Phase 5

Toronto Metropolitan University

Simon Lin (501103322),
Dylan Ha (501056670),
Enes Polat (501061594)

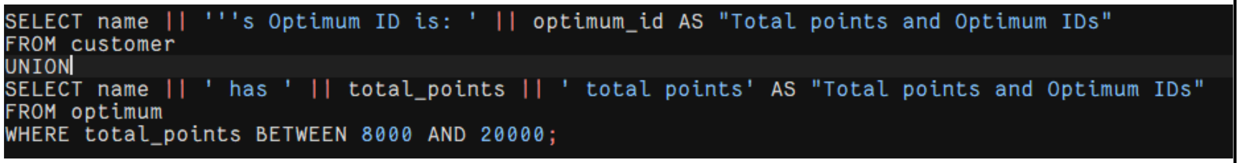
CPS510 - Database Systems

Point of Sale System for Shopper Drug Marts

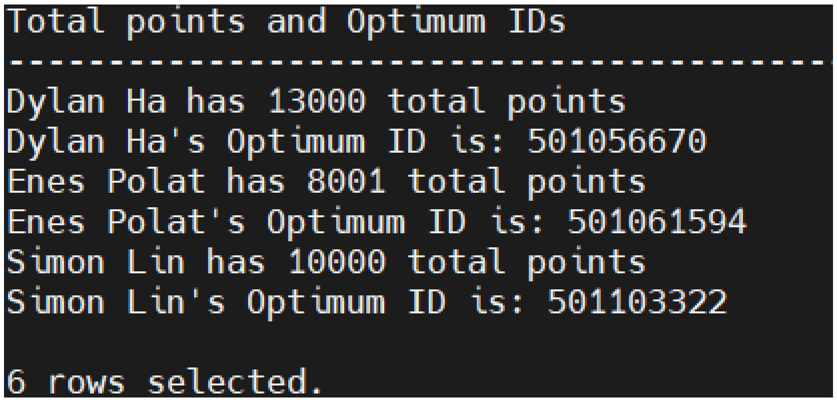
Interesting Advanced Queries:

Union:

Description: Returns a list of customers and their corresponding optimum ids with points ranging from 8000 to 20000.
Combines two tables using UNION, will delete the duplicate values



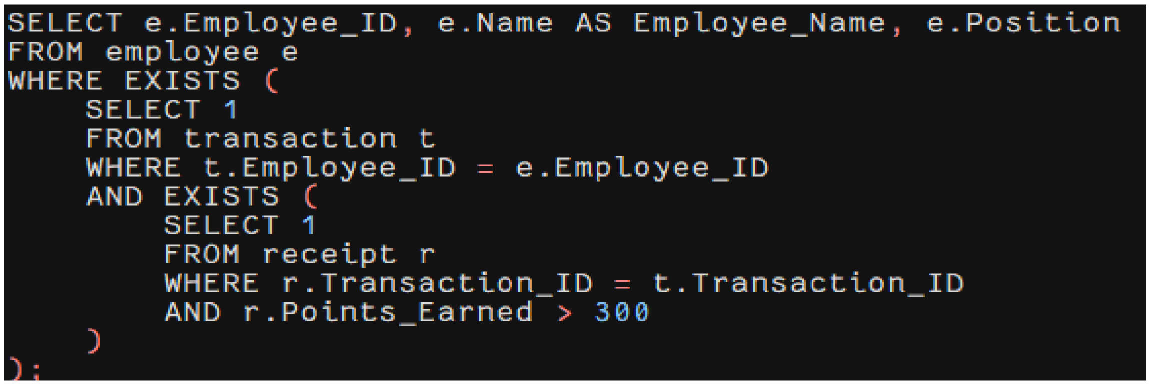
```
SELECT name || '''s Optimum ID is: ' || optimum_id AS "Total points and Optimum IDs"
FROM customer
UNION
SELECT name || ' has ' || total_points || ' total points' AS "Total points and Optimum IDs"
FROM optimum
WHERE total_points BETWEEN 8000 AND 20000;
```



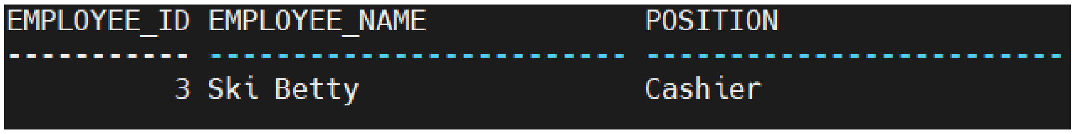
```
Total points and Optimum IDs
----------------------------------------
Dylan Ha has 13000 total points
Dylan Ha's Optimum ID is: 501056670
Enes Polat has 8001 total points
Enes Polat's Optimum ID is: 501061594
Simon Lin has 10000 total points
Simon Lin's Optimum ID is: 501103322

6 rows selected.
```

Exists:

Description: Retrieves the Employee_ID, Employee_Name, and Position of employees who have processed transactions resulting in receipts with points over 300. It uses the EXISTS operator to check if there are any related transactions for each employee, ensuring they are only included if at least one corresponding receipt meets the specified points condition. (select 1 = checks if one row exists that matches))



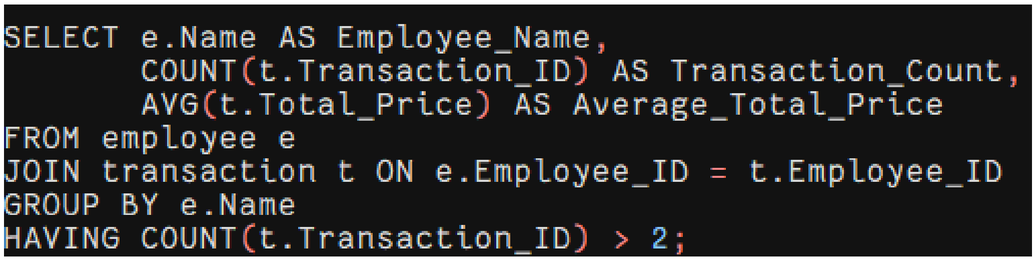
```
SELECT e.Employee_ID, e.Name AS Employee_Name, e.Position
FROM employee e
WHERE EXISTS (
    SELECT 1
    FROM transaction t
    WHERE t.Employee_ID = e.Employee_ID
    AND EXISTS (
        SELECT 1
        FROM receipt r
        WHERE r.Transaction_ID = t.Transaction_ID
        AND r.Points_Earned > 300
    )
);
```



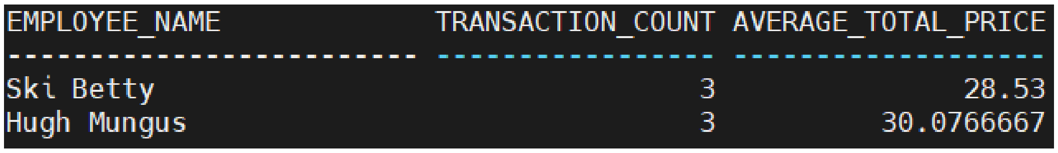
```
EMPLOYEE_ID EMPLOYEE_NAME                POSITION
----------- ------------------------- -------------------------
          3 Ski Betty                 Cashier
```

Count & Having:

Description: Returns a table of employees that have completed at least two sales, averaging the price of their total sales price.
Tables are grouped by employee names.



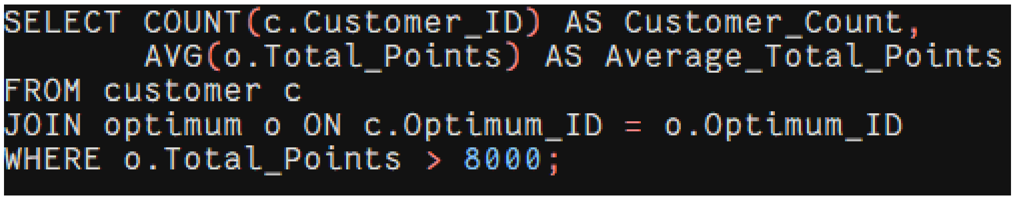
```
SELECT e.Name AS Employee_Name,
       COUNT(t.Transaction_ID) AS Transaction_Count,
       AVG(t.Total_Price) AS Average_Total_Price
FROM employee e
JOIN transaction t ON e.Employee_ID = t.Employee_ID
GROUP BY e.Name
HAVING COUNT(t.Transaction_ID) > 2;
```



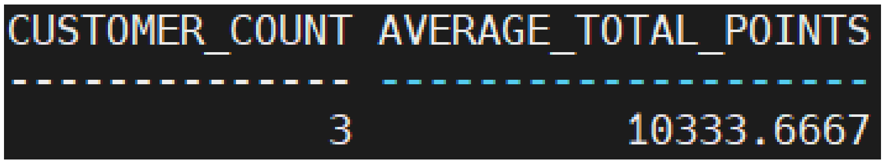
```
EMPLOYEE_NAME             TRANSACTION_COUNT AVERAGE_TOTAL_PRICE
------------------------- ----------------- -------------------
Ski Betty                                 3               28.53
Hugh Mungus                               3          30.0766667
```

Count:

Description: Returns the number of customers who have a total points balance greater than 8,000 and calculates the average total points for those customers.

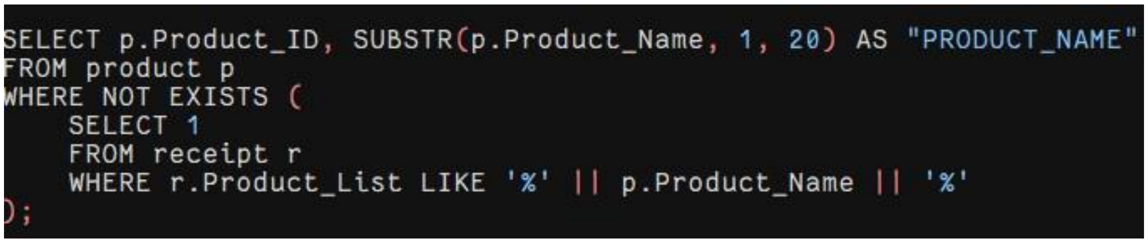


```
SELECT COUNT(c.Customer_ID) AS Customer_Count,
       AVG(o.Total_Points) AS Average_Total_Points
FROM customer c
JOIN optimum o ON c.Optimum_ID = o.Optimum_ID
WHERE o.Total_Points > 8000;
```

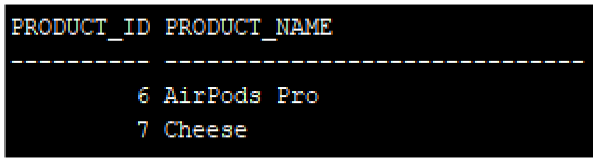


```
CUSTOMER_COUNT AVERAGE_TOTAL_POINTS
-------------- --------------------
             3           10333.6667
```

NOT EXISTS: Returns the number of products that haven't been sold yet. Uses a NOT EXISTS clause to check receipts for products.

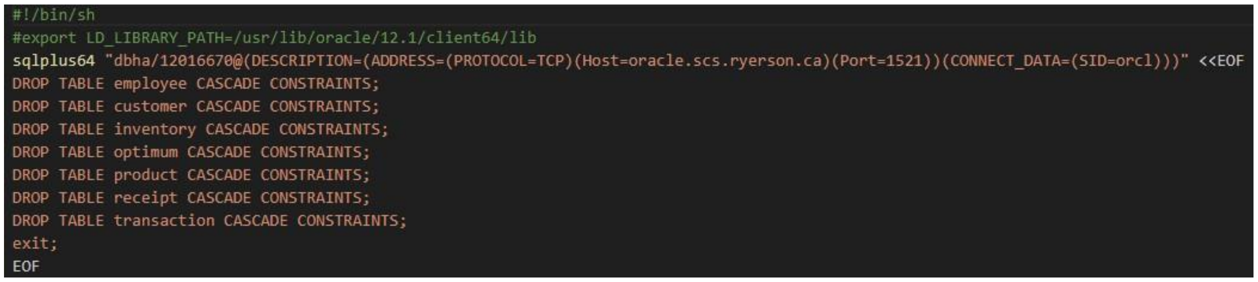


```
SELECT p.Product_ID, SUBSTR(p.Product_Name, 1, 20) AS "PRODUCT_NAME"
FROM product p
WHERE NOT EXISTS (
    SELECT 1
    FROM receipt r
    WHERE r.Product_List LIKE '%' || p.Product_Name || '%'
);
```



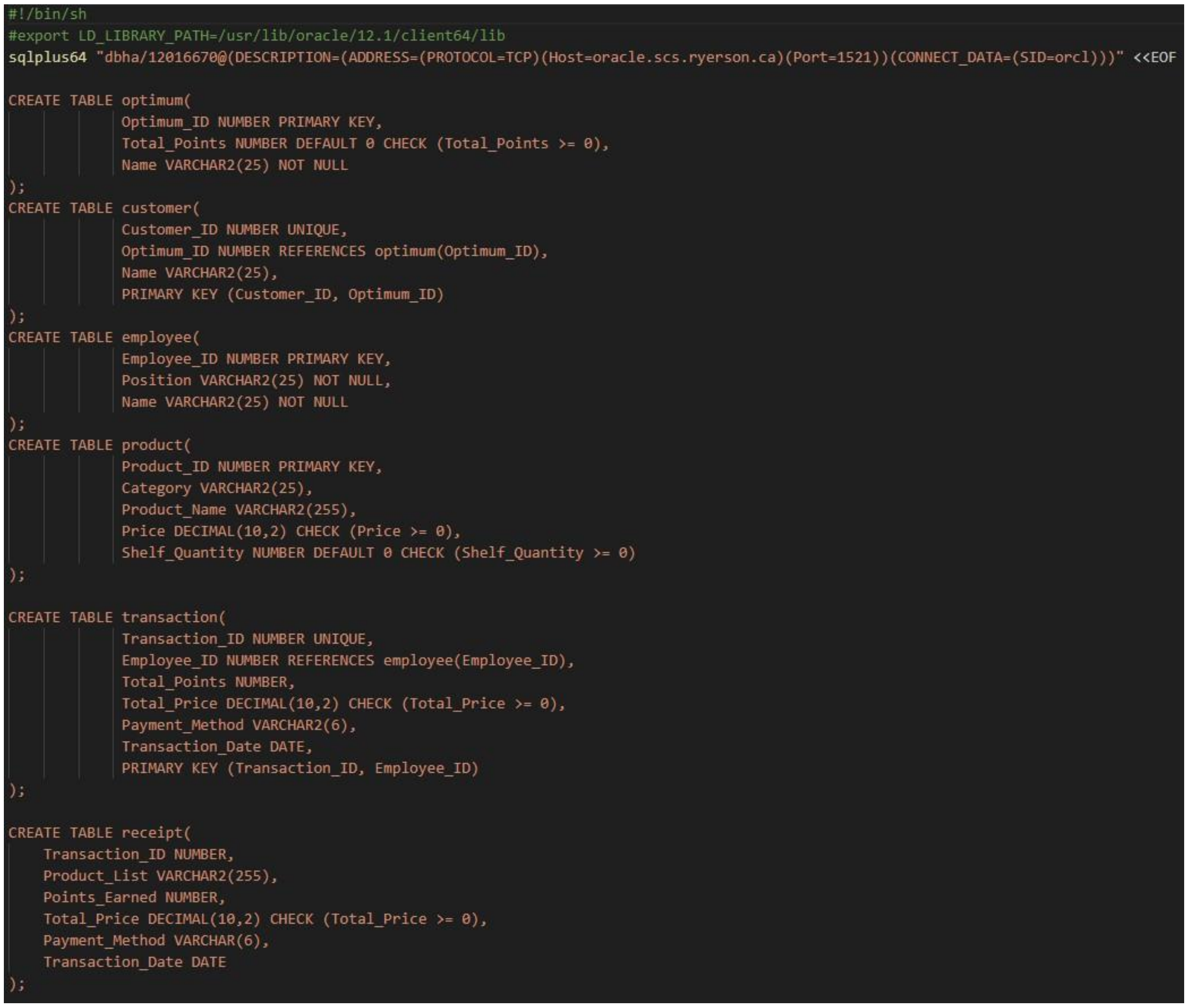
```
PRODUCT_ID PRODUCT_NAME
---------- ------------------------------
         6 AirPods Pro
         7 Cheese
```

Source Code for Unix Implementation -
Included in the Screenshots and LAB 5 code submission
Drop Tables:



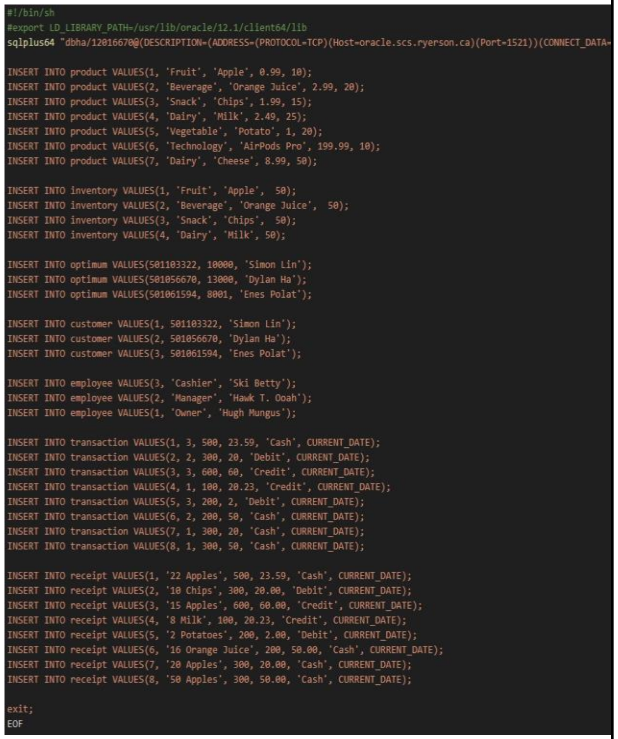
```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "dbha/12016670@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF
DROP TABLE employee CASCADE CONSTRAINTS;
DROP TABLE customer CASCADE CONSTRAINTS;
DROP TABLE inventory CASCADE CONSTRAINTS;
DROP TABLE optimum CASCADE CONSTRAINTS;
DROP TABLE product CASCADE CONSTRAINTS;
DROP TABLE receipt CASCADE CONSTRAINTS;
DROP TABLE transaction CASCADE CONSTRAINTS;
exit;
EOF
```

Create Tables:

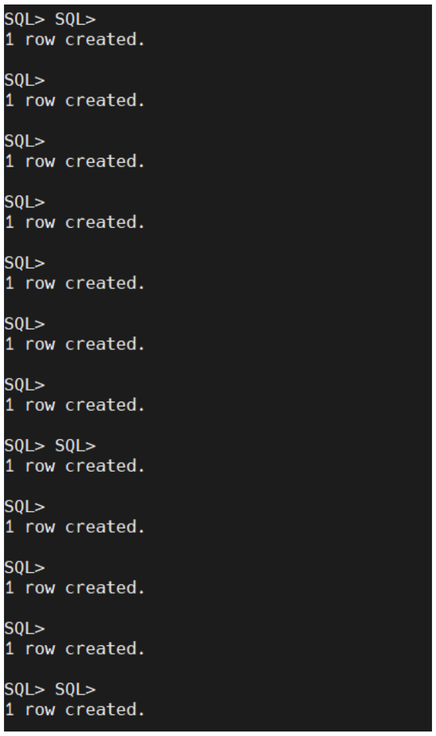


```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "dbha/12016670@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

CREATE TABLE optimum(
            Optimum_ID NUMBER PRIMARY KEY,
            Total_Points NUMBER DEFAULT 0 CHECK (Total_Points >= 0),
            Name VARCHAR2(25) NOT NULL
);
CREATE TABLE customer(
            Customer_ID NUMBER UNIQUE,
            Optimum_ID NUMBER REFERENCES optimum(Optimum_ID),
            Name VARCHAR2(25),
            PRIMARY KEY (Customer_ID, Optimum_ID)
);
CREATE TABLE employee(
            Employee_ID NUMBER PRIMARY KEY,
            Position VARCHAR2(25) NOT NULL,
            Name VARCHAR2(25) NOT NULL
);
CREATE TABLE product(
            Product_ID NUMBER PRIMARY KEY,
            Category VARCHAR2(25),
            Product_Name VARCHAR2(255),
            Price DECIMAL(10,2) CHECK (Price >= 0),
            Shelf_Quantity NUMBER DEFAULT 0 CHECK (Shelf_Quantity >= 0)
);

CREATE TABLE transaction(
            Transaction_ID NUMBER UNIQUE,
            Employee_ID NUMBER REFERENCES employee(Employee_ID),
            Total_Points NUMBER,
            Total_Price DECIMAL(10,2) CHECK (Total_Price >= 0),
            Payment_Method VARCHAR2(6),
            Transaction_Date DATE,
            PRIMARY KEY (Transaction_ID, Employee_ID)
);

CREATE TABLE receipt(
    Transaction_ID NUMBER,
    Product_List VARCHAR2(255),
    Points_Earned NUMBER,
    Total_Price DECIMAL(10,2) CHECK (Total_Price >= 0),
    Payment_Method VARCHAR(6),
    Transaction_Date DATE
);
```

Populate Tables:

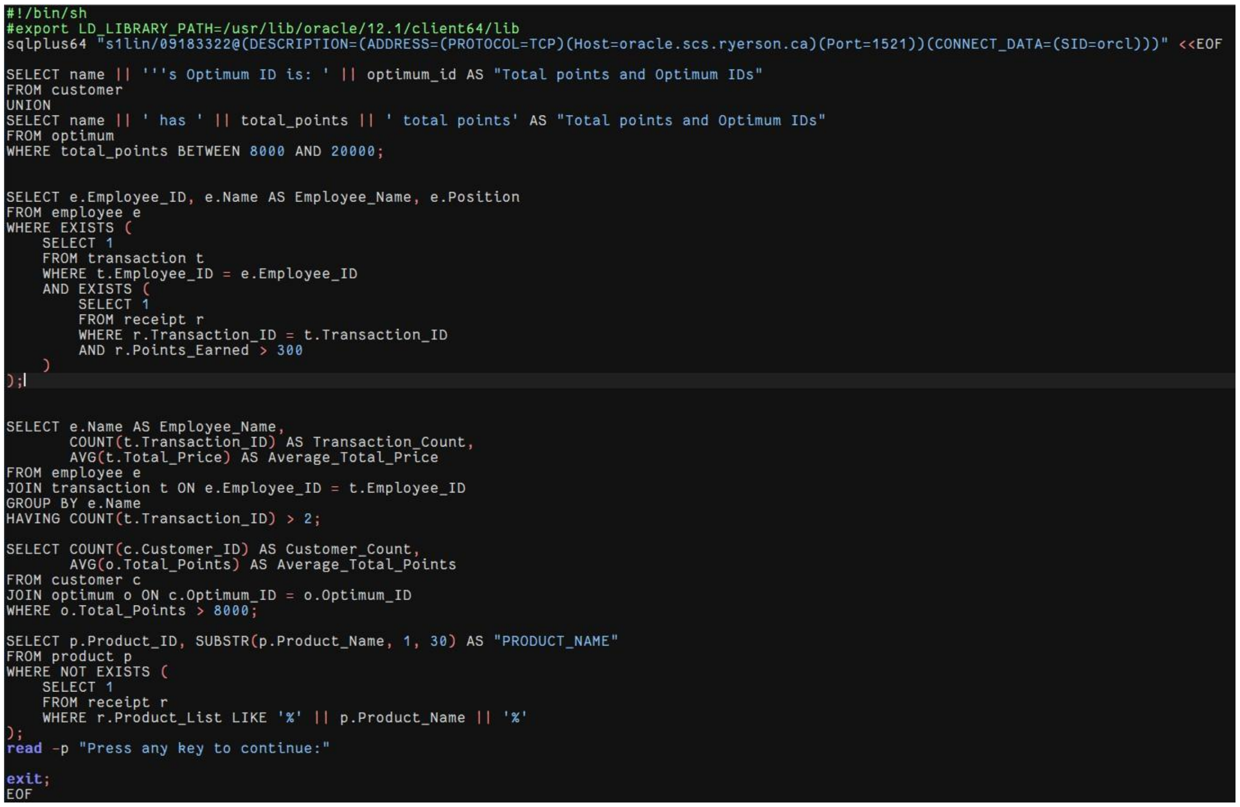


```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "dbha/12016670@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=

INSERT INTO product VALUES(1, 'Fruit', 'Apple', 0.99, 10);
INSERT INTO product VALUES(2, 'Beverage', 'Orange Juice', 2.99, 20);
INSERT INTO product VALUES(3, 'Snack', 'Chips', 1.99, 15);
INSERT INTO product VALUES(4, 'Dairy', 'Milk', 2.49, 25);
INSERT INTO product VALUES(5, 'Vegetable', 'Potato', 1, 20);
INSERT INTO product VALUES(6, 'Technology', 'AirPods Pro', 199.99, 10);
INSERT INTO product VALUES(7, 'Dairy', 'Cheese', 8.99, 50);

INSERT INTO inventory VALUES(1, 'Fruit', 'Apple',  50);
INSERT INTO inventory VALUES(2, 'Beverage', 'Orange Juice',  50);
INSERT INTO inventory VALUES(3, 'Snack', 'Chips',  50);
INSERT INTO inventory VALUES(4, 'Dairy', 'Milk', 50);

INSERT INTO optimum VALUES(501103322, 10000, 'Simon Lin');
INSERT INTO optimum VALUES(501056670, 13000, 'Dylan Ha');
INSERT INTO optimum VALUES(501061594, 8001, 'Enes Polat');

INSERT INTO customer VALUES(1, 501103322, 'Simon Lin');
INSERT INTO customer VALUES(2, 501056670, 'Dylan Ha');
INSERT INTO customer VALUES(3, 501061594, 'Enes Polat');

INSERT INTO employee VALUES(3, 'Cashier', 'Ski Betty');
INSERT INTO employee VALUES(2, 'Manager', 'Hawk T. Ooah');
INSERT INTO employee VALUES(1, 'Owner', 'Hugh Mungus');

INSERT INTO transaction VALUES(1, 3, 500, 23.59, 'Cash', CURRENT_DATE);
INSERT INTO transaction VALUES(2, 2, 300, 20, 'Debit', CURRENT_DATE);
INSERT INTO transaction VALUES(3, 3, 600, 60, 'Credit', CURRENT_DATE);
INSERT INTO transaction VALUES(4, 1, 100, 20.23, 'Credit', CURRENT_DATE);
INSERT INTO transaction VALUES(5, 3, 200, 2, 'Debit', CURRENT_DATE);
INSERT INTO transaction VALUES(6, 2, 200, 50, 'Cash', CURRENT_DATE);
INSERT INTO transaction VALUES(7, 1, 300, 20, 'Cash', CURRENT_DATE);
INSERT INTO transaction VALUES(8, 1, 300, 50, 'Cash', CURRENT_DATE);

INSERT INTO receipt VALUES(1, '22 Apples', 500, 23.59, 'Cash', CURRENT_DATE);
INSERT INTO receipt VALUES(2, '10 Chips', 300, 20.00, 'Debit', CURRENT_DATE);
INSERT INTO receipt VALUES(3, '15 Apples', 600, 60.00, 'Credit', CURRENT_DATE);
INSERT INTO receipt VALUES(4, '8 Milk', 100, 20.23, 'Credit', CURRENT_DATE);
INSERT INTO receipt VALUES(5, '2 Potatoes', 200, 2.00, 'Debit', CURRENT_DATE);
INSERT INTO receipt VALUES(6, '16 Orange Juice', 200, 50.00, 'Cash', CURRENT_DATE);
INSERT INTO receipt VALUES(7, '20 Apples', 300, 20.00, 'Cash', CURRENT_DATE);
INSERT INTO receipt VALUES(8, '50 Apples', 300, 50.00, 'Cash', CURRENT_DATE);

exit;
EOF
```



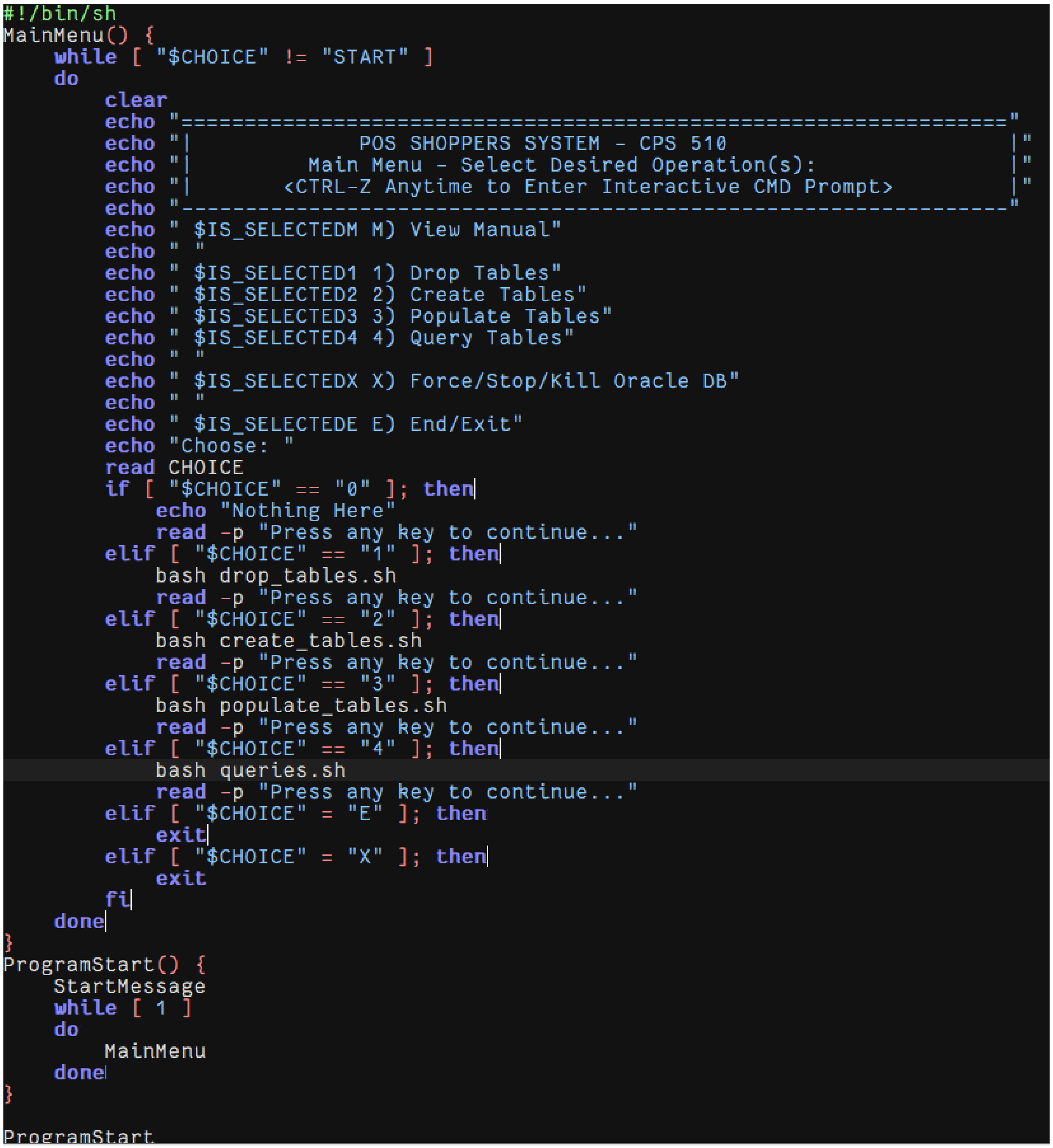
```
SQL> SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
1 row created.

SQL>
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.
```

Query Tables:



```
#!/bin/sh
#export LD_LIBRARY_PATH=/usr/lib/oracle/12.1/client64/lib
sqlplus64 "s1lin/09183322@(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=oracle.scs.ryerson.ca)(Port=1521))(CONNECT_DATA=(SID=orcl)))" <<EOF

SELECT name || '''s Optimum ID is: ' || optimum_id AS "Total points and Optimum IDs"
FROM customer
UNION
SELECT name || ' has ' || total_points || ' total points' AS "Total points and Optimum IDs"
FROM optimum
WHERE total_points BETWEEN 8000 AND 20000;


SELECT e.Employee_ID, e.Name AS Employee_Name, e.Position
FROM employee e
WHERE EXISTS (
    SELECT 1
    FROM transaction t
    WHERE t.Employee_ID = e.Employee_ID
    AND EXISTS (
        SELECT 1
        FROM receipt r
        WHERE r.Transaction_ID = t.Transaction_ID
        AND r.Points_Earned > 300
    )
);


SELECT e.Name AS Employee_Name,
       COUNT(t.Transaction_ID) AS Transaction_Count,
       AVG(t.Total_Price) AS Average_Total_Price
FROM employee e
JOIN transaction t ON e.Employee_ID = t.Employee_ID
GROUP BY e.Name
HAVING COUNT(t.Transaction_ID) > 2;

SELECT COUNT(c.Customer_ID) AS Customer_Count,
       AVG(o.Total_Points) AS Average_Total_Points
FROM customer c
JOIN optimum o ON c.Optimum_ID = o.Optimum_ID
WHERE o.Total_Points > 8000;

SELECT p.Product_ID, SUBSTR(p.Product_Name, 1, 30) AS "PRODUCT_NAME"
FROM product p
WHERE NOT EXISTS (
    SELECT 1
    FROM receipt r
    WHERE r.Product_List LIKE '%' || p.Product_Name || '%'
);
read -p "Press any key to continue:"

exit;
EOF
```

Menu:



```
#!/bin/sh
MainMenu() {
    while [ "$CHOICE" != "START" ]
    do
        clear
        echo "================================================================="
        echo "|             POS SHOPPERS SYSTEM - CPS 510                      |"
        echo "|         Main Menu - Select Desired Operation(s):               |"
        echo "|       <CTRL-Z Anytime to Enter Interactive CMD Prompt>         |"
        echo "-----------------------------------------------------------------"
        echo " $IS_SELECTEDM M) View Manual"
        echo " "
        echo " $IS_SELECTED1 1) Drop Tables"
        echo " $IS_SELECTED2 2) Create Tables"
        echo " $IS_SELECTED3 3) Populate Tables"
        echo " $IS_SELECTED4 4) Query Tables"
        echo " "
        echo " $IS_SELECTEDX X) Force/Stop/Kill Oracle DB"
        echo " "
        echo " $IS_SELECTEDE E) End/Exit"
        echo "Choose: "
        read CHOICE
        if [ "$CHOICE" == "0" ]; then
            echo "Nothing Here"
            read -p "Press any key to continue..."
        elif [ "$CHOICE" == "1" ]; then
            bash drop_tables.sh
            read -p "Press any key to continue..."
        elif [ "$CHOICE" == "2" ]; then
            bash create_tables.sh
            read -p "Press any key to continue..."
        elif [ "$CHOICE" == "3" ]; then
            bash populate_tables.sh
            read -p "Press any key to continue..."
        elif [ "$CHOICE" == "4" ]; then
            bash queries.sh
            read -p "Press any key to continue..."
        elif [ "$CHOICE" = "E" ]; then
            exit
        elif [ "$CHOICE" = "X" ]; then
            exit
        fi
    done
}
ProgramStart() {
    StartMessage
    while [ 1 ]
    do
        MainMenu
    done
}

ProgramStart
```