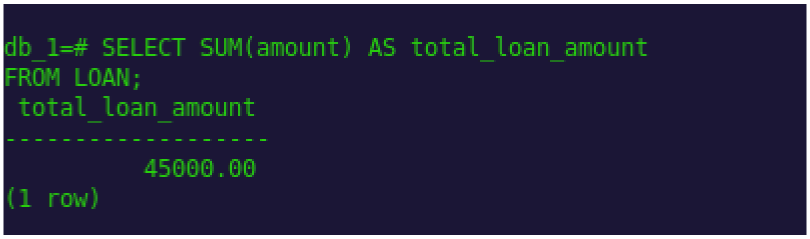
```
db_1=# SELECT branch_name, AVG(amount) AS average_loan_amount
FROM LOAN
GROUP BY branch_name;
 branch_name |  average_loan_amount
-------------+-----------------------
 Main        | 12500.0000000000000000
 Kollam      |     20000.000000000000
(2 rows)
```



```
db_1=# SELECT branch_name, AVG(amount) AS average_loan_amount
FROM LOAN
GROUP BY branch_name;
 branch_name |  average_loan_amount
-------------+-----------------------
 Main        | 12500.0000000000000000
 Kollam      |     20000.000000000000
(2 rows)
```



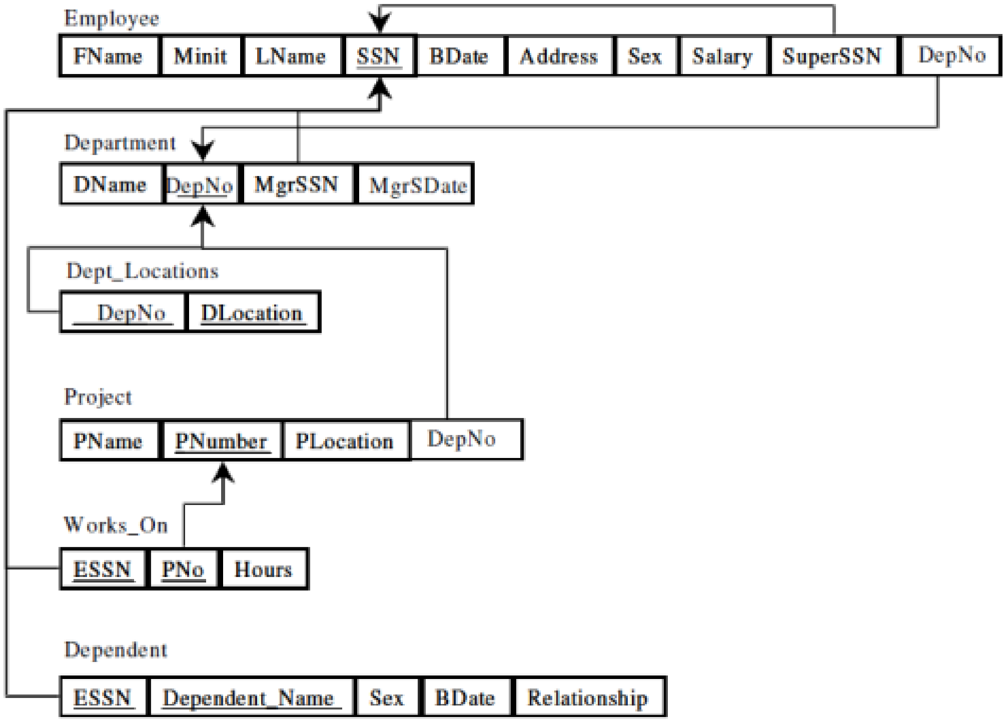
```
db_1=# SELECT customer_no
FROM BORROWER
GROUP BY customer_no
HAVING COUNT(loan_number) > 1;
 customer_no
-------------
 C001
(1 row)
```



```
db_1=# SELECT SUM(amount) AS total_loan_amount
FROM LOAN;
 total_loan_amount
-------------------
          45000.00
(1 row)
```

Question 2

Consider the logical schema given below. Create tables along with the constraints based on it, insert appropriate data and answer the following queries:



Employee

FName | Minit | LName | SSN | BDate | Address | Sex | Salary | SuperSSN | DepNo

Department

DName | DepNo | MgrSSN | MgrSDate

Dept_Locations

DepNo | DLocation

Project

PName | PNumber | PLocation | DepNo

Works_On

ESSN | PNo | Hours

Dependent

ESSN | Dependent_Name | Sex | BDate | Relationship

1. List the names of all employees who work in department 5.
2. List the names and salaries of all employees ordered by salary.
3. List the name of employees whose salary is between 30000 and 50000.
4. List the name and salary of employees who lives in Kollam.
5. List the name of employees who doesn't have a supervisor.
6. List department number and number of employees in departments that have more than 2 employees, ordered by department number.
7. List the ESSN of employees who works on project 3388 or project 1945.

8. List the location of department 1, 3, and 5.

9. List the name of all female employees.

```
CREATE TABLE Employee (
  FName VARCHAR(20),
  Minit CHAR(1),
  LName VARCHAR(20),
  SSN CHAR(9) PRIMARY KEY,
  BDate DATE,
  Address VARCHAR(50),
  Sex CHAR(1),
  Salary NUMERIC(10, 2),
  SuperSSN CHAR(9),
  DepNo INT,
  FOREIGN KEY (DepNo) REFERENCES Department(DepNo)
);
```

```
REATE TABLE Department (
  DName VARCHAR(20),
  DepNo INT PRIMARY KEY,
  MgrSSN CHAR(9),
  MgrSDate DATE,
  FOREIGN KEY (MgrSSN) REFERENCES Employee(SSN)
);
```

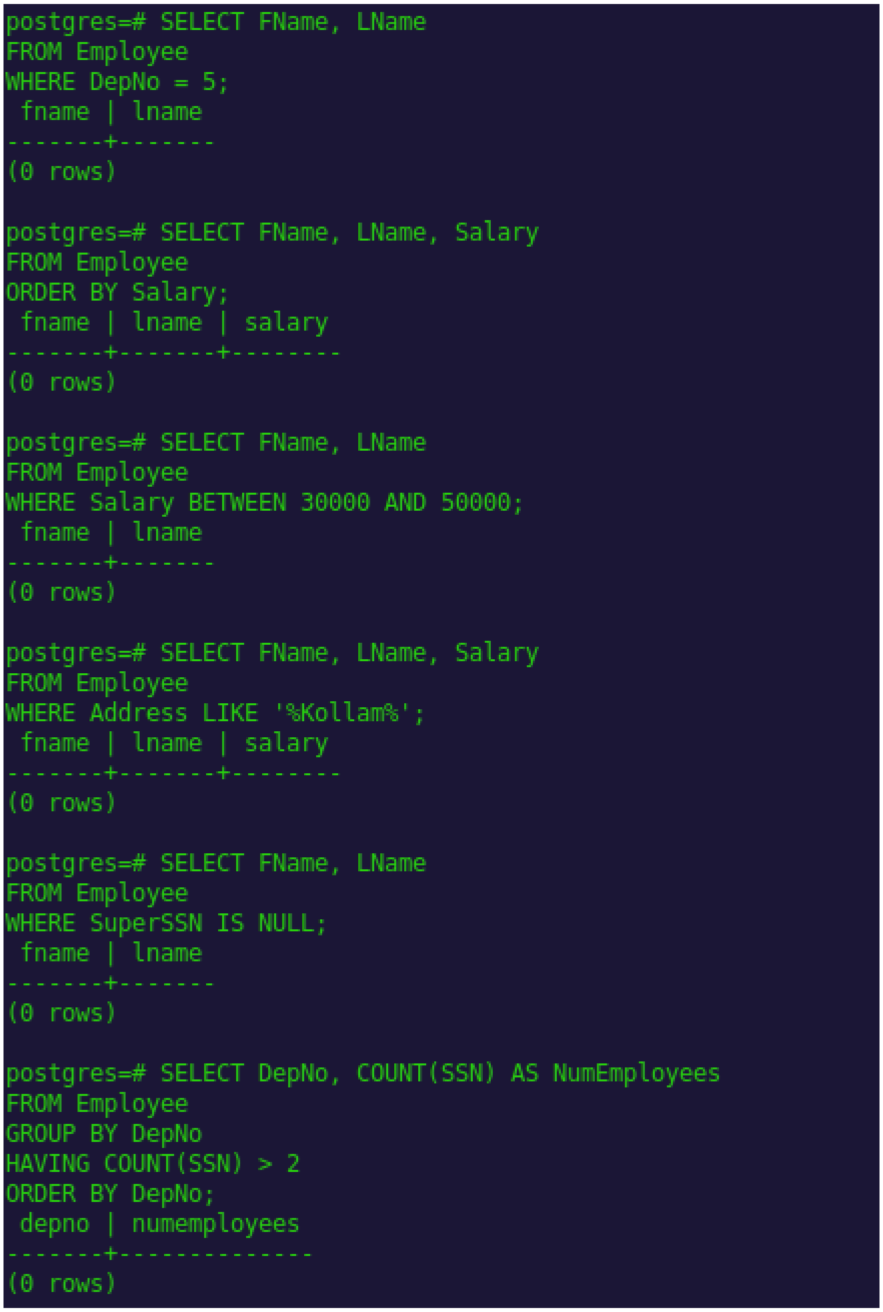
```
REATE TABLE Project (
  PName VARCHAR(20),
  PNumber INT PRIMARY KEY,
  PLocation VARCHAR(50),
  DepNo INT,
  FOREIGN KEY (DepNo) REFERENCES Department(DepNo)
);
```

```
REATE TABLE Works_On (
  ESSN CHAR(9),
  PNo INT,
  Hours NUMERIC(3, 1),
  PRIMARY KEY (ESSN, PNo),
  FOREIGN KEY (ESSN) REFERENCES Employee(SSN),
  FOREIGN KEY (PNo) REFERENCES Project(PNumber)
);
```

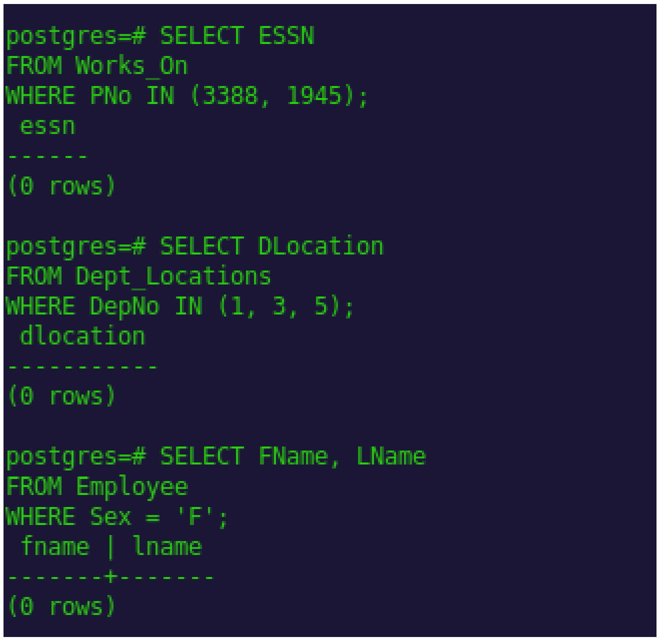
```
CREATE TABLE Dept_Locations (
  DepNo INT,
  DLocation VARCHAR(50),
  PRIMARY KEY (DepNo, DLocation),
  FOREIGN KEY (DepNo) REFERENCES Department(DepNo)
);
```

```
REATE TABLE Dependent (
  ESSN CHAR(9),
  Dependent_Name VARCHAR(20),
  Sex CHAR(1),
  BDate DATE,
  Relationship VARCHAR(10),
  FOREIGN KEY (ESSN) REFERENCES Employee(SSN)
);
```

Queries



```
postgres=# SELECT FName, LName
FROM Employee
WHERE DepNo = 5;
 fname | lname
-------+-------
(0 rows)

postgres=# SELECT FName, LName, Salary
FROM Employee
ORDER BY Salary;
 fname | lname | salary
-------+-------+--------
(0 rows)

postgres=# SELECT FName, LName
FROM Employee
WHERE Salary BETWEEN 30000 AND 50000;
 fname | lname
-------+-------
(0 rows)

postgres=# SELECT FName, LName, Salary
FROM Employee
WHERE Address LIKE '%Kollam%';
 fname | lname | salary
-------+-------+--------
(0 rows)

postgres=# SELECT FName, LName
FROM Employee
WHERE SuperSSN IS NULL;
 fname | lname
-------+-------
(0 rows)

postgres=# SELECT DepNo, COUNT(SSN) AS NumEmployees
FROM Employee
GROUP BY DepNo
HAVING COUNT(SSN) > 2
ORDER BY DepNo;
 depno | numemployees
-------+--------------
(0 rows)
```



```
postgres=# SELECT ESSN
FROM Works_On
WHERE PNo IN (3388, 1945);
 essn
------
(0 rows)

postgres=# SELECT DLocation
FROM Dept_Locations
WHERE DepNo IN (1, 3, 5);
 dlocation
-----------
(0 rows)

postgres=# SELECT FName, LName
FROM Employee
WHERE Sex = 'F';
 fname | lname
-------+-------
(0 rows)
```