Group Number	**64**
Section	**06**
Assignment	**10**

Name	**Student Number**
Saikot Paul	**500918922**
Fuoad Ibrahim	**500843602**
Akash Patel	**500896220**

Due to the current environment, many people refrain from visiting physical retail services at the concern of their own health. One of these services include visiting the local bank branch to handle their financial ordeals. The Retail Banking System allows customers to access their bank accounts from anywhere they'd like and be provided with the banking services that they are already familiar with. The

Customers are able to engage in the same transactions at a physical location, for example: checking account balances/transaction history, paying bills, transferring money between accounts, applying for loans, etc. Additionally, staff members of the banking system are able to access customers' accounts and its information should they require assistance, approve loan requests, etc.

Entities

Legend:

- Bold = primary key

Customer:

- **Customer_id**
- Customer_name
- Pin
- City
- Street
- apt#
- postal_code

Account

- **Account_no**
- balance

Savings_account

- **Account_no**
- Balance
- Interest

Chequings_account

- **Account_no**
- Balance

Transaction

- **Transaction_id**
- Transaction_description
- Amount
- Account_no

Loan

- **Loan_no**
- Loan_type
- Amount
- Customer_id
- employee_id

Branch

- **Branch_no**
- Address
- Bank_name
- Phone_no
- emp_id

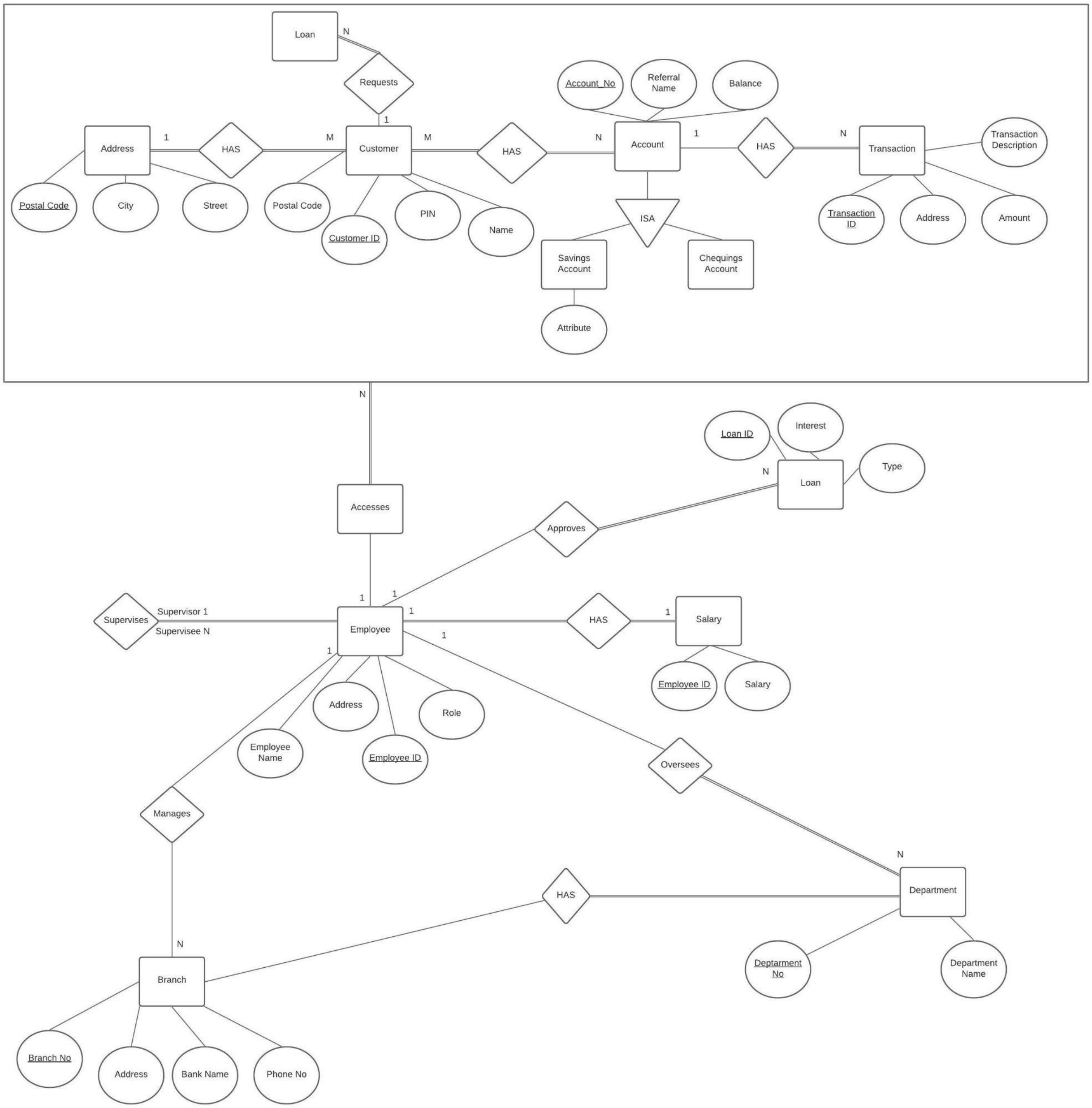
Department

- **Department_no**
- department_name

Employee

- **Emp_id**
- _Emp_name
- Address
- Emp_role

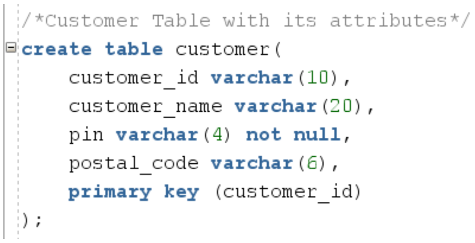
ER Diagram



SQL Code

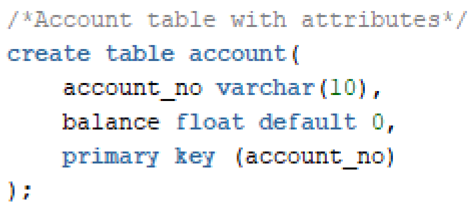
Create Tables:

Customer Table



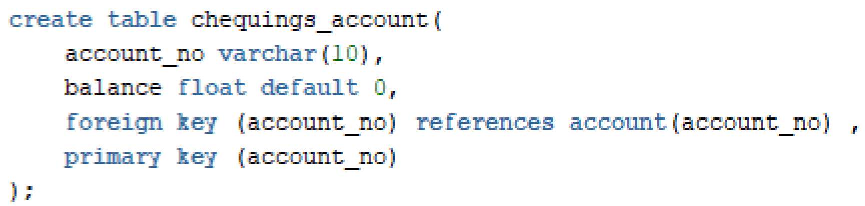
```
/*Customer Table with its attributes*/
create table customer(
    customer_id varchar(10),
    customer_name varchar(20),
    pin varchar(4) not null,
    postal_code varchar(6),
    primary key (customer_id)
);
```

Account Table



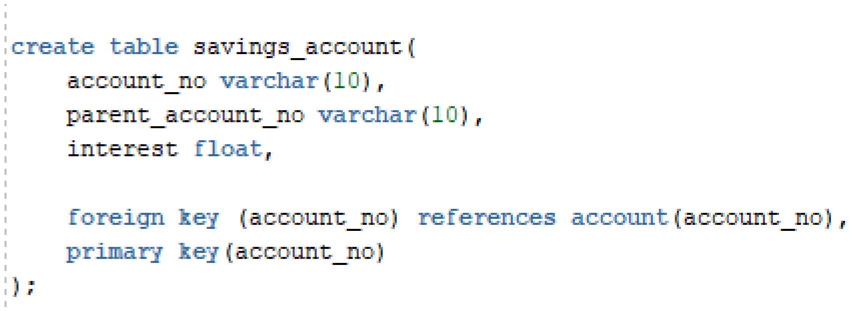
```sql
/*Account table with attributes*/
create table account(
    account_no varchar(10),
    balance float default 0,
    primary key (account_no)
);
```

Chequings Account Table



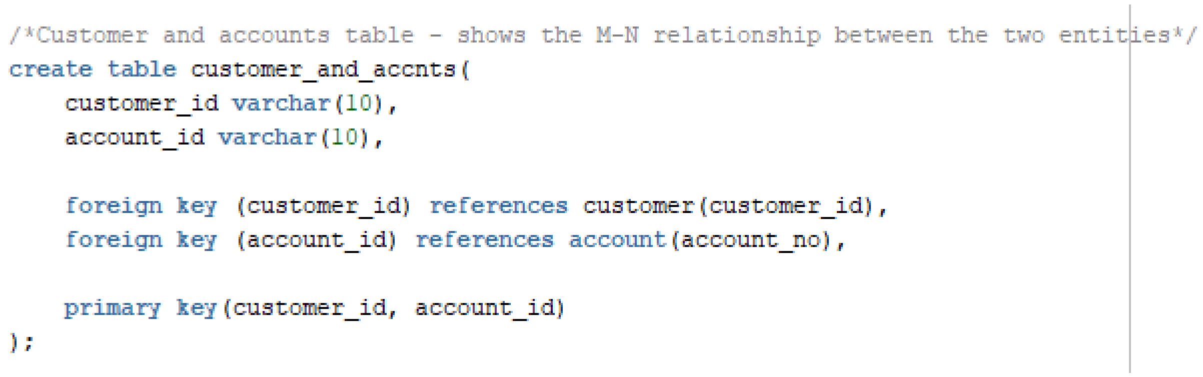
```sql
create table chequings_account(
    account_no varchar(10),
    balance float default 0,
    foreign key (account_no) references account(account_no) ,
    primary key (account_no)
);
```

Savings Account Table



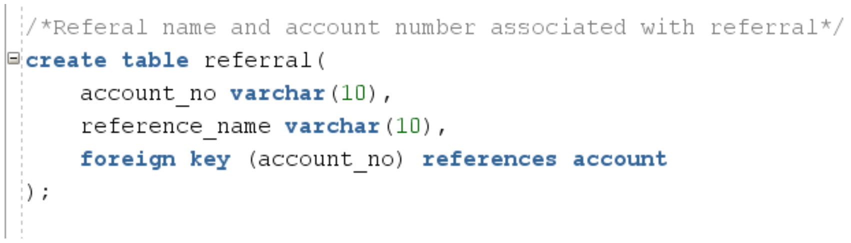
```sql
create table savings_account(
    account_no varchar(10),
    parent_account_no varchar(10),
    interest float,

    foreign key (account_no) references account(account_no),
    primary key(account_no)
);
```

Customer and Accounts Relationship



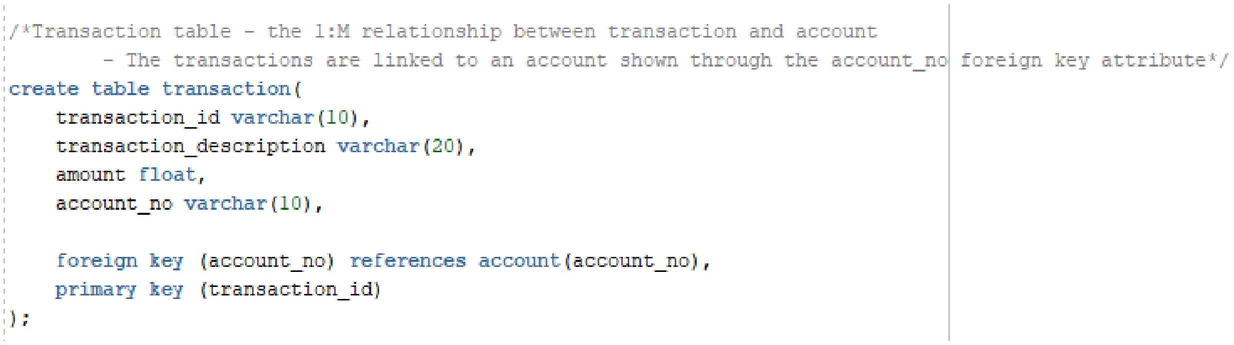
```sql
/*Customer and accounts table - shows the M-N relationship between the two entities*/
create table customer_and_accnts(
    customer_id varchar(10),
    account_id varchar(10),

    foreign key (customer_id) references customer(customer_id),
    foreign key (account_id) references account(account_no),

    primary key(customer_id, account_id)
);
```

Referral Table



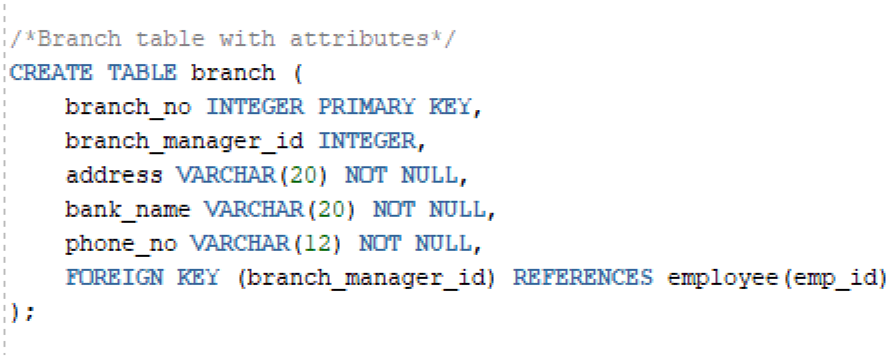
```sql
/*Referal name and account number associated with referral*/
create table referral(
    account_no varchar(10),
    reference_name varchar(10),
    foreign key (account_no) references account
);
```

Transaction Table



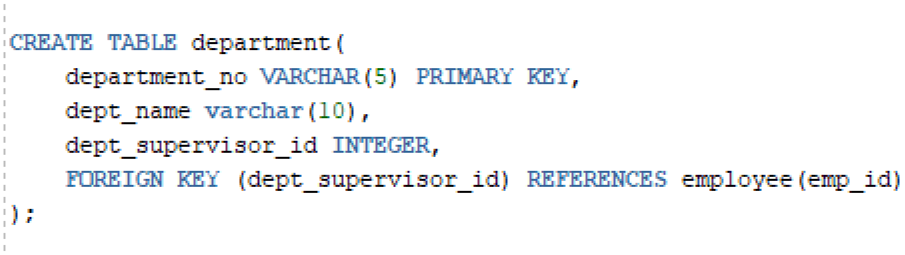
```
/*Transaction table - the 1:M relationship between transaction and account
        - The transactions are linked to an account shown through the account_no foreign key attribute*/
create table transaction(
    transaction_id varchar(10),
    transaction_description varchar(20),
    amount float,
    account_no varchar(10),

    foreign key (account_no) references account(account_no),
    primary key (transaction_id)
);
```

Branch Table



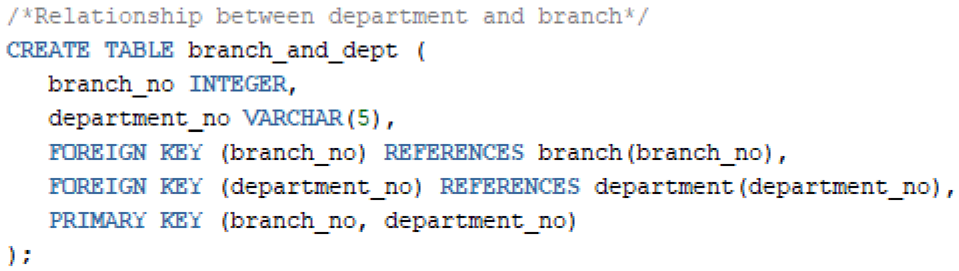
```
/*Branch table with attributes*/
CREATE TABLE branch (
    branch_no INTEGER PRIMARY KEY,
    branch_manager_id INTEGER,
    address VARCHAR(20) NOT NULL,
    bank_name VARCHAR(20) NOT NULL,
    phone_no VARCHAR(12) NOT NULL,
    FOREIGN KEY (branch_manager_id) REFERENCES employee(emp_id)
);
```

Department Table



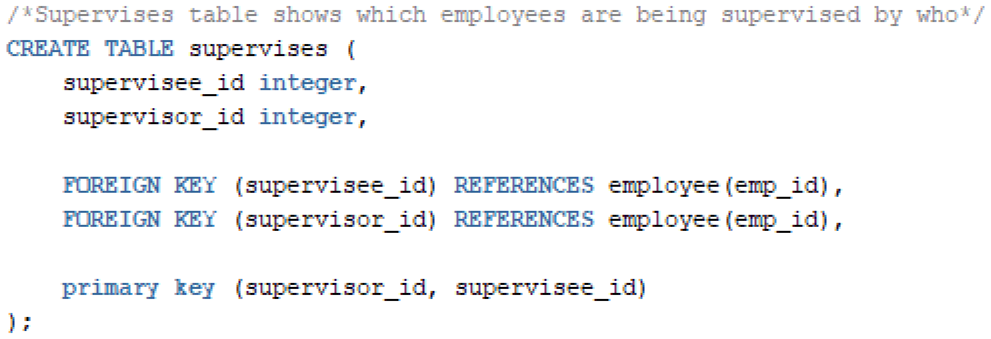
```
CREATE TABLE department(
    department_no VARCHAR(5) PRIMARY KEY,
    dept_name varchar(10),
    dept_supervisor_id INTEGER,
    FOREIGN KEY (dept_supervisor_id) REFERENCES employee(emp_id)
);
```

Branch and Department Relationship



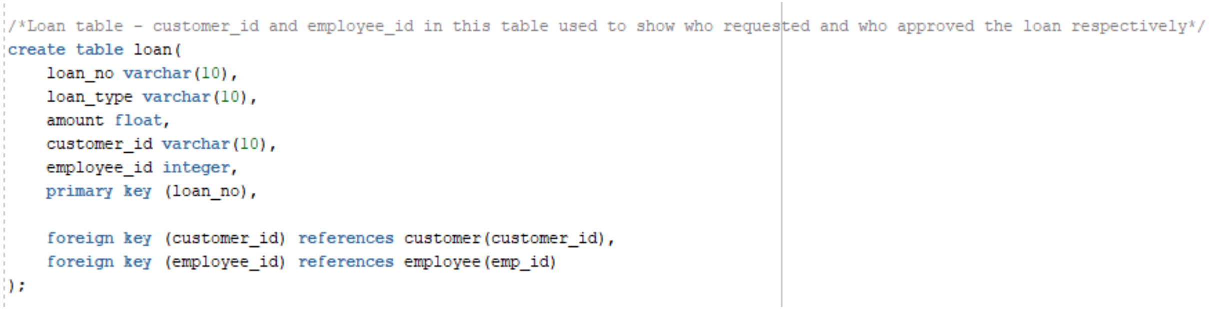
```
/*Relationship between department and branch*/
CREATE TABLE branch_and_dept (
   branch_no INTEGER,
   department_no VARCHAR(5),
   FOREIGN KEY (branch_no) REFERENCES branch(branch_no),
   FOREIGN KEY (department_no) REFERENCES department(department_no),
   PRIMARY KEY (branch_no, department_no)
);
```

Supervises Relationship



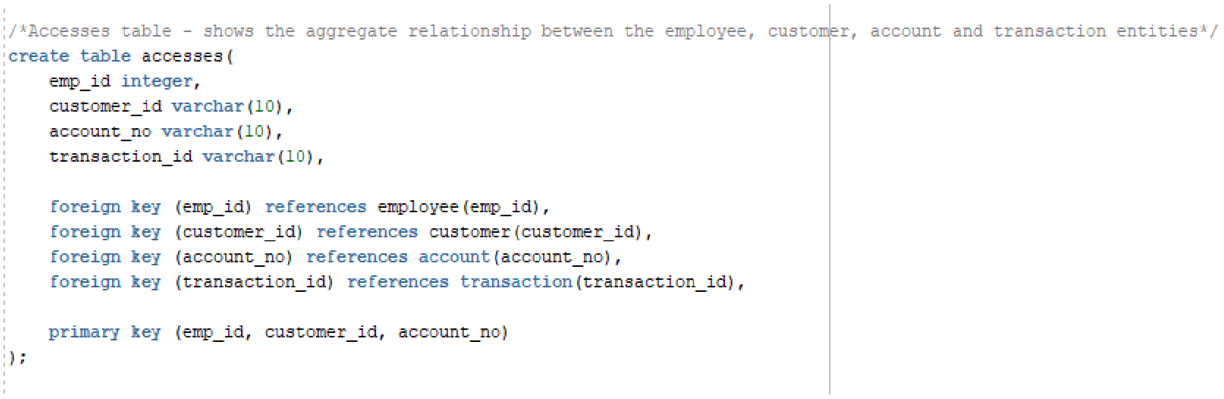
```
/*Supervises table shows which employees are being supervised by who*/
CREATE TABLE supervises (
    supervisee_id integer,
    supervisor_id integer,

    FOREIGN KEY (supervisee_id) REFERENCES employee(emp_id),
    FOREIGN KEY (supervisor_id) REFERENCES employee(emp_id),

    primary key (supervisor_id, supervisee_id)
);
```

Loan Table



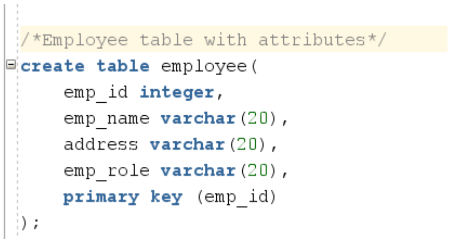
```
/*Loan table - customer_id and employee_id in this table used to show who requested and who approved the loan respectively*/
create table loan(
    loan_no varchar(10),
    loan_type varchar(10),
    amount float,
    customer_id varchar(10),
    employee_id integer,
    primary key (loan_no),

    foreign key (customer_id) references customer(customer_id),
    foreign key (employee_id) references employee(emp_id)
);
```

Accesses Relationship



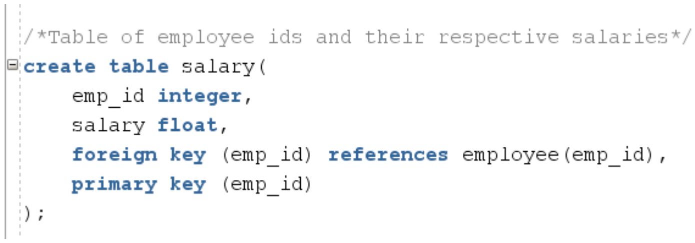
```
/*Accesses table - shows the aggregate relationship between the employee, customer, account and transaction entities*/
create table accesses(
    emp_id integer,
    customer_id varchar(10),
    account_no varchar(10),
    transaction_id varchar(10),

    foreign key (emp_id) references employee(emp_id),
    foreign key (customer_id) references customer(customer_id),
    foreign key (account_no) references account(account_no),
    foreign key (transaction_id) references transaction(transaction_id),

    primary key (emp_id, customer_id, account_no)
);
```

Employee Table



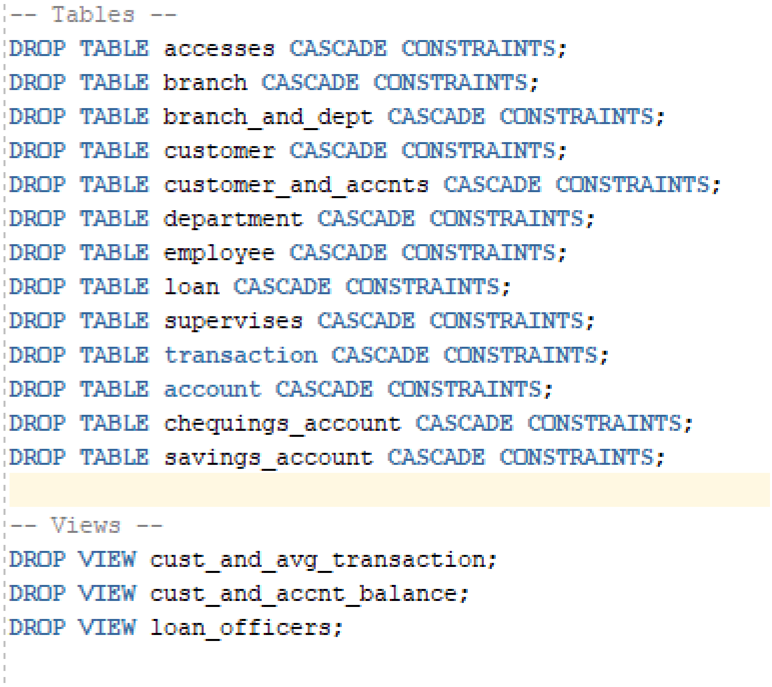
```
/*Employee table with attributes*/
create table employee(
    emp_id integer,
    emp_name varchar(20),
    address varchar(20),
    emp_role varchar(20),
    primary key (emp_id)
);
```

Salary Table



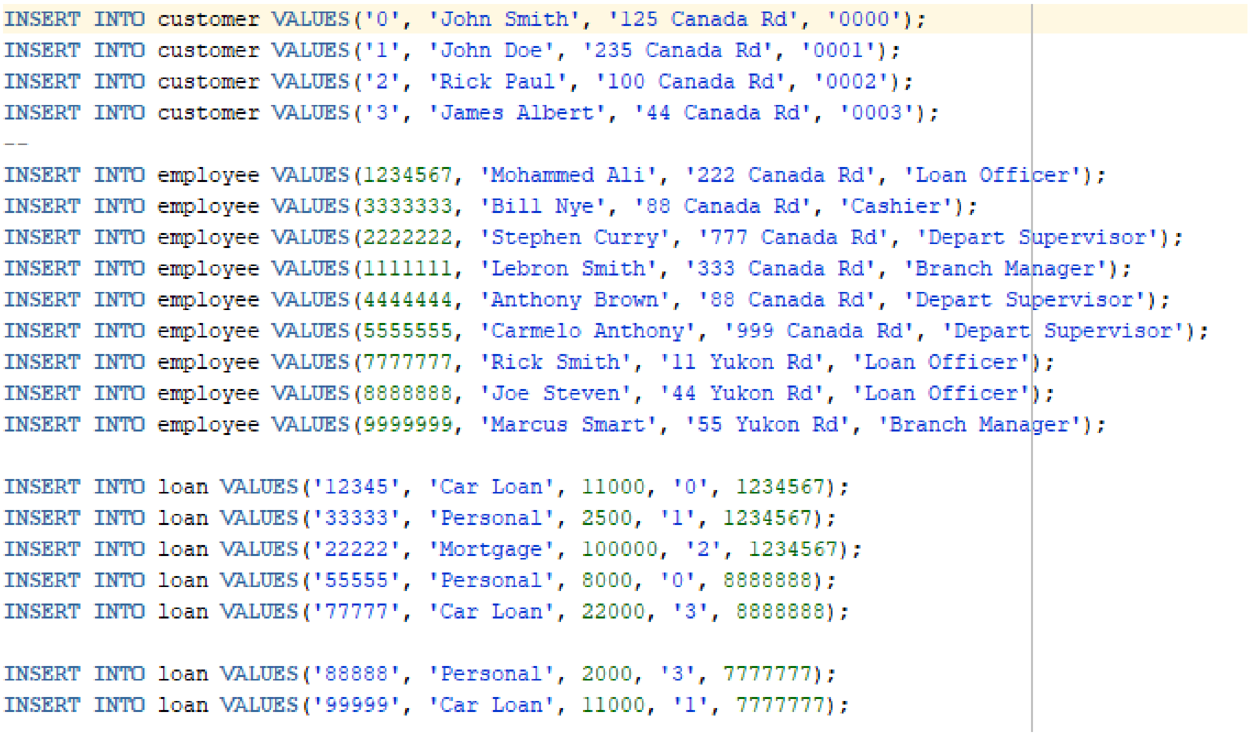
```
/*Table of employee ids and their respective salaries*/
create table salary(
    emp_id integer,
    salary float,
    foreign key (emp_id) references employee(emp_id),
    primary key (emp_id)
);
```

Drop Tables:

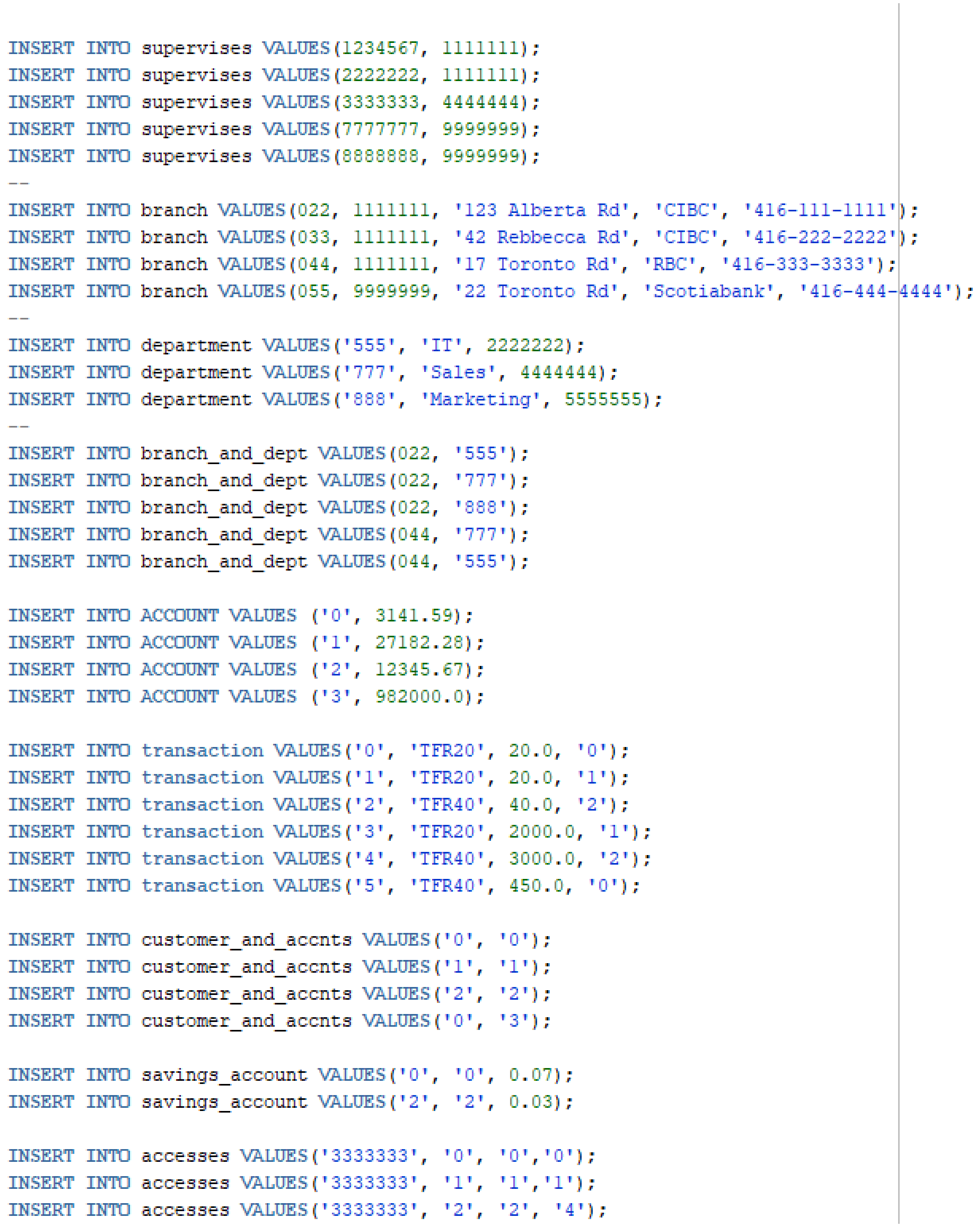


```
-- Tables --
DROP TABLE accesses CASCADE CONSTRAINTS;
DROP TABLE branch CASCADE CONSTRAINTS;
DROP TABLE branch_and_dept CASCADE CONSTRAINTS;
DROP TABLE customer CASCADE CONSTRAINTS;
DROP TABLE customer_and_accnts CASCADE CONSTRAINTS;
DROP TABLE department CASCADE CONSTRAINTS;
DROP TABLE employee CASCADE CONSTRAINTS;
DROP TABLE loan CASCADE CONSTRAINTS;
DROP TABLE supervises CASCADE CONSTRAINTS;
DROP TABLE transaction CASCADE CONSTRAINTS;
DROP TABLE account CASCADE CONSTRAINTS;
DROP TABLE chequings_account CASCADE CONSTRAINTS;
DROP TABLE savings_account CASCADE CONSTRAINTS;

-- Views --
DROP VIEW cust_and_avg_transaction;
DROP VIEW cust_and_accnt_balance;
DROP VIEW loan_officers;
```

Table Inserts:



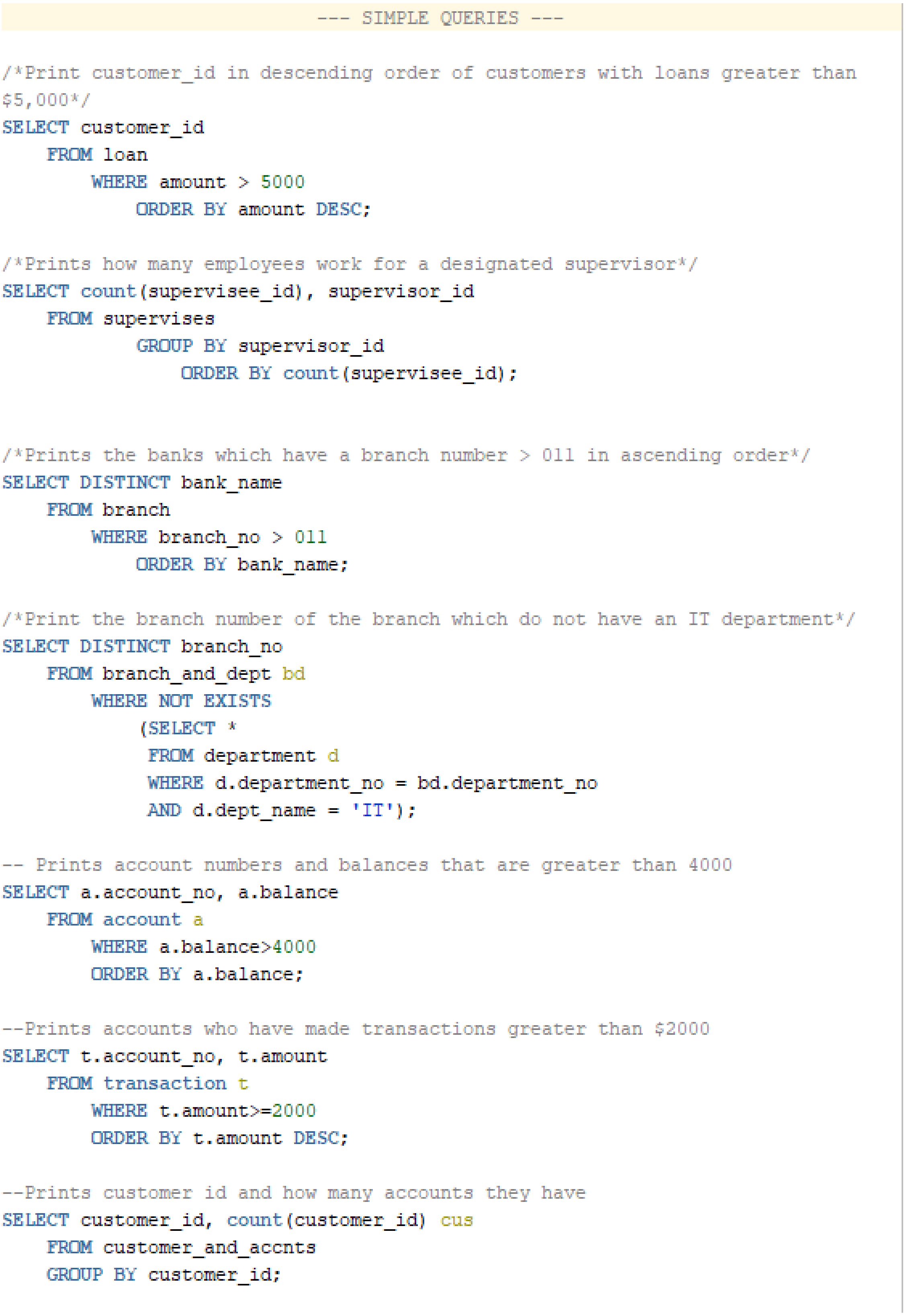
```sql
INSERT INTO customer VALUES('0', 'John Smith', '125 Canada Rd', '0000');
INSERT INTO customer VALUES('1', 'John Doe', '235 Canada Rd', '0001');
INSERT INTO customer VALUES('2', 'Rick Paul', '100 Canada Rd', '0002');
INSERT INTO customer VALUES('3', 'James Albert', '44 Canada Rd', '0003');
--
INSERT INTO employee VALUES(1234567, 'Mohammed Ali', '222 Canada Rd', 'Loan Officer');
INSERT INTO employee VALUES(3333333, 'Bill Nye', '88 Canada Rd', 'Cashier');
INSERT INTO employee VALUES(2222222, 'Stephen Curry', '777 Canada Rd', 'Depart Supervisor');
INSERT INTO employee VALUES(1111111, 'Lebron Smith', '333 Canada Rd', 'Branch Manager');
INSERT INTO employee VALUES(4444444, 'Anthony Brown', '88 Canada Rd', 'Depart Supervisor');
INSERT INTO employee VALUES(5555555, 'Carmelo Anthony', '999 Canada Rd', 'Depart Supervisor');
INSERT INTO employee VALUES(7777777, 'Rick Smith', '11 Yukon Rd', 'Loan Officer');
INSERT INTO employee VALUES(8888888, 'Joe Steven', '44 Yukon Rd', 'Loan Officer');
INSERT INTO employee VALUES(9999999, 'Marcus Smart', '55 Yukon Rd', 'Branch Manager');

INSERT INTO loan VALUES('12345', 'Car Loan', 11000, '0', 1234567);
INSERT INTO loan VALUES('33333', 'Personal', 2500, '1', 1234567);
INSERT INTO loan VALUES('22222', 'Mortgage', 100000, '2', 1234567);
INSERT INTO loan VALUES('55555', 'Personal', 8000, '0', 8888888);
INSERT INTO loan VALUES('77777', 'Car Loan', 22000, '3', 8888888);

INSERT INTO loan VALUES('88888', 'Personal', 2000, '3', 7777777);
INSERT INTO loan VALUES('99999', 'Car Loan', 11000, '1', 7777777);
```



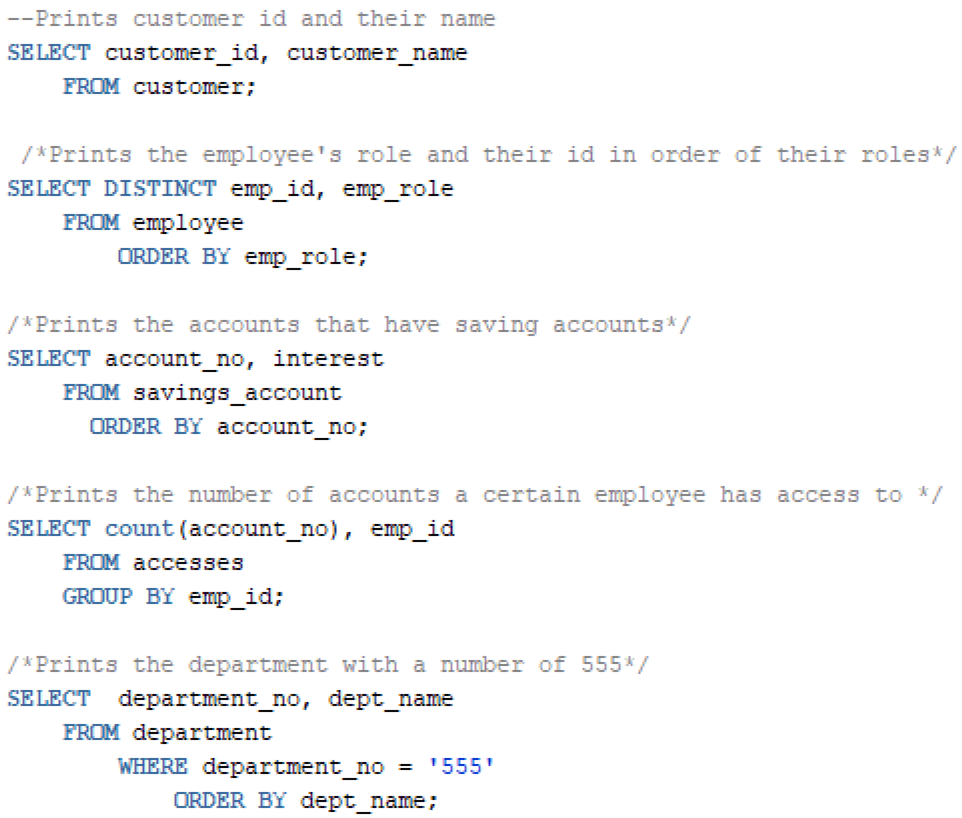
```
INSERT INTO supervises VALUES(1234567, 1111111);
INSERT INTO supervises VALUES(2222222, 1111111);
INSERT INTO supervises VALUES(3333333, 4444444);
INSERT INTO supervises VALUES(7777777, 9999999);
INSERT INTO supervises VALUES(8888888, 9999999);
--
INSERT INTO branch VALUES(022, 1111111, '123 Alberta Rd', 'CIBC', '416-111-1111');
INSERT INTO branch VALUES(033, 1111111, '42 Rebbecca Rd', 'CIBC', '416-222-2222');
INSERT INTO branch VALUES(044, 1111111, '17 Toronto Rd', 'RBC', '416-333-3333');
INSERT INTO branch VALUES(055, 9999999, '22 Toronto Rd', 'Scotiabank', '416-444-4444');
--
INSERT INTO department VALUES('555', 'IT', 2222222);
INSERT INTO department VALUES('777', 'Sales', 4444444);
INSERT INTO department VALUES('888', 'Marketing', 5555555);
--
INSERT INTO branch_and_dept VALUES(022, '555');
INSERT INTO branch_and_dept VALUES(022, '777');
INSERT INTO branch_and_dept VALUES(022, '888');
INSERT INTO branch_and_dept VALUES(044, '777');
INSERT INTO branch_and_dept VALUES(044, '555');

INSERT INTO ACCOUNT VALUES ('0', 3141.59);
INSERT INTO ACCOUNT VALUES ('1', 27182.28);
INSERT INTO ACCOUNT VALUES ('2', 12345.67);
INSERT INTO ACCOUNT VALUES ('3', 982000.0);

INSERT INTO transaction VALUES('0', 'TFR20', 20.0, '0');
INSERT INTO transaction VALUES('1', 'TFR20', 20.0, '1');
INSERT INTO transaction VALUES('2', 'TFR40', 40.0, '2');
INSERT INTO transaction VALUES('3', 'TFR20', 2000.0, '1');
INSERT INTO transaction VALUES('4', 'TFR40', 3000.0, '2');
INSERT INTO transaction VALUES('5', 'TFR40', 450.0, '0');

INSERT INTO customer_and_accnts VALUES('0', '0');
INSERT INTO customer_and_accnts VALUES('1', '1');
INSERT INTO customer_and_accnts VALUES('2', '2');
INSERT INTO customer_and_accnts VALUES('0', '3');

INSERT INTO savings_account VALUES('0', '0', 0.07);
INSERT INTO savings_account VALUES('2', '2', 0.03);

INSERT INTO accesses VALUES('3333333', '0', '0','0');
INSERT INTO accesses VALUES('3333333', '1', '1','1');
INSERT INTO accesses VALUES('3333333', '2', '2', '4');
```

Queries/Relational Algebra

Simple



```
                            --- SIMPLE QUERIES ---

/*Print customer_id in descending order of customers with loans greater than
$5,000*/
SELECT customer_id
    FROM loan
        WHERE amount > 5000
            ORDER BY amount DESC;

/*Prints how many employees work for a designated supervisor*/
SELECT count(supervisee_id), supervisor_id
    FROM supervises
            GROUP BY supervisor_id
                ORDER BY count(supervisee_id);


/*Prints the banks which have a branch number > 011 in ascending order*/
SELECT DISTINCT bank_name
    FROM branch
        WHERE branch_no > 011
            ORDER BY bank_name;

/*Print the branch number of the branch which do not have an IT department*/
SELECT DISTINCT branch_no
    FROM branch_and_dept bd
        WHERE NOT EXISTS
            (SELECT *
             FROM department d
             WHERE d.department_no = bd.department_no
             AND d.dept_name = 'IT');

-- Prints account numbers and balances that are greater than 4000
SELECT a.account_no, a.balance
    FROM account a
        WHERE a.balance>4000
        ORDER BY a.balance;

--Prints accounts who have made transactions greater than $2000
SELECT t.account_no, t.amount
    FROM transaction t
        WHERE t.amount>=2000
        ORDER BY t.amount DESC;

--Prints customer id and how many accounts they have
SELECT customer_id, count(customer_id) cus
    FROM customer_and_accnts
    GROUP BY customer_id;
```



```
--Prints customer id and their name
SELECT customer_id, customer_name
    FROM customer;

 /*Prints the employee's role and their id in order of their roles*/
SELECT DISTINCT emp_id, emp_role
    FROM employee
        ORDER BY emp_role;

/*Prints the accounts that have saving accounts*/
SELECT account_no, interest
    FROM savings_account
      ORDER BY account_no;

/*Prints the number of accounts a certain employee has access to */
SELECT count(account_no), emp_id
    FROM accesses
    GROUP BY emp_id;

/*Prints the department with a number of 555*/
SELECT  department_no, dept_name
    FROM department
        WHERE department_no = '555'
            ORDER BY dept_name;
```

a) $\pi_{CUSTOMER\ ID}(\sigma_{AMOUNT>5000}(loan))$
   - Selects customer ID from loan relation where amount is greater than 5000

b) ${}_{SUPERVISOR\ ID}F_{COUNT\ SUPERVISOR\ ID}(supervises)$
   - Groups by supervisor id and counts the number of supervisees under the supervisor

c) $\pi_{BANK\ NAME}(F_{DISTINCT\ BRANCH\ NAME}(\sigma_{BRANCH\ NO>011}(branch)))$
   - Selects bank name from branch relation where branch no>011

d) {t.branch_no | branch_and_dept(t) AND NOT ($\exists u$)(department(u)
   AND t.department_no = u.department_no
   AND u.dept_name = ‘IT’}
   - Selects all tuples from branch_and_dept relation where branch does not have an it department

e) $\pi_{ACCOUNT\ NO,\ AMOUNT}(\sigma_{AMOUNT\geq 2000}(transaction))$
   - Select account_no and amount where amount≥2000 in transaction relation

f) $\pi_{CUSTOMER\ ID,\ CUSTOMER\ NAME}(customer)$

- Project customer id and name from customer table

g) $\pi_{EMPLOYEE\ \ ID,\ EMPLOYEE\ NAME}(employee)$

- Project employee id and name from employee table

h) $\pi_{ACCOUNT\ NO,\ INTEREST,}(savings\ account)$

- Project account no and interest from savings account table

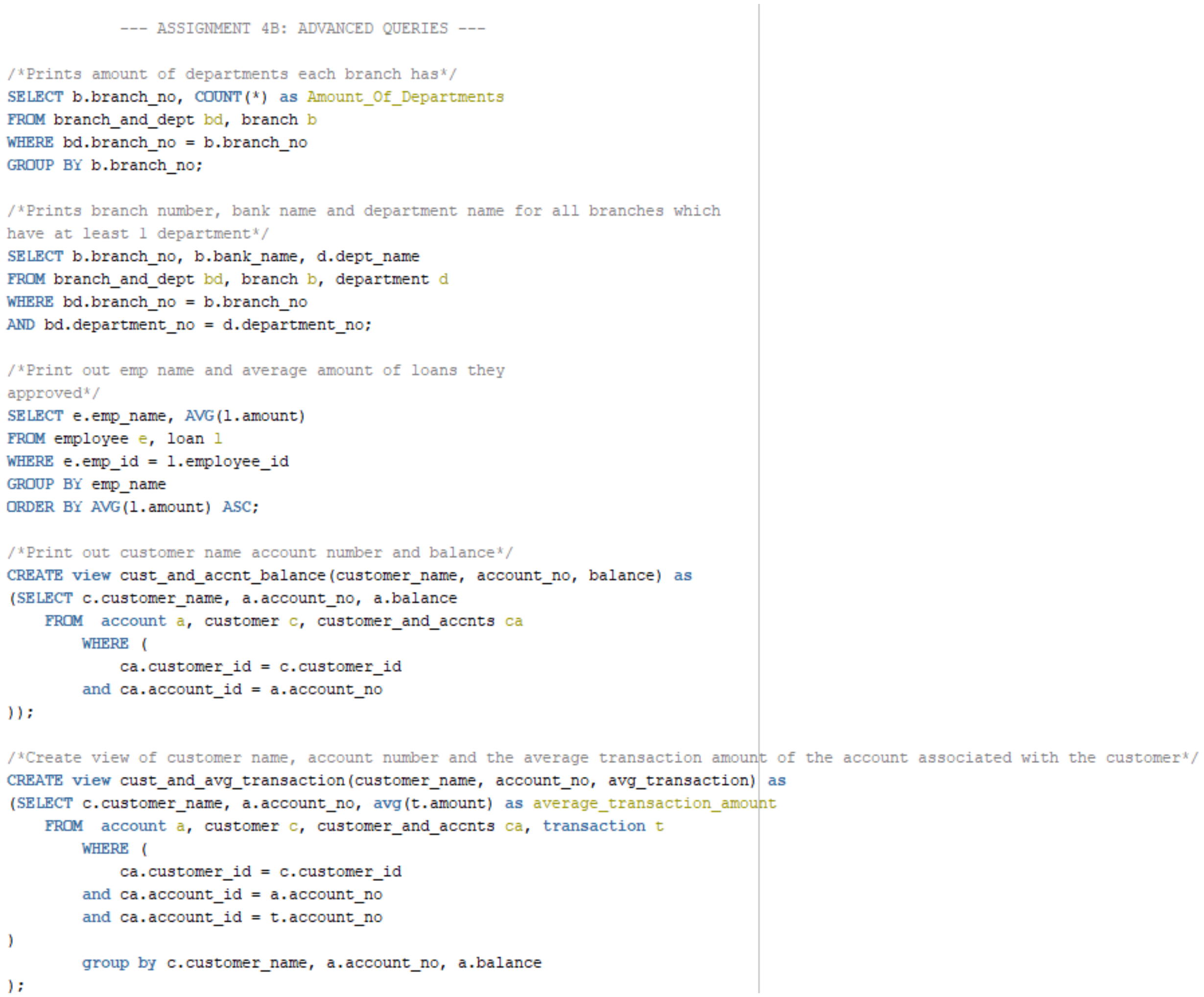
i) ${}_{EMP\ ID}F_{COUNT\ ACCOUNT\ NO}(accesses)$

- Group by employee id, count how many times an employee has had accessed various accounts

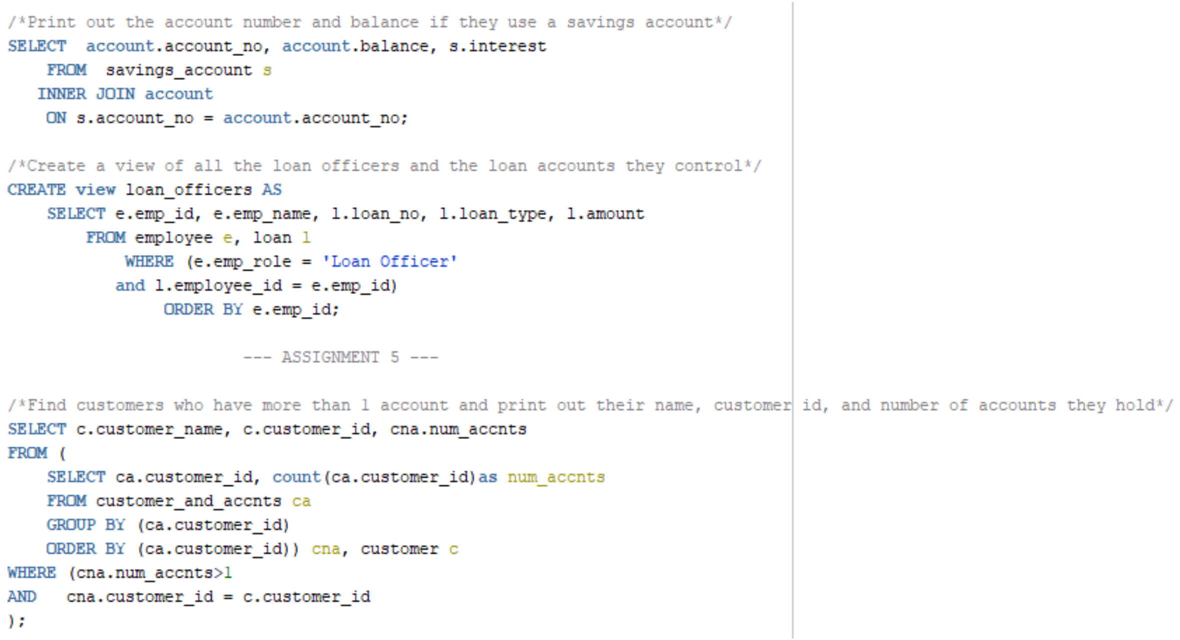
j) $\pi_{DEPT\ NO,\ \ DEPT\ NAME}(\sigma_{department\ no\ =\ '555'}(department))$

- Project department no and name where department no = 555

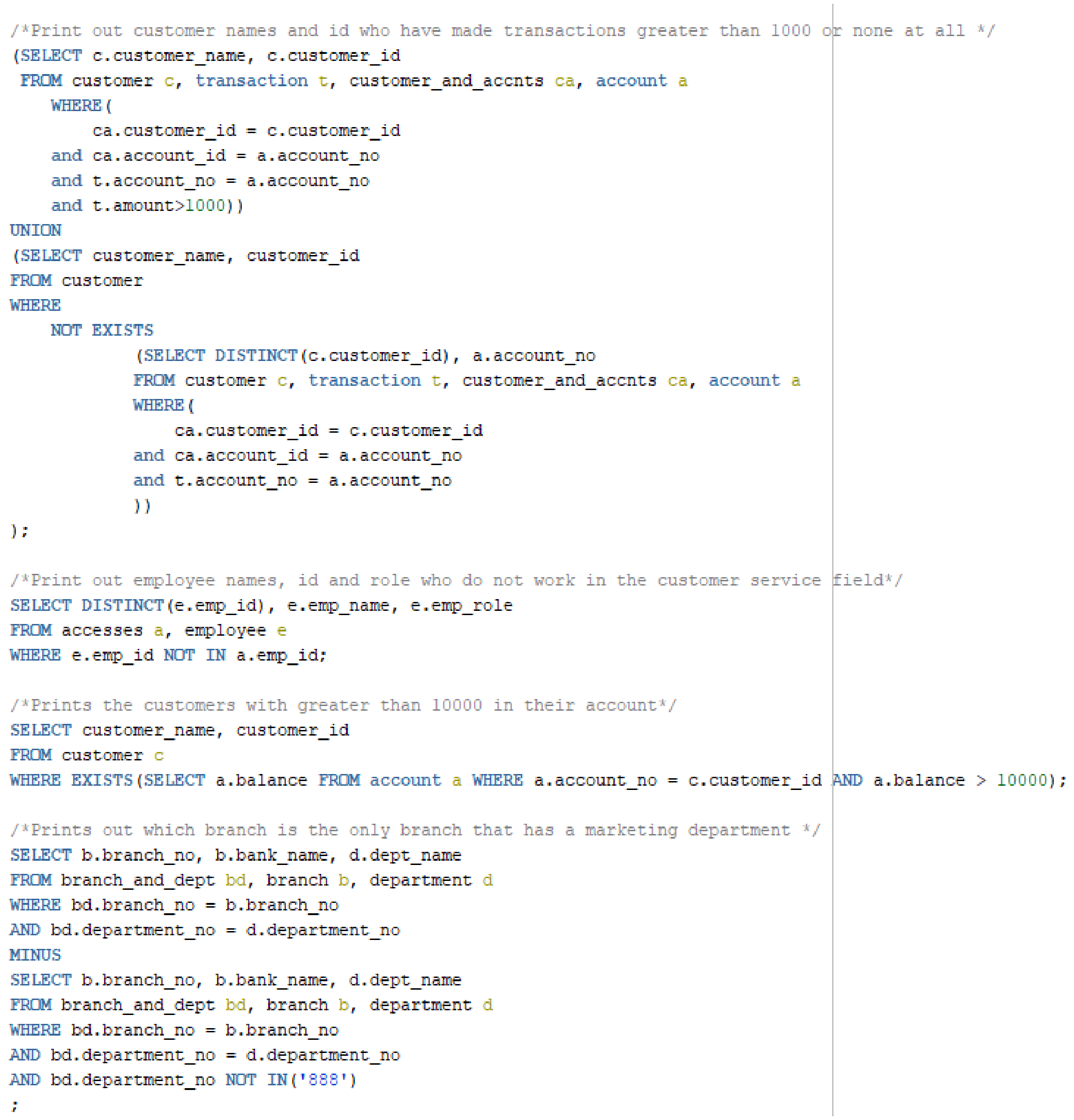
Advanced



```
            --- ASSIGNMENT 4B: ADVANCED QUERIES ---

/*Prints amount of departments each branch has*/
SELECT b.branch_no, COUNT(*) as Amount_Of_Departments
FROM branch_and_dept bd, branch b
WHERE bd.branch_no = b.branch_no
GROUP BY b.branch_no;

/*Prints branch number, bank name and department name for all branches which
have at least 1 department*/
SELECT b.branch_no, b.bank_name, d.dept_name
FROM branch_and_dept bd, branch b, department d
WHERE bd.branch_no = b.branch_no
AND bd.department_no = d.department_no;

/*Print out emp name and average amount of loans they
approved*/
SELECT e.emp_name, AVG(l.amount)
FROM employee e, loan l
WHERE e.emp_id = l.employee_id
GROUP BY emp_name
ORDER BY AVG(l.amount) ASC;

/*Print out customer name account number and balance*/
CREATE view cust_and_accnt_balance(customer_name, account_no, balance) as
(SELECT c.customer_name, a.account_no, a.balance
    FROM  account a, customer c, customer_and_accnts ca
        WHERE (
            ca.customer_id = c.customer_id
        and ca.account_id = a.account_no
));

/*Create view of customer name, account number and the average transaction amount of the account associated with the customer*/
CREATE view cust_and_avg_transaction(customer_name, account_no, avg_transaction) as
(SELECT c.customer_name, a.account_no, avg(t.amount) as average_transaction_amount
    FROM  account a, customer c, customer_and_accnts ca, transaction t
        WHERE (
            ca.customer_id = c.customer_id
        and ca.account_id = a.account_no
        and ca.account_id = t.account_no
)
        group by c.customer_name, a.account_no, a.balance
);
```



```
/*Print out the account number and balance if they use a savings account*/
SELECT  account.account_no, account.balance, s.interest
    FROM  savings_account s
   INNER JOIN account
    ON s.account_no = account.account_no;

/*Create a view of all the loan officers and the loan accounts they control*/
CREATE view loan_officers AS
    SELECT e.emp_id, e.emp_name, l.loan_no, l.loan_type, l.amount
        FROM employee e, loan l
            WHERE (e.emp_role = 'Loan Officer'
           and l.employee_id = e.emp_id)
                ORDER BY e.emp_id;

                        --- ASSIGNMENT 5 ---

/*Find customers who have more than 1 account and print out their name, customer id, and number of accounts they hold*/
SELECT c.customer_name, c.customer_id, cna.num_accnts
FROM (
    SELECT ca.customer_id, count(ca.customer_id)as num_accnts
    FROM customer_and_accnts ca
    GROUP BY (ca.customer_id)
    ORDER BY (ca.customer_id)) cna, customer c
WHERE (cna.num_accnts>1
AND   cna.customer_id = c.customer_id
);
```



```
/*Print out customer names and id who have made transactions greater than 1000 or none at all */
(SELECT c.customer_name, c.customer_id
 FROM customer c, transaction t, customer_and_accnts ca, account a
    WHERE(
        ca.customer_id = c.customer_id
    and ca.account_id = a.account_no
    and t.account_no = a.account_no
    and t.amount>1000))
UNION
(SELECT customer_name, customer_id
FROM customer
WHERE
    NOT EXISTS
            (SELECT DISTINCT(c.customer_id), a.account_no
            FROM customer c, transaction t, customer_and_accnts ca, account a
            WHERE(
                ca.customer_id = c.customer_id
            and ca.account_id = a.account_no
            and t.account_no = a.account_no
            ))
);

/*Print out employee names, id and role who do not work in the customer service field*/
SELECT DISTINCT(e.emp_id), e.emp_name, e.emp_role
FROM accesses a, employee e
WHERE e.emp_id NOT IN a.emp_id;

/*Prints the customers with greater than 10000 in their account*/
SELECT customer_name, customer_id
FROM customer c
WHERE EXISTS(SELECT a.balance FROM account a WHERE a.account_no = c.customer_id AND a.balance > 10000);

/*Prints out which branch is the only branch that has a marketing department */
SELECT b.branch_no, b.bank_name, d.dept_name
FROM branch_and_dept bd, branch b, department d
WHERE bd.branch_no = b.branch_no
AND bd.department_no = d.department_no
MINUS
SELECT b.branch_no, b.bank_name, d.dept_name
FROM branch_and_dept bd, branch b, department d
WHERE bd.branch_no = b.branch_no
AND bd.department_no = d.department_no
AND bd.department_no NOT IN('888')
;
```

a) $\pi_{BRANCH\ NO,\ COUNT\ DEPT\ NO}(\ {}_{BRANCH\ NO}F_{COUNT\ DEPT\ NO}\ (\sigma_{BRANCH\ AND\ DEPT.BRANCH\ NO\ =\ BRANCH.BRANCH\ NO\ (AND)\ DEPARTMENT.DEPT\ NO\ =\ BRANCH\ AND\ DEPT.DEPT\ NO}$ (branch_and_department x branch x department)))

- Project branch no and department no from result relation
- Result relation is the computing the number of departments grouped by branch no on the cartesian product of branch_and_dept, branch and department

b) {t.branch_no, t.bank_name, u.department_name |branch(t) AND department(u)
AND $(\exists v)$(branch_and_dept(v)
AND   t.branch_no = v.branch_no
AND   v.department_no = u.department_no}

- Select all tuples with at least one branch

c) $\pi_{EMPNAME,\ AVG\ AMOUNT}(\ {}_{EMPNAME}F_{AVG\ AMOUNT}(\sigma_{EMPLOYEE.ID=LOAN.ID}(employee\ \times\ loan))$

- Project emp name and amount on result relation
- Result relation calculates the average loan amount grouped by emp name on cartesian product of employee and loan

d) $F_{AVG\ AMOUNT}(\sigma_{CA.CUSTOMER\ ID\ =\ C.CUSTOMER\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ A.ACCOUNT\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ T.ACCOUNT\ NO})$
(customer x account x transaction)

- Calculate average transaction amount per account

e) ${}_{CUSTOMER\ NAME,\ ACCOUNT\ NO,\ BALANCE}F_{AVG\ AMOUNT}(\sigma_{CA.CUSTOMER\ ID\ =\ C.CUSTOMER\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ A.ACCOUNT\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ T.ACCOUNT\ NO})$

- Group by customer name, account, balance and compute average transaction

f) $\pi_{ACCOUNT\ NO,\ BALANCE,\ INTEREST}(\sigma_{S.ACCOUNT\ =\ A.ACCOUNT\ NO}(savings\ account\ \ \times\ account)$

- Inner product computes the intersection of accounts and savings account where the account numbers are the same
- Project the account no, balance and interest

g) {t.emp_id, t.emp_name, u.loan_no, u.loan_type, u.amount | employee(t) AND loan(u)
AND t.emp_role = ‘Loan Officer’
AND u.emp_id = t.emp_id
}

- Select all tuples where emp id is loan officer, and emp id in loan is the same in employee relation

h) t1 = $\pi_{CUSTOMER\ ID,\ COUNT\ CUSTOMER\ ID}(\ \ {}_{CUSTOMER\ ID}F_{COUNT\ CUSTOMER\ ID}(customer\ and\ accnts))$
result =
$\pi_{CUSTOMER\ NAME.\ CUSTOMER\ ID,\ COUNT\ CUSTOMER\ ID}(\sigma_{C.CUSTOMER\ ID\ =\ T1.CUSTOMER\ ID\ AND\ COUNT\ CUSTOMER\ ID\ >1}(customer\ \ \times$

- t1 is a table composed of the customer id and the count of id
- Result a relation that projects customer name, id and count of the id from the cartesian product of t1 and customer relations

i) t1 = $\sigma_{CA.CUSTOMER\ ID\ =\ C.CUSTOMER\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ A.ACCOUNT\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ T.ACCOUNT\ NO\ AND\ T.AMOUNT>1000}$
t2 = $\sigma_{CA.CUSTOMER\ ID\ =\ C.CUSTOMER\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ A.ACCOUNT\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ T.ACCOUNT\ NO\ AND\ T.AMOUNT>0}$
t3 = $\pi_{CUSTOMER\ NAME,\ CUSTOMER\ ID}(\sigma_{CA.CUSTOMER\ ID\ =\ C.CUSTOMER\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ A.ACCOUNT\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ T.ACCOUNT\ NO})$
result = t1 ∪ (t3-t2)

- t1 is a table with customers who have made transactions greater than 1000
- t2 is a table where customers have made a transaction
- t3 - t2 is a table with customers who have not made any transactions
- the union of t1 and t3 -t2 is a query that provides a table with customers who have made transactions greater than 1000 or none at all

j) {u.emp_id, u.emp_name, u.emp_role | employee(u)
AND $(\exists u)$(accesses(t)
AND u.emp_id ≠ t.emp_id)}

- Select tuples in accesses and employee where employee tuple u.emp_id is not in accesses relation

h) $\pi_{CUSTOMER\ NAME,\ CUSTOMER\ ID}(\sigma_{CA.CUSTOMER\ ID\ =\ C.CUSTOMER\ ID\ (AND)\ CA.ACCOUNT\ ID\ =\ A.ACCOUNT\ ID\ (AND)\ A.BALANCE>1000})$

- Project customer name and id on result relation

- Result relation is a table with customer who have more than 10000 in their account

i) f1 = {t.branch_no, t.bank_name, u.dept_name | branch_and_dept(t) AND department(u) AND ($\exists v$)(branch(v) AND t.branch_no = v.branch_no
AND v.department_no = u.department_no)
f2 = {t.branch_no, t.bank_name, t.dept_name | f1(t) AND t.department_no≠'888'}

Result = f2 - f1

<u>Functional Dependencies</u>

Legend:

- Bold + underlined = primary key
- Underlined = foreign key

- customer(<u>**customer_id**</u>, customer_name, pin, city, street, apt#, postal_code)
  - postal_code -> city, street
  - customer_id → {customer_name, address, pin}
  - Reasoning:
    - customer name, address and pin all are dependent on customer id

- account(<u>**account_no**</u>, balance)
  - account_no → {balance}
  - Reasoning:
    - Balance is determined by the account number

- savings_account(<u>**account_no**</u>, balance, interest)
  - account_no → {balance, interest}
  - Reasoning
    - Balance and interest are functionally dependent on primary key account number

- transaction(<u>**transaction_id**</u>, transaction_description, amount, <u>account_no</u>)
  - Relationship:
    - account (one) has transaction (many)
  - transaction_id → {transaction_description, amount, account_no}
  - Reasoning:
    - All the attributes on right hand side are determined by the transaction id

- loan(<u>**loan_no**</u>, loan_type, amount, <u>customer_id, employee_id</u>)
  - Relationship:
    - customer (one) requests loan (many)
    - employee (one) approves loan (many)
  - loan_no → {loan_type, amount, customer_id, employee_id}

  - Reasoning
    - Loan type, and amount are determined by the loan number
    - The many to one relationship between customer and employee shows that both customer id and employee id are determined by the loan number

- customer_and_accnts(**customer_id**, **account_id**, reference_name)
  - account_id -> reference_name
  - customer_id , account_id -> reference_name
  - This table shows many to many relationship between customer and accounts therefore there are no functional dependencies

- accesses(customer_id, account_no, transaction_id, emp_id)
  - Relationship
    - Aggregate function, many (customer_id, account_no, transaction_id) to one (emp_id) relationship
    - Employee (one) accesses assignment (many)
  - {customer_id, account_no, transaction_id} → emp_id
  - Reasoning:
    - Due to the nature of the relationship the employee id is functionally determined by the other remaining attributes

- Branch(**branch_no,** address, bank_name, phone_no, emp_id)
  - Relationship
    - Branch (many) managed by employee (one)
  - branch_no → {address, bank_name, phone_no, emp_id}
  - Reasoning:
    - Branch number is the primary key and the value of each non candidate key is determined by the branch number

- branch_and_dept
  - Relationship:
    - Branch (many) has department (many)
  - Many to many relationship, no functional dependencies.

- Employee(**emp_id,** emp_name, address, emp_role)
  - emp_id → {emp_name, address, emp_role}
  - Reasoning:
    - Employee id is the primary key and each non candidate key is functionally determined by the employee id

- supervises(supervisor_id, supervisee_id)
  - Relationship
    - supervisor (one) supervises supervisee (many)

    - supervisee_id → supervisor_id
    - Reasoning:
        - Many to one relationship, therefore the supervisor is functionally determined by the supervisee id

3NF Normalization

Legend:
- Bold + underlined = primary key
- Underlined = foreign key
-

*Extra columns were assumed to complete 3NF Normalization

customer:

2NF

R1.1 customer(**customer_id**, customer_name, pin, city, street, apt#, postal_code)

FD: customer_id → customer_name
customer_id → pin
customer_id → city
customer_id → street
customer_id → apt#
customer_id → postal_code
postal_code → city
postal_code → street

Customer_id$^+$ = {customer_name, pin, city, street, apt#, postal_code}
Customer_id$_1^+$ = {pin, city, street, apt#, postal_code}
customer

3NF

R1.1 (**customer_id**, customer_name, pin, apt #, postal_code)
R1.2 (**postal_code,** city, street)

FD: customer_id → {customer_name, pin, apt#, postal_code}
postal_code → {city, street}

- Postal code is unique only locally/nationally

customer_and_accnts:

2NF

R = customer_and_accnts(**customer_id, account_no**, reference_name)

FD: customer_id, account_no → {reference_name}
account_no → reference_name

3NF
R2.1 (**customer_id,** account_no)
R2.2 (**account_no**, reference_name)

Supervises:
1NF
R = supervises(supervisor_id, supervisee_id, department_no)
FD: supervisee_id → supervisor_id
supervisee_id, supervisor_id → department_no

2NF
R1.1 (supervisor_id, supervisee_id)
R1.2 (supervisor_id, supervisee_id, department_no)

***Compound FD** because reference name depends on two different attributes, both customer_id and account_no

3NF Algorithm

1

● cs_id → cs_name
● cs_id → pin
● cs_id → city
● cs_id → st
● cs_id → ps-code
ps_code → city
ps_code → st

Redundant (cs_id → city, cs_id → st)
R1 (cs_id → pin … cs_id → ps-code)
R2 (ps_code → city, ps_code → st)

R1 (cs_id, cs_name, pin, ps_code)
R2 (ps_code, city, street)

2a

$cs\_id^{+} = \{ cs\_name, pin, city, st, ps\text{-}code \}$
$cs\_id^{+}_{1} = \{ pin, city, st, ps\text{-}code \}$

3

NLHS ∧ NRH } KEY
LHS ∧ NRHS } KEY
NLHS ∧ RHS } ≠ KEY

cs_id ≐ KEY

Transitive dependency in both examples.

cust_and_accnts (cs_id, accnt_no, ref_name)

1/2 cs_id, accnt_no → ref_name } redundant
accnt_no → ref_name

$\{ cs\_id, accnt\_no \}^+ = \{ ref\_name \}$

$\{ accnt\_no \}^+ = \{ refname \}$

3 cs_id, accnt_no ⇒ KEYS

4 R1 (cs_id, accnt_no)
R2 (accnt_no, refname)

- ps_code and cs_id both functionally determine city and street
- cs_id and accnt_no both functionally determine ref_name

BCNF Normalization

Employee(emp_id, emp_name, address, emp_role, salary)

- {emp_id, address} → {emp_role}
- emp_id → salary
- **Not in BCNF** because emp_name → salary holds and emp_role is not a candidate key
- **Partial dependency** because non key attribute salary is functionally determined by part of composite key
    - composite key = {emp_name, address}

R = (**emp_id**, emp_nameaddress, emp_role, salary)

R1 = (**emp_id**, salary)

- emp_id→ salary
    - emp_name is the key for R1
    - R1 is in BCNF

R2 = (**emp_id**, **address**, emp_name, emp_role)

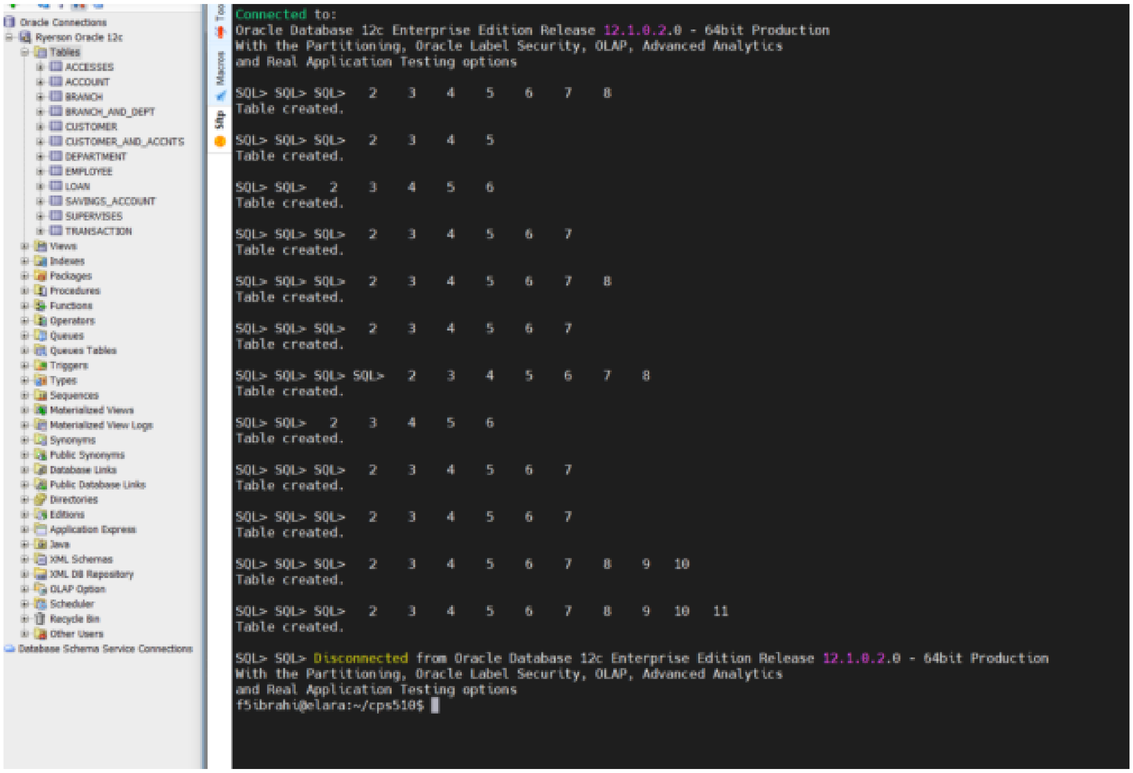
- emp_name, address → emp_role
    - {emp_name, address} is the key for R2
    - R2 is in BCNF

R1 & R2 are both in BCNF

The following relations are in BCNF because the determinant for their functional dependencies are candidate keys.

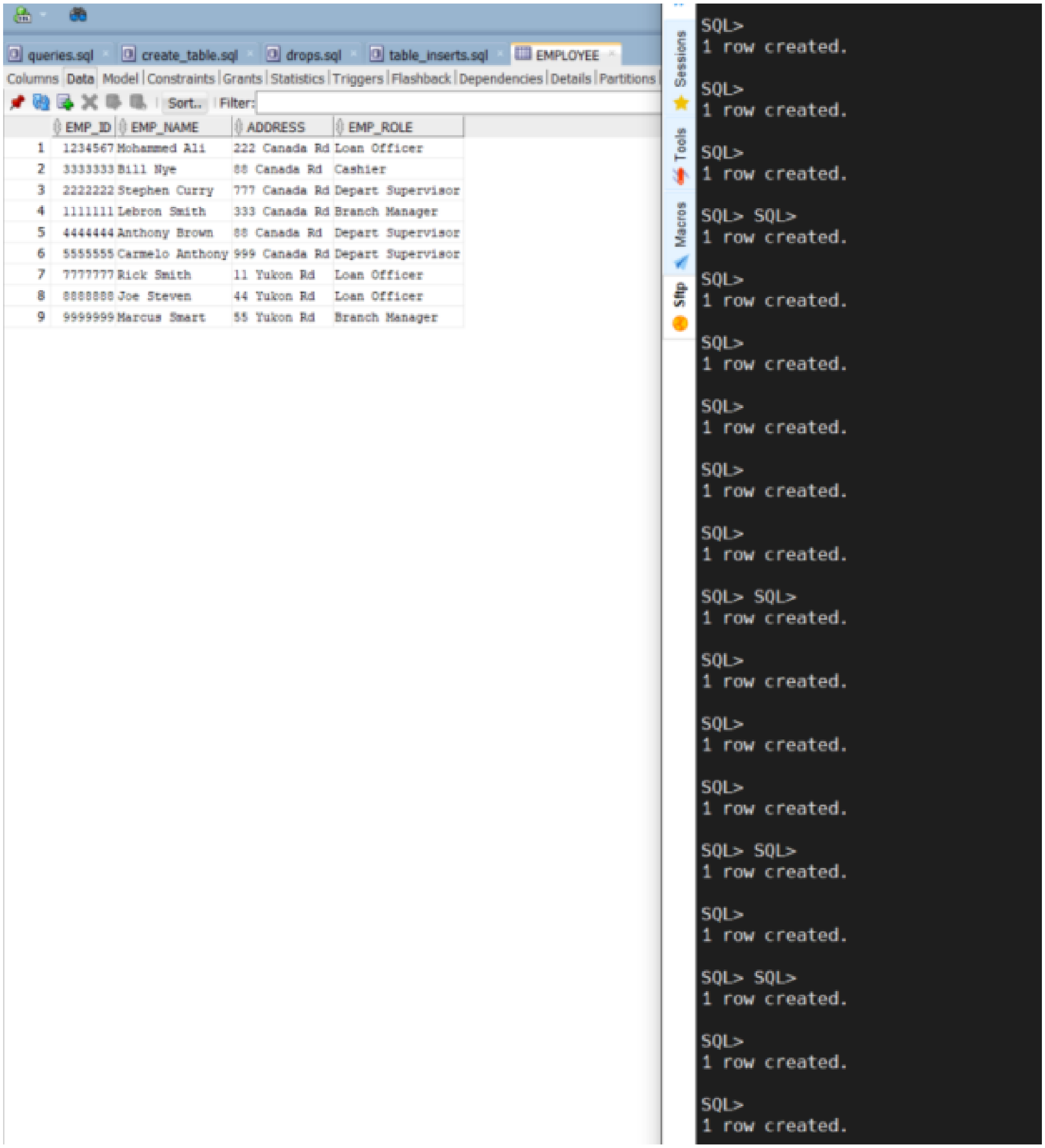
UNIX

Output from selecting “2” to create tables:



```
Connected to:
Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced Analytics
and Real Application Testing options

SQL> SQL> SQL>   2    3    4    5    6    7    8
Table created.

SQL> SQL> SQL>   2    3    4    5
Table created.

SQL> SQL>   2    3    4    5    6
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7    8
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL> SQL> SQL>   2    3    4    5    6    7    8
Table created.

SQL> SQL>   2    3    4    5    6
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7    8    9   10
Table created.

SQL> SQL> SQL>   2    3    4    5    6    7    8    9   10   11
Table created.

SQL> SQL> Disconnected from Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production
With the Partitioning, Oracle Label Security, OLAP, Advanced Analytics
and Real Application Testing options
f5ibrahi@elara:~/cps510$ 
```

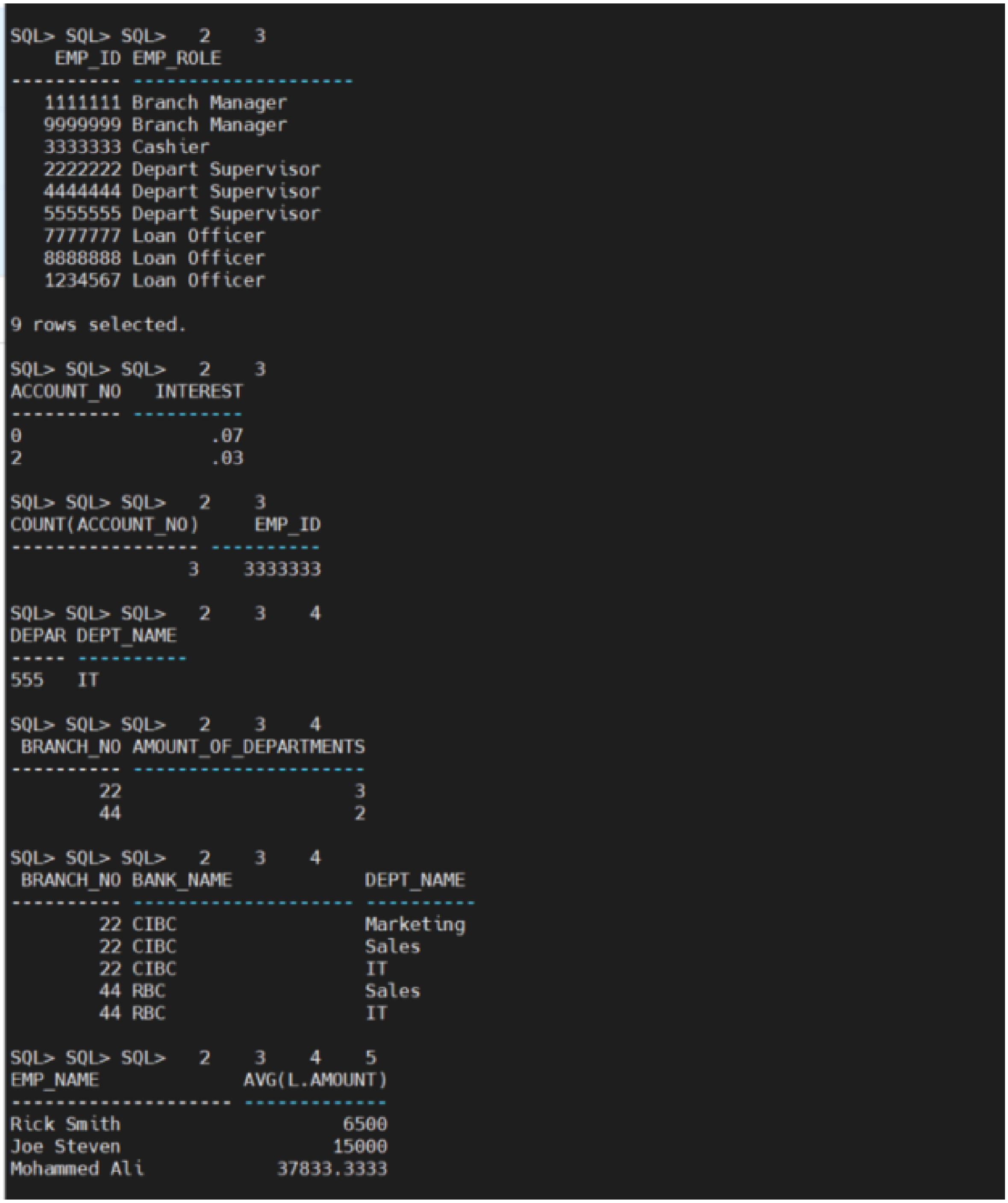
Output from selecting “3” to populate tables. After selecting the employee table we can see it has successfully populated.



	EMP_ID	EMP_NAME	ADDRESS	EMP_ROLE
1	1234567	Mohammed Ali	222 Canada Rd	Loan Officer
2	3333333	Bill Nye	88 Canada Rd	Cashier
3	2222222	Stephen Curry	777 Canada Rd	Depart Supervisor
4	1111111	Lebron Smith	333 Canada Rd	Branch Manager
5	4444444	Anthony Brown	88 Canada Rd	Depart Supervisor
6	5555555	Carmelo Anthony	999 Canada Rd	Depart Supervisor
7	7777777	Rick Smith	11 Yukon Rd	Loan Officer
8	8888888	Joe Steven	44 Yukon Rd	Loan Officer
9	9999999	Marcus Smart	55 Yukon Rd	Branch Manager

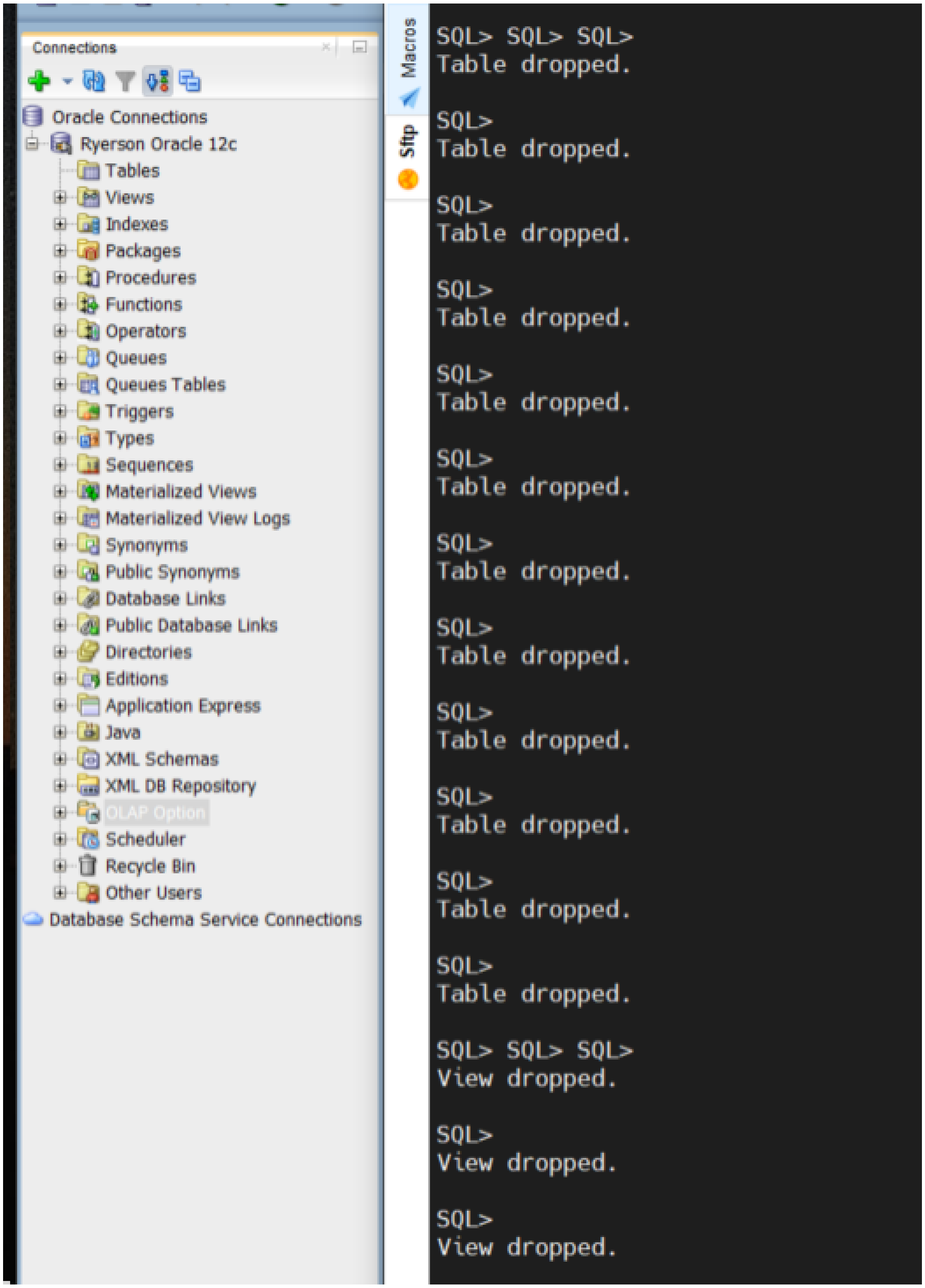
```
SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.
```

Output from selecting “4” to print out queries:



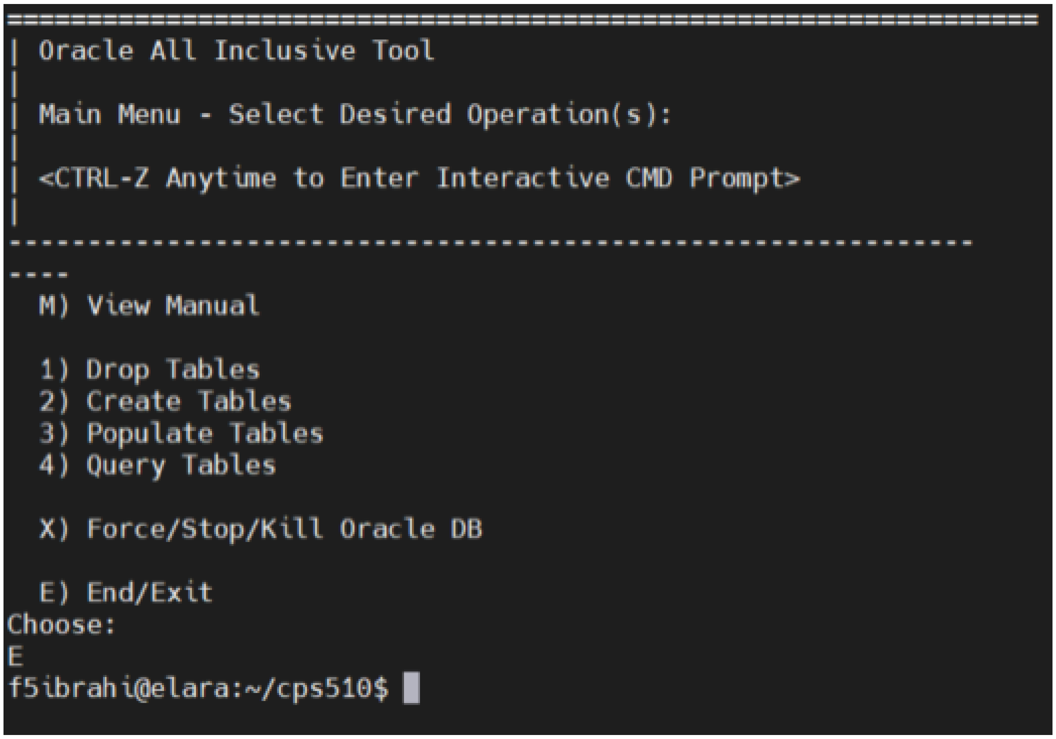
```
SQL> SQL> SQL>   2    3
    EMP_ID EMP_ROLE
---------- --------------------
   1111111 Branch Manager
   9999999 Branch Manager
   3333333 Cashier
   2222222 Depart Supervisor
   4444444 Depart Supervisor
   5555555 Depart Supervisor
   7777777 Loan Officer
   8888888 Loan Officer
   1234567 Loan Officer

9 rows selected.

SQL> SQL> SQL>   2    3
ACCOUNT_NO   INTEREST
---------- ----------
0                 .07
2                 .03

SQL> SQL> SQL>   2    3
COUNT(ACCOUNT_NO)     EMP_ID
----------------- ----------
                3    3333333

SQL> SQL> SQL>   2    3    4
DEPAR DEPT_NAME
----- ----------
555   IT

SQL> SQL> SQL>   2    3    4
 BRANCH_NO AMOUNT_OF_DEPARTMENTS
---------- ---------------------
        22                     3
        44                     2

SQL> SQL> SQL>   2    3    4
 BRANCH_NO BANK_NAME            DEPT_NAME
---------- -------------------- ----------
        22 CIBC                 Marketing
        22 CIBC                 Sales
        22 CIBC                 IT
        44 RBC                  Sales
        44 RBC                  IT

SQL> SQL> SQL>   2    3    4    5
EMP_NAME             AVG(L.AMOUNT)
-------------------- -------------
Rick Smith                    6500
Joe Steven                   15000
Mohammed Ali            37833.3333
```

Output from selecting “4” to drop tables:



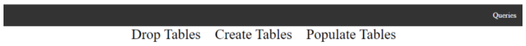
```
SQL> SQL> SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL>
Table dropped.

SQL> SQL> SQL>
View dropped.

SQL>
View dropped.

SQL>
View dropped.
```

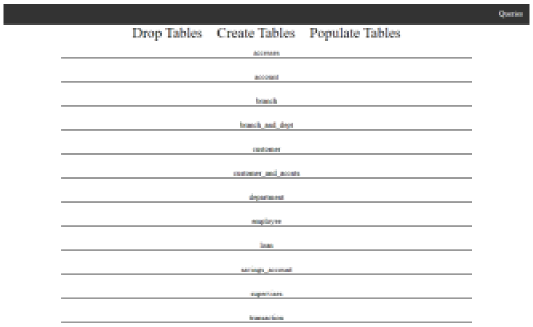
Selecting “E” successfully exits the Main Menu program.



```
=================================================================
| Oracle All Inclusive Tool
|
| Main Menu - Select Desired Operation(s):
|
| <CTRL-Z Anytime to Enter Interactive CMD Prompt>
|
-------------------------------------------------------------
----
  M) View Manual

  1) Drop Tables
  2) Create Tables
  3) Populate Tables
  4) Query Tables

  X) Force/Stop/Kill Oracle DB

  E) End/Exit
Choose:
E
f5ibrahi@elara:~/cps510$ 
```

UI





accesses

account

branch

branch_and_dept

customer

customer_and_accnts

department

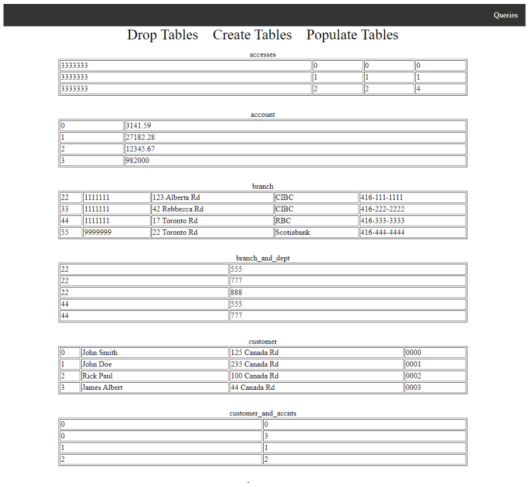
employee

loan

savings_account

supervises

transaction



accesses

3333333	0	0	0
3333333	1	1	1
3333333	2	2	4

account

0	3141.59
1	27182.28
2	12345.67
3	982000

branch

22	1111111	123 Alberta Rd	CIBC	416-111-1111
33	1111111	42 Rebbecca Rd	CIBC	416-222-2222
44	1111111	17 Toronto Rd	RBC	416-333-3333
55	9999999	22 Toronto Rd	Scotiabank	416-444-4444

branch_and_dept

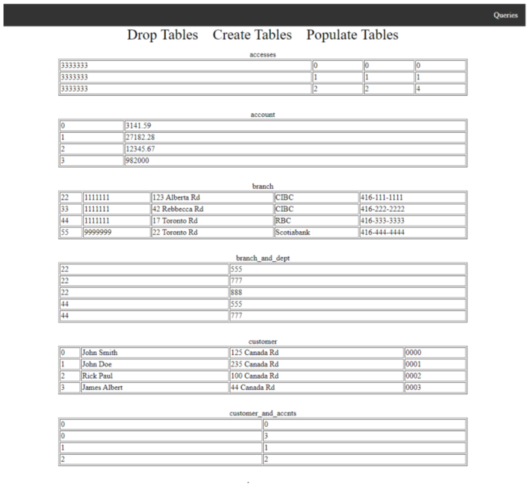
22	555
22	777
22	888
44	555
44	777

customer

0	John Smith	125 Canada Rd	0000
1	John Doe	235 Canada Rd	0001
2	Rick Paul	100 Canada Rd	0002
3	James Albert	44 Canada Rd	0003

customer_and_accnts

0	0
0	3
1	1
2	2



accesses

3333333	0	0	0
3333333	1	1	1
3333333	2	2	4

account

0	3141.59
1	27182.28
2	12345.67
3	982000

branch

22	1111111	123 Alberta Rd	CIBC	416-111-1111
33	1111111	42 Rebbecca Rd	CIBC	416-222-2222
44	1111111	17 Toronto Rd	RBC	416-333-3333
55	9999999	22 Toronto Rd	Scotiabank	416-444-4444

branch_and_dept

22	555
22	777
22	888
44	555
44	777

customer

0	John Smith	125 Canada Rd	0000
1	John Doe	235 Canada Rd	0001
2	Rick Paul	100 Canada Rd	0002
3	James Albert	44 Canada Rd	0003

customer_and_accnts

0	0
0	3
1	1
2	2

Since the webpage is directly connected to the Ryerson server, there is no need to install anything, just simply load the page. However in order to access the webpage one must be connected to the Ryerson VPN

https://webdev.scs.ryerson.ca/~f5ibrahi/menu.php

The webpage created performs 4 functions, when loaded the webpage has an empty screen such as the top left image, this also occurs when using the drop tables button. Another function of the webpage is to create tables, which is the middle button, when clicked it provides the image in the top right. In order to populate tables, you click the populate tables button and the webpage will load an image such as the one shown in the bottom left corner. Lastly when performing queries, the webpage loads an image shown in the bottom right corner.