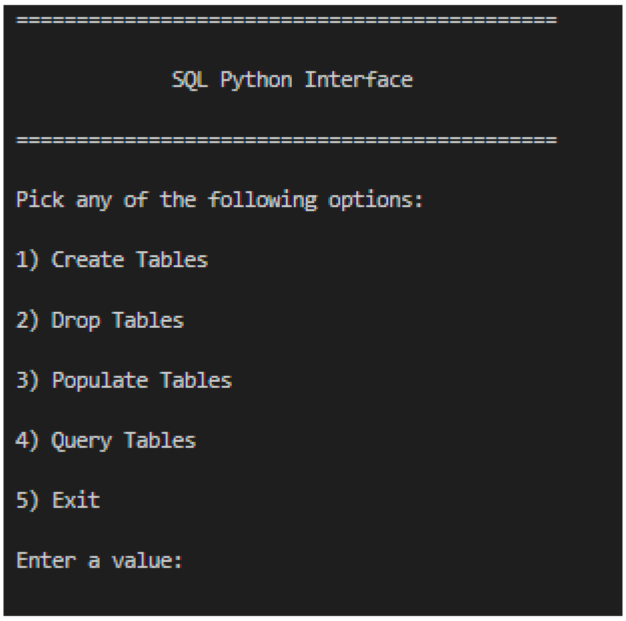
Athavan Thanaraj - 501027195 - Section7
Hashim Farooq - 501048604 - Section7
Umair Tariq - 501050119 - Section7

Lab 9

Python Shell Menu:

The purpose of this Python Shell Menu below is to give us a user interface in order to make modifications to our database such as Drop, Create, Populate and Query Tables through the Python Program.

Python Shell Menu Screenshot:



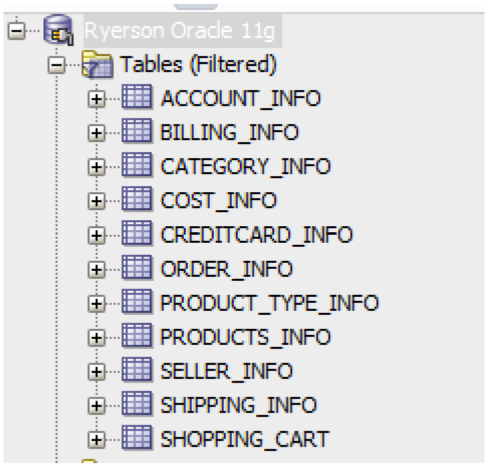
```
=============================================

             SQL Python Interface

=============================================

Pick any of the following options:

1) Create Tables

2) Drop Tables

3) Populate Tables

4) Query Tables

5) Exit

Enter a value:
```

DROP TABLES option:

The drop tables option allows us to drop the tables that we have created in the create tables option (which will be explained further below), once the drop tables option is selected it will drop the following tables as well as cascade all constraints in the database connected.

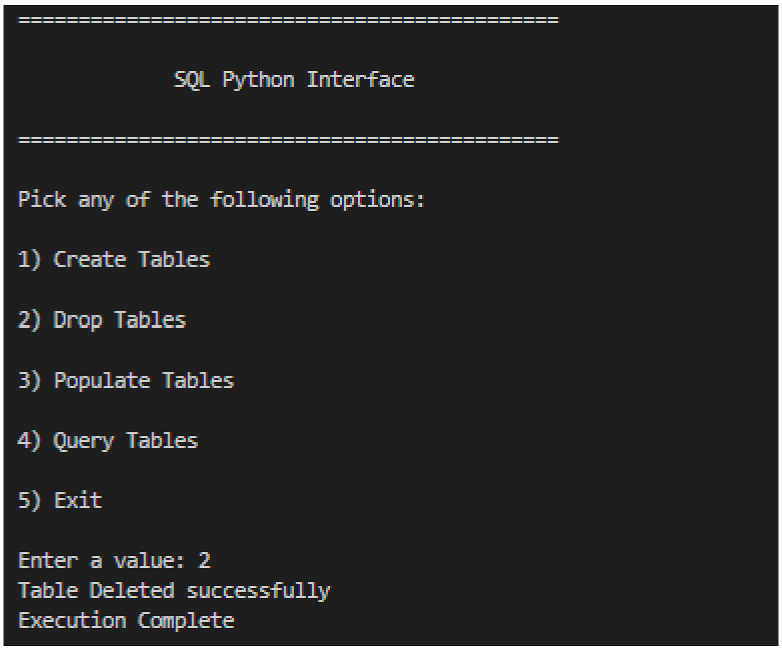
DROP TABLES screenshot:

Here we have the tables created:



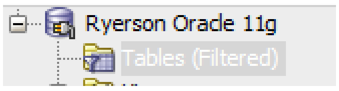
```
Ryerson Oracle 11g
  Tables (Filtered)
    ACCOUNT_INFO
    BILLING_INFO
    CATEGORY_INFO
    COST_INFO
    CREDITCARD_INFO
    ORDER_INFO
    PRODUCT_TYPE_INFO
    PRODUCTS_INFO
    SELLER_INFO
    SHIPPING_INFO
    SHOPPING_CART
```

Here the we select the option to delete the tables:



```
=============================================

             SQL Python Interface

=============================================

Pick any of the following options:

1) Create Tables

2) Drop Tables

3) Populate Tables

4) Query Tables

5) Exit

Enter a value: 2
Table Deleted successfully
Execution Complete
```

Here we can see the changes in the database:

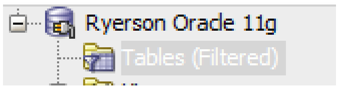


CREATE TABLES option:

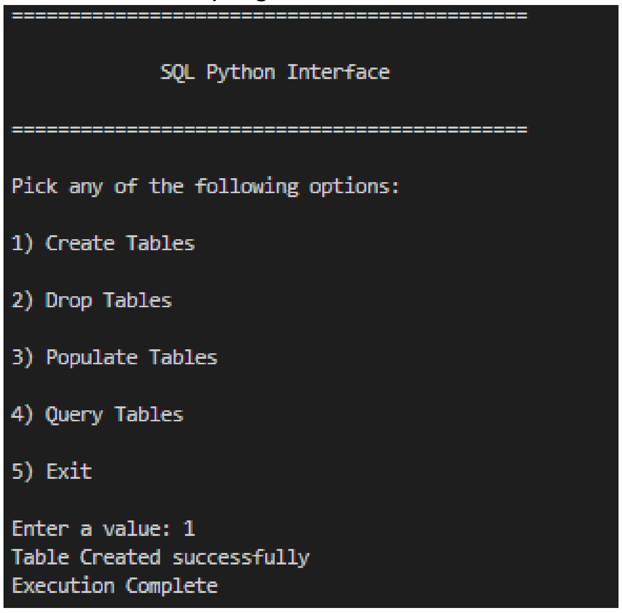
The create tables option allows us to create specified tables in the database that we are connected to. After these tables are created in the database, we can then populate, drop, or even query them.

CREATE TABLES screenshot:

Here we can see that the tables are empty:

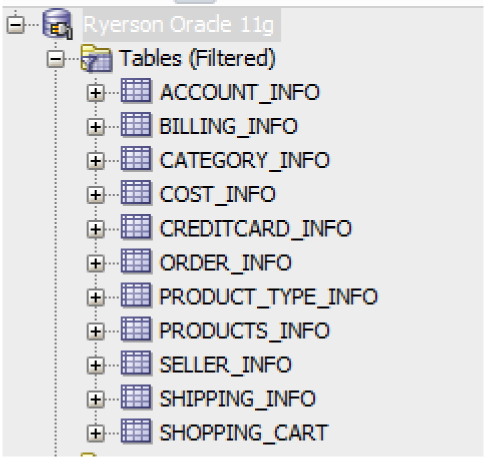


Then we run the option to create tables in our program:



```
=============================================

             SQL Python Interface

=============================================

Pick any of the following options:

1) Create Tables

2) Drop Tables

3) Populate Tables

4) Query Tables

5) Exit

Enter a value: 1
Table Created successfully
Execution Complete
```

Here we can see the changes created in our program:



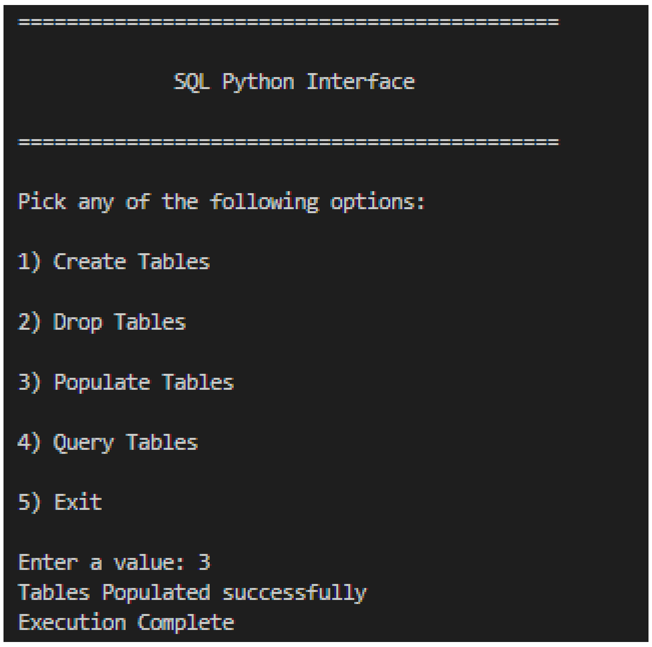
```
Ryerson Oracle 11g
  Tables (Filtered)
    ACCOUNT_INFO
    BILLING_INFO
    CATEGORY_INFO
    COST_INFO
    CREDITCARD_INFO
    ORDER_INFO
    PRODUCT_TYPE_INFO
    PRODUCTS_INFO
    SELLER_INFO
    SHIPPING_INFO
    SHOPPING_CART
```

<u>POPULATE TABLES option:</u>

The populate tables option allows us to populate the tables that we have created above using the Python shell menu we have created.

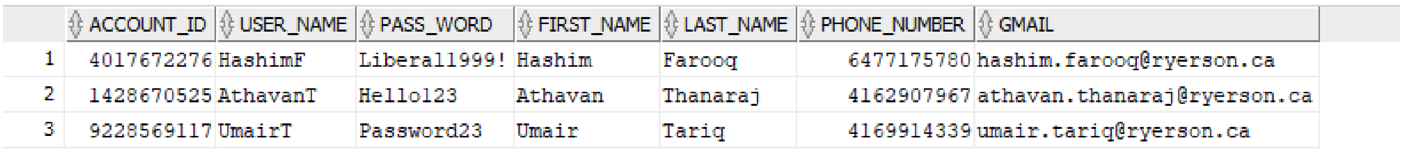
<u>POPULATE TABLES screenshot:</u>

Here we select the option to populate the tables:

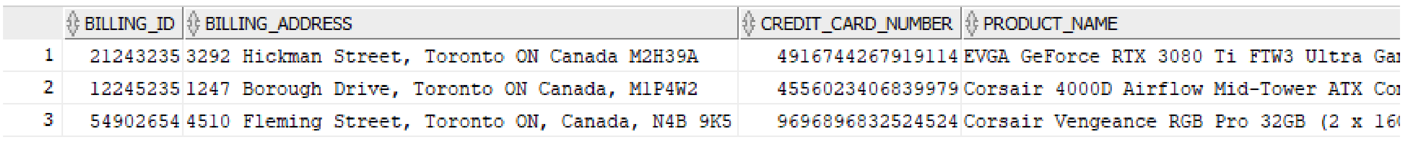


```
==============================================

            SQL Python Interface

==============================================

Pick any of the following options:

1) Create Tables

2) Drop Tables

3) Populate Tables

4) Query Tables

5) Exit

Enter a value: 3
Tables Populated successfully
Execution Complete
```

And we can see that tables have been populated in database:



	ACCOUNT_ID	USER_NAME	PASS_WORD	FIRST_NAME	LAST_NAME	PHONE_NUMBER	GMAIL
1	4017672276	HashimF	Liberal1999!	Hashim	Farooq	6477175780	hashim.farooq@ryerson.ca
2	1428670525	AthavanT	Hello123	Athavan	Thanaraj	4162907967	athavan.thanaraj@ryerson.ca
3	9228569117	UmairT	Password23	Umair	Tariq	4169914339	umair.tariq@ryerson.ca



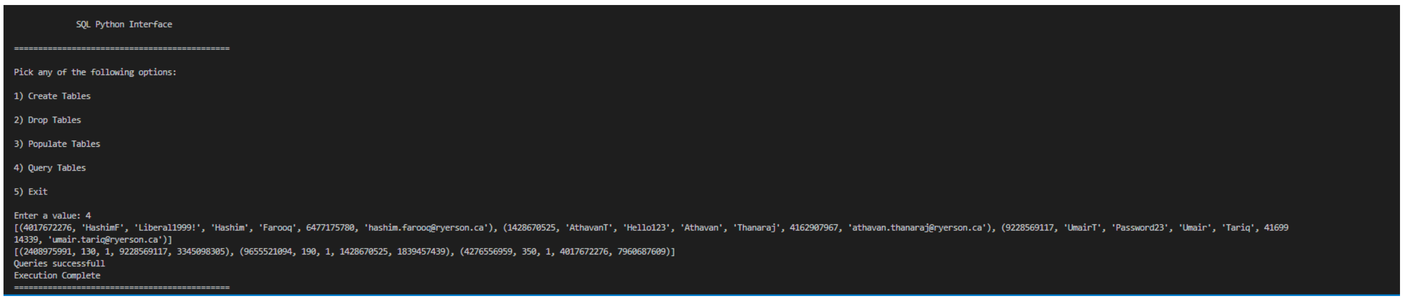
	BILLING_ID	BILLING_ADDRESS	CREDIT_CARD_NUMBER	PRODUCT_NAME
1	21243235	3292 Hickman Street, Toronto ON Canada M2H39A	4916744267919114	EVGA GeForce RTX 3080 Ti FTW3 Ultra Ga
2	12245235	1247 Borough Drive, Toronto ON Canada, M1P4W2	4556023406839979	Corsair 4000D Airflow Mid-Tower ATX Co
3	54902654	4510 Fleming Street, Toronto ON, Canada, N4B 9K5	9696896832524524	Corsair Vengeance RGB Pro 32GB (2 x 16

QUERY TABLES option:

The query tables option allows us to query the tables we have created using the Python shell menu options.

QUERY TABLES screenshot:

Here we select the option to query some tables and then our specified queries are ran and output a result:



```
          SQL Python Interface

============================================

Pick any of the following options:

1) Create Tables

2) Drop Tables

3) Populate Tables

4) Query Tables

5) Exit

Enter a value: 4
[(4017672276, 'HashimF', 'Liberal1999!', 'Hashim', 'Farooq', 6477175780, 'hashim.farooq@ryerson.ca'), (1428670525, 'AthavanT', 'Hello123', 'Athavan', 'Thanaraj', 4162907967, 'athavan.thanaraj@ryerson.ca'), (9228569117, 'UmairT', 'Password23', 'Umair', 'Tariq', 41699
14339, 'umair.tariq@ryerson.ca')]
[(2408975991, 130, 1, 9228569117, 3345098305), (9655521094, 190, 1, 1428670525, 1839457439), (4276556959, 350, 1, 4017672276, 7960687609)]
Queries successfull
Execution Complete
============================================
```

OUR CODE FOR A9:

```python
import cx_Oracle
import os


def MainMenu():
    PrintMenu()
    MenuSelection()
    exit()


def MenuSelection():
    select = input("Enter a value: ")
    if (select == "1"):
        CreateTable()
        # os.system('cls')
        PrintMenu()
        MenuSelection()
    elif (select == "2"):
        DropTables()
        # os.system('cls')
        PrintMenu()
        MenuSelection()
    elif (select == "3"):
        PopTables()
        # os.system('cls')
        PrintMenu()
        MenuSelection()
    elif (select == "4"):
        QueryTable()
        # os.system('cls')
        PrintMenu()
        MenuSelection()
    elif (select == "5"):
        print("Bye")
        exit()
    else:
        print("Invalid input please try again!!!")
        # os.system('cls')
        PrintMenu()
        MenuSelection()


def PrintMenu():
    # os.system('cls')
    print("=============================================\n")
    print("             SQL Python Interface            \n")
    print("=============================================\n")
    print("Pick any of the following options: \n")
    print("1) Create Tables \n")
    print("2) Drop Tables \n")
    print("3) Populate Tables \n")
    print("4) Query Tables \n")
    print("5) Exit \n")


def CreateTable():
    dsnStr = cx_Oracle.makedsn("oracle.scs.ryerson.ca", "1521",
sid="orcl")

    con = None
    try:
        con = cx_Oracle.connect(
            user="athanara", password="07087195", dsn=dsnStr)

        cur = con.cursor()

        cur.execute(
            "CREATE TABLE account_info(account_id INTEGER PRIMARY
KEY,user_name VARCHAR(100) NOT NULL UNIQUE,pass_word VARCHAR(30) NOT
NULL,first_name VARCHAR(30) NOT NULL,last_name VARCHAR(30) NOT
NULL,phone_number INTEGER NOT NULL UNIQUE,gmail VARCHAR(100) NOT NULL
UNIQUE)"
        )

        cur.execute(
            "CREATE TABLE seller_info(seller_id INTEGER NOT NULL PRIMARY
KEY, user_name VARCHAR(100) NOT NULL UNIQUE, pass_word VARCHAR(30) NOT
NULL, first_name VARCHAR(30) NOT NULL, last_name VARCHAR(30) NOT NULL,
phone_number INTEGER NOT NULL UNIQUE, gmail VARCHAR(100) NOT NULL
UNIQUE, amount_earned INTEGER NOT NULL, number_of_orders INTEGER NOT
NULL, account_info INTEGER REFERENCES account_info(account_id), FOREIGN
KEY (account_info) REFERENCES account_info(account_id))"
        )

        cur.execute(
            "CREATE TABLE products_info(product_id INTEGER PRIMARY KEY,
product_name VARCHAR(100) NOT NULL UNIQUE, num_of_product INTEGER NOT
NULL, warehouse_loc VARCHAR(100) NOT NULL,product_price INTEGER NOT
NULL, seller_info INTEGER REFERENCES seller_info(seller_id), FOREIGN
KEY(seller_info) REFERENCES seller_info(seller_id))"
        )

        cur.execute(
            "CREATE TABLE category_info(category_name VARCHAR(100) NOT
NULL PRIMARY KEY, desc_ription VARCHAR(1000), products_info INTEGER
REFERENCES products_info(product_id), FOREIGN KEY(products_info)
REFERENCES products_info(product_id))"
        )

        cur.execute(
            "CREATE TABLE shopping_cart(cart_id INTEGER NOT NULL PRIMARY
KEY, total_price DECIMAL NOT NULL, num_order INTEGER NOT NULL,
account_info INTEGER REFERENCES account_info(account_id),products_info
INTEGER REFERENCES products_info(product_id), FOREIGN KEY(account_info)
REFERENCES account_info(account_id), FOREIGN KEY(products_info)
REFERENCES products_info(product_id))"
        )
        cur.execute(
            "CREATE TABLE order_info(order_id INTEGER PRIMARY KEY,
address VARCHAR(100) NOT NULL,total_price INTEGER NOT NULL,
shipping_date VARCHAR(50), order_date VARCHAR(50), shopping_cart INTEGER
REFERENCES shopping_cart(cart_id), FOREIGN KEY(shopping_cart) REFERENCES
shopping_cart(cart_id))"
        )

        cur.execute(
            "CREATE TABLE billing_info(billing_id INTEGER NOT NULL
PRIMARY KEY, billing_address VARCHAR(100) NOT NULL, credit_card_number
INTEGER NOT NULL, product_name VARCHAR(100) NOT NULL,order_info INTEGER
REFERENCES order_info(order_id), FOREIGN KEY(order_info) REFERENCES
order_info(order_id))"
        )

        cur.execute(
            "CREATE TABLE creditcard_info(credit_card_number INTEGER
PRIMARY KEY, credit_cardcvv INTEGER NOT NULL, billing_info INTEGER
REFERENCES billing_info(billing_id),FOREIGN KEY(billing_info) REFERENCES
billing_info(billing_id))"
        )

        cur.execute(
            "CREATE TABLE cost_info(billing_address VARCHAR(100) NOT
NULL, credit_card_number INTEGER NOT NULL, total_cost INTEGER NOT NULL,
billing_info INTEGER REFERENCES billing_info(billing_id), PRIMARY
KEY(billing_address, credit_card_number), FOREIGN KEY(billing_info)
REFERENCES billing_info(billing_id))"
        )

        cur.execute(
            "CREATE TABLE product_type_info(product_name VARCHAR(100)
NOT NULL PRIMARY KEY, product_type VARCHAR(100) NOT NULL, products_info
INTEGER REFERENCES products_info(product_id), FOREIGN KEY(products_info)
REFERENCES products_info(product_id), billing_info INTEGER REFERENCES
billing_info(billing_id), FOREIGN KEY(billing_info) REFERENCES
billing_info(billing_id))"
        )
        cur.execute(
            "CREATE TABLE shipping_info(tracking_number INTEGER NOT NULL
PRIMARY KEY, shipping_rates INTEGER NOT NULL, shipping_address
VARCHAR(100) NOT NULL, order_info INTEGER REFERENCES
order_info(order_id), FOREIGN KEY(order_info) REFERENCES
order_info(order_id))"
        )

        print("Table Created successfully")

    except cx_Oracle.DatabaseError as e:
        print("There is a problem with Oracle", e)

    finally:
        if con:
            cur.close()
            con.close()

    print("Execution Complete")


def DropTables():
    dsnStr = cx_Oracle.makedsn("oracle.scs.ryerson.ca", "1521",
sid="orcl")

    con = None
    try:
        con = cx_Oracle.connect(
            user="athanara", password="07087195", dsn=dsnStr)

        cur = con.cursor()

        cur.execute(
            "DROP TABLE account_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE seller_info CASCADE CONSTRAINTS"
        )
        cur.execute(
            "DROP TABLE products_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE category_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE shopping_cart CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE order_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE billing_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE creditcard_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE cost_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE product_type_info CASCADE CONSTRAINTS"
        )

        cur.execute(
            "DROP TABLE shipping_info CASCADE CONSTRAINTS"
        )

        print("Table Deleted successfully")

    except cx_Oracle.DatabaseError as e:
        print("There is a problem with Oracle", e)

    finally:
        if con:
            cur.close()
            con.close()

    print("Execution Complete")


def PopTables():
    dsnStr = cx_Oracle.makedsn("oracle.scs.ryerson.ca", "1521",
sid="orcl")

    con = None
    try:
        con = cx_Oracle.connect(
            user="athanara", password="07087195", dsn=dsnStr)

        cur = con.cursor()

        cur.execute(
            "INSERT INTO account_info(account_id, user_name, pass_word,
first_name, last_name, phone_number, gmail)VALUES(4017672276, 'HashimF',
'Liberal1999!', 'Hashim','Farooq', 6477175780,
'hashim.farooq@ryerson.ca')"
        )

        cur.execute(
            "INSERT INTO account_info(account_id, user_name, pass_word,
first_name, last_name, phone_number, gmail)VALUES(1428670525,
'AthavanT', 'Hello123', 'Athavan','Thanaraj', 4162907967,
'athavan.thanaraj@ryerson.ca')"
        )

        cur.execute(
            "INSERT INTO account_info(account_id, user_name, pass_word,
first_name, last_name, phone_number, gmail)VALUES(9228569117, 'UmairT',
'Password23', 'Umair','Tariq', 4169914339, 'umair.tariq@ryerson.ca')"
        )
        cur.execute(
            "INSERT INTO seller_info(seller_id, user_name, pass_word,
first_name, last_name, phone_number, gmail, amount_earned,
number_of_orders, account_info)VALUES(3452134532, 'KyleL',
'NorthPhil@12', 'Kyle', 'Lowe',4169922339,'KyleLowe@gmail.ca', 189, 2,
4017672276)"
        )

        cur.execute(
            "INSERT INTO seller_info(seller_id, user_name, pass_word,
first_name, last_name, phone_number, gmail, amount_earned,
number_of_orders, account_info)VALUES(5673247853, 'KevinD', 'Snake!123',
'Kevin', 'Devin',4161921339, 'KyleDevin@gmail.ca', 289.99, 1,
1428670525)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(4354902852, 'EVGA GeForce RTX 3080 Ti FTW3 Ultra
Gaming 12GB GDDR6X Video Card', 1, 'Milton', 1299.99, 3452134532)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(0973076309, 'Corsair iCUE H150i Elite LCD Display
360mm Liquid CPU Cooling System', 2, 'Scarborough', 349.99, 3452134532)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(5689023852, 'Corsair RM850X 850-Watt ATX Modular
Power Supply',1, 'Mississauga', 179.99, 3452134532)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(7960687609, 'Corsair Vengeance RGB Pro 32GB (2 x
16GB) DDR4 3600MHz Desktop Memory',1, 'Mississauga', 149.99,
5673247853)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(1839457439, 'Asus TUF GAMING X570-PLUS (WI-FI) ATX
AM4 Motherboard',1, 'Mississauga', 269.99, 5673247853)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(5010501190, 'Corsair 4000D Airflow Mid-Tower ATX
Computer Case',1, 'Brampton', 114.99, 3452134532)"
        )

        cur.execute(
            "INSERT INTO products_info(product_id, product_name,
num_of_product, warehouse_loc, product_price,
seller_info)VALUES(3345098305, 'Samsung 970 EVO Plus 1TB M.2 NVMe
Internal Solid State Drive',1, 'Brampton', 129.99, 3452134532)"
        )

        cur.execute(
            "INSERT INTO shopping_cart(cart_id, total_price, num_order,
account_info, products_info)VALUES(4276556959, 349.98, 1, 4017672276,
7960687609)"
        )

        cur.execute(
            "INSERT INTO shopping_cart(cart_id, total_price, num_order,
account_info, products_info)VALUES(2408975991, 129.99, 1, 9228569117,
3345098305)"
        )
        cur.execute(
            "INSERT INTO shopping_cart(cart_id, total_price, num_order,
account_info, products_info)VALUES(9655521094, 189.78, 1, 1428670525,
1839457439)"
        )

        cur.execute(
            "INSERT INTO order_info(order_id, address, total_price,
shipping_date, order_date, shopping_cart)VALUES(9703060707, '3292
Hickman Street, Toronto ON, Canada, M2H39A',349.98, '09/12/2023',
'09/01/2023', 4276556959)"
        )

        cur.execute(
            "INSERT INTO order_info(order_id, address, total_price,
shipping_date, order_date, shopping_cart)VALUES(4624186017, '4510
Fleming Street, Toronto ON, Canada, N4B 9K5',129.99, '09/20/2023',
'09/25/2023', 2408975991)"
        )

        cur.execute(
            "INSERT INTO order_info(order_id, address, total_price,
shipping_date, order_date, shopping_cart)VALUES(0995020068, '1247
Borough Drive, Toronto ON, Canada, M1P 4W2', 189.78, '09/30/2023',
'09/01/2023', 9655521094)"
        )

        cur.execute(
            "INSERT INTO billing_info(billing_id, billing_address,
credit_card_number, product_name, order_info)VALUES(21243235, '3292
Hickman Street, Toronto ON Canada M2H39A', 4916744267919114,'EVGA
GeForce RTX 3080 Ti FTW3 Ultra Gaming 12GB GDDR6X Video Card',
9703060707)"
        )

        cur.execute(
            "INSERT INTO billing_info(billing_id, billing_address,
credit_card_number, product_name, order_info)VALUES(12245235, '1247
Borough Drive, Toronto ON Canada, M1P4W2', 4556023406839979,'Corsair
4000D Airflow Mid-Tower ATX Computer Case', 4624186017)"
```

```
        )

        cur.execute(
            "INSERT INTO billing_info(billing_id, billing_address, 
credit_card_number,  product_name, order_info)VALUES(54902654, '4510 
Fleming Street, Toronto ON, Canada, N4B 9K5', 9696896832524524, 'Corsair 
Vengeance RGB Pro 32GB (2 x 16GB) DDR4 3600MHz Desktop Memory', 
0995020068)"
        )

        cur.execute(
            "INSERT INTO shipping_info(tracking_number, shipping_rates, 
shipping_address, order_info)VALUES(4926195528, 1.40,'3292 Hickman 
Street, Toronto ON, Canada, M2H39A', 9703060707)"
        )

        cur.execute(
            "INSERT INTO shipping_info(tracking_number, shipping_rates, 
shipping_address, order_info)VALUES(6317623588, 1.49,'4510 Fleming 
Street, Toronto ON, Canada, N4B 9K5', 4624186017)"
        )

        cur.execute(
            "INSERT INTO shipping_info(tracking_number, shipping_rates, 
shipping_address, order_info)VALUES(3598347189, 1.40, '1247 Borough 
Drive, Toronto ON, Canada, M1P 4W2', 0995020068)"
        )

        con.commit()
        print("Tables Populated successfully")

    except cx_Oracle.DatabaseError as e:
        print("There is a problem with Oracle", e)

    finally:
        if con:
            cur.close()
            con.close()

    print("Execution Complete")
```

```
def QueryTable():
    dsnStr = cx_Oracle.makedsn("oracle.scs.ryerson.ca", "1521",
sid="orcl")

    con = None
    try:
        con = cx_Oracle.connect(
            user="athanara", password="07087195", dsn=dsnStr)

        cur = con.cursor()

        cur.execute(
            "SELECT * from account_info"
        )
        X = cur.fetchall()
        print(X)

        cur.execute(
            "SELECT * FROM shopping_cart ORDER BY total_price ASC"
        )
        Y = cur.fetchall()
        print(Y)

        print("Queries successfull")

    except cx_Oracle.DatabaseError as e:
        print("There is a problem with Oracle", e)

    finally:
        if con:
            cur.close()
            con.close()

    print("Execution Complete")


MainMenu()
```

Normalized tables:

```
CREATE TABLE products_info(product_id INTEGER PRIMARY KEY, product_name
VARCHAR(100) NOT NULL UNIQUE, num_of_product INTEGER NOT NULL,
warehouse_loc VARCHAR(100) NOT NULL,product_price INTEGER NOT NULL,
seller_info INTEGER REFERENCES seller_info(seller_id), FOREIGN
KEY(seller_info) REFERENCES seller_info(seller_id))
```

```
CREATE TABLE billing_info(billing_id INTEGER NOT NULL PRIMARY KEY,
billing_address VARCHAR(100) NOT NULL, credit_card_number INTEGER NOT
NULL, product_name VARCHAR(100) NOT NULL,order_info INTEGER REFERENCES
order_info(order_id), FOREIGN KEY(order_info) REFERENCES
order_info(order_id))
```

```
CREATE TABLE creditcard_info(credit_card_number INTEGER PRIMARY KEY,
credit_cardcvv INTEGER NOT NULL, billing_info INTEGER REFERENCES
billing_info(billing_id),FOREIGN KEY(billing_info) REFERENCES
billing_info(billing_id))
```

```
CREATE TABLE cost_info(billing_address VARCHAR(100) NOT NULL,
credit_card_number INTEGER NOT NULL, total_cost INTEGER NOT NULL,
billing_info INTEGER REFERENCES billing_info(billing_id), PRIMARY
KEY(billing_address, credit_card_number), FOREIGN KEY(billing_info)
REFERENCES billing_info(billing_id))
```

```
CREATE TABLE product_type_info(product_name VARCHAR(100) NOT NULL
PRIMARY KEY, product_type VARCHAR(100) NOT NULL, products_info INTEGER
REFERENCES products_info(product_id), FOREIGN KEY(products_info)
REFERENCES products_info(product_id), billing_info INTEGER REFERENCES
billing_info(billing_id), FOREIGN KEY(billing_info) REFERENCES
billing_info(billing_id))
```