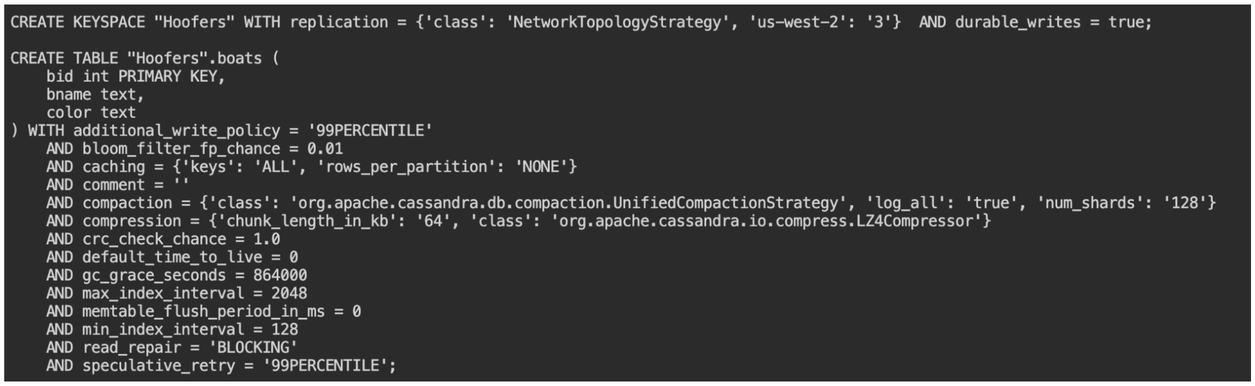
Last Name:Jengirapas First Name:Natcha Student ID:85939811

1.

a) CQL Query:

```
DESC "Hoofers";
```

b) Result:



```
CREATE KEYSPACE "Hoofers" WITH replication = {'class': 'NetworkTopologyStrategy', 'us-west-2': '3'}  AND durable_writes = true;

CREATE TABLE "Hoofers".boats (
    bid int PRIMARY KEY,
    bname text,
    color text
) WITH additional_write_policy = '99PERCENTILE'
    AND bloom_filter_fp_chance = 0.01
    AND caching = {'keys': 'ALL', 'rows_per_partition': 'NONE'}
    AND comment = ''
    AND compaction = {'class': 'org.apache.cassandra.db.compaction.UnifiedCompactionStrategy', 'log_all': 'true', 'num_shards': '128'}
    AND compression = {'chunk_length_in_kb': '64', 'class': 'org.apache.cassandra.io.compress.LZ4Compressor'}
    AND crc_check_chance = 1.0
    AND default_time_to_live = 0
    AND gc_grace_seconds = 864000
    AND max_index_interval = 2048
    AND memtable_flush_period_in_ms = 0
    AND min_index_interval = 128
    AND read_repair = 'BLOCKING'
    AND speculative_retry = '99PERCENTILE';
```

c) Answers:

- The keyspace maintains 3 copies of the data
- The cloud region is Us-west-2
- Greater than 3

2.

a) CQL CREATE Statements:

```
CREATE TABLE Customers (
      customer_id VARCHAR,
      PRIMARY KEY(customer_id)
);

CREATE TABLE Orders (
      order_id VARCHAR,
      total_price DECIMAL,
      time_placed TIMESTAMP,
      pickup_time TIMESTAMP,
      customer_id VARCHAR,
      shopper_id VARCHAR,
      state VARCHAR,
      license_plate VARCHAR,
      store_id VARCHAR,
      time_fulfilled TIMESTAMP,
      PRIMARY KEY (order_id)
);

CREATE TABLE OrderItems (
      item_id VARCHAR,
      order_id VARCHAR,
      product_id VARCHAR,
      qty INT,
      selling_price DECIMAL,
      PRIMARY KEY(item_id, order_id)
);

CREATE TABLE Products (
      product_id VARCHAR,
      category VARCHAR,
      name VARCHAR,
      description VARCHAR,
      list_price DECIMAL,
      PRIMARY KEY (product_id)
);
```

3.

a) PostgreSQL COPY commands:

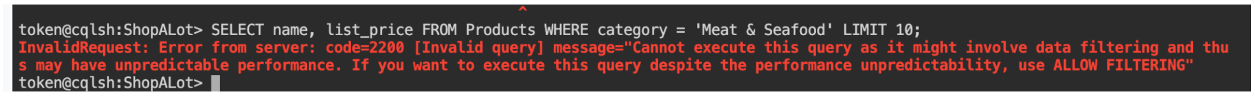
```
copy Customers to '/Users/rew/Desktop/School/CS 122D/HW/HW2/CSV
files/customers.csv' DELIMITER ',' CSV HEADER;
copy Products to '/Users/rew/Desktop/School/CS 122D/HW/HW2/CSV
files/products.csv' DELIMITER ',' CSV HEADER;
copy Orders to '/Users/rew/Desktop/School/CS 122D/HW/HW2/CSV
files/orders.csv' DELIMITER ',' CSV HEADER;
copy Orderitems to '/Users/rew/Desktop/School/CS 122D/HW/HW2/CSV
files/orderitems.csv' DELIMITER ',' CSV HEADER;
```

4.

a) First CQL Query:

```
SELECT name, list_price FROM Products WHERE category = 'Meat & Seafood'
LIMIT 10;
```

b) Result:

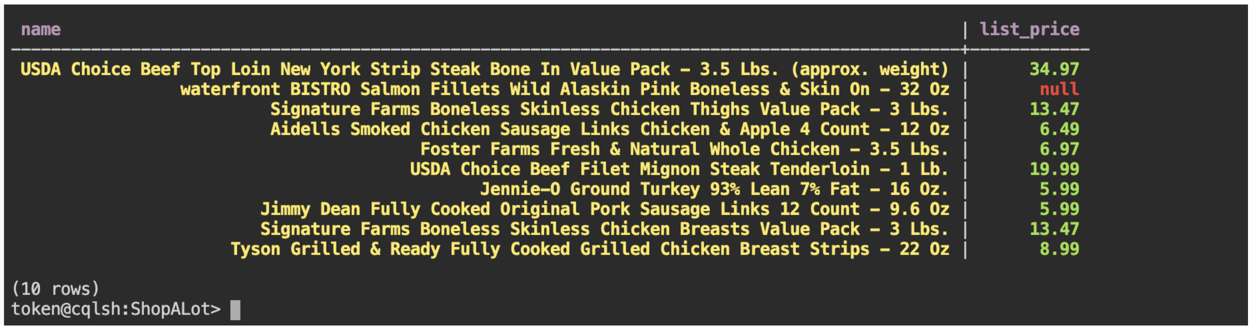


```
token@cqlsh:ShopALot> SELECT name, list_price FROM Products WHERE category = 'Meat & Seafood' LIMIT 10;
InvalidRequest: Error from server: code=2200 [Invalid query] message="Cannot execute this query as it might involve data filtering and thu
s may have unpredictable performance. If you want to execute this query despite the performance unpredictability, use ALLOW FILTERING"
token@cqlsh:ShopALot>
```

c) Modified CQL Query:

```
SELECT name, list_price FROM Products WHERE category = 'Meat & Seafood'
LIMIT 10 ALLOW FILTERING;
```

d) Result:



```
 name                                                                      | list_price
---------------------------------------------------------------------------+------------
 USDA Choice Beef Top Loin New York Strip Steak Bone In Value Pack - 3.5 Lbs. (approx. weight) |      34.97
           waterfront BISTRO Salmon Fillets Wild Alaskin Pink Boneless & Skin On - 32 Oz |       null
                    Signature Farms Boneless Skinless Chicken Thighs Value Pack - 3 Lbs. |      13.47
                    Aidells Smoked Chicken Sausage Links Chicken & Apple 4 Count - 12 Oz |       6.49
                                   Foster Farms Fresh & Natural Whole Chicken - 3.5 Lbs. |       6.97
                                  USDA Choice Beef Filet Mignon Steak Tenderloin - 1 Lb. |      19.99
                                         Jennie-O Ground Turkey 93% Lean 7% Fat - 16 Oz. |       5.99
                  Jimmy Dean Fully Cooked Original Pork Sausage Links 12 Count - 9.6 Oz |       5.99
                   Signature Farms Boneless Skinless Chicken Breasts Value Pack - 3 Lbs. |      13.47
                Tyson Grilled & Ready Fully Cooked Grilled Chicken Breast Strips - 22 Oz |       8.99

(10 rows)
token@cqlsh:ShopALot>
```

5.

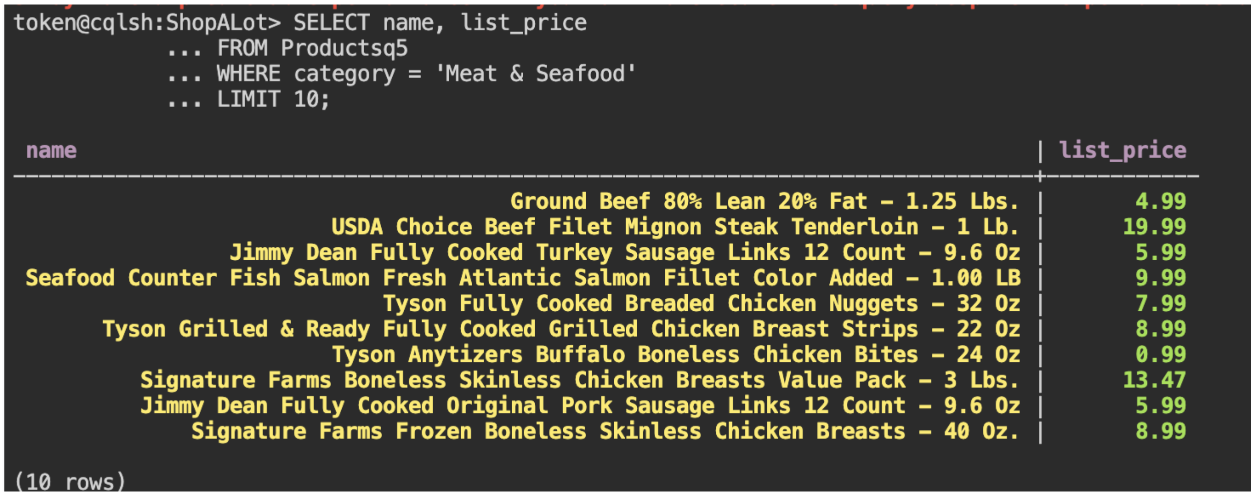
a) CQL Create Statement:

```
CREATE TABLE Productsq5 (
      product_id VARCHAR,
      category VARCHAR,
      name VARCHAR,
      description VARCHAR,
      list_price DECIMAL,
      PRIMARY KEY (category, product_id)
);
```

b) CQL Query:

```
SELECT name, list_price
FROM Productsq5
WHERE category = 'Meat & Seafood'
LIMIT 10;
```

c) Result:



```
token@cqlsh:ShopALot> SELECT name, list_price
            ... FROM Productsq5
            ... WHERE category = 'Meat & Seafood'
            ... LIMIT 10;

 name                                                                    | list_price
-------------------------------------------------------------------------+------------
                       Ground Beef 80% Lean 20% Fat - 1.25 Lbs. |       4.99
             USDA Choice Beef Filet Mignon Steak Tenderloin - 1 Lb. |      19.99
       Jimmy Dean Fully Cooked Turkey Sausage Links 12 Count - 9.6 Oz |       5.99
 Seafood Counter Fish Salmon Fresh Atlantic Salmon Fillet Color Added - 1.00 LB |       9.99
                Tyson Fully Cooked Breaded Chicken Nuggets - 32 Oz |       7.99
      Tyson Grilled & Ready Fully Cooked Grilled Chicken Breast Strips - 22 Oz |       8.99
                Tyson Anytizers Buffalo Boneless Chicken Bites - 24 Oz |       0.99
         Signature Farms Boneless Skinless Chicken Breasts Value Pack - 3 Lbs. |      13.47
         Jimmy Dean Fully Cooked Original Pork Sausage Links 12 Count - 9.6 Oz |       5.99
              Signature Farms Frozen Boneless Skinless Chicken Breasts - 40 Oz. |       8.99

(10 rows)
```

d) Explanation:

Partition keys allow Cassandra to use hashing to find where that key-values would be. Having the category as a partitioning key, Cassandra knows which node it has to retrieve data from, which it is allowed.

If I only included the category as a primary key, the table will only have 16 rows representing every 16 categories since the partition key is required to be unique. However, if I included the category as partition key and product_id as clustering key, the table will contain all the data such that each category will have all the sorted product_id in ascending order of that category.

6.

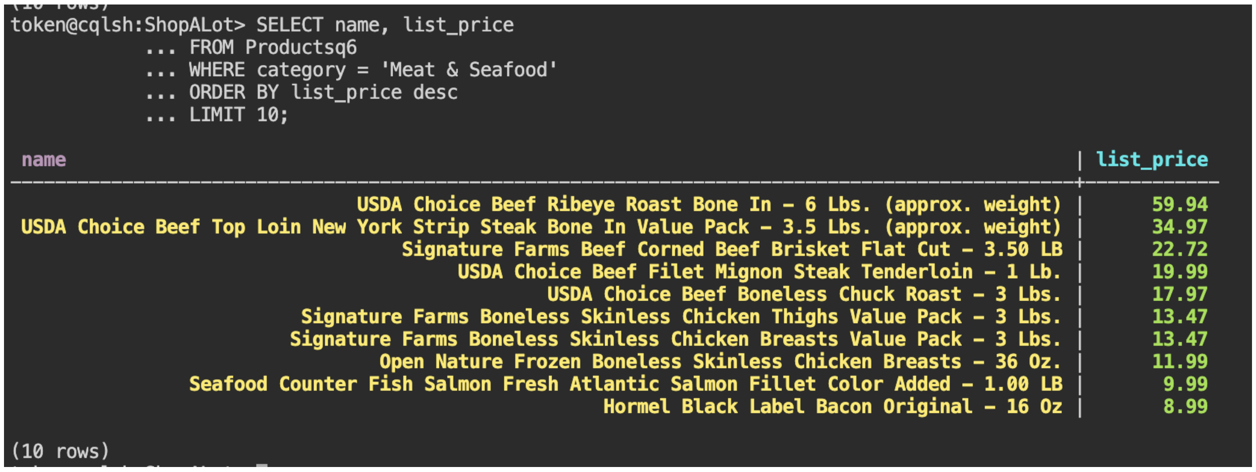
a) CQL Query:

```
SELECT name, list_price
FROM Productsq6
WHERE category = 'Meat & Seafood'
ORDER BY list_price desc
LIMIT 10;
```

b) CQL CREATE Statement:

```
CREATE TABLE Productsq6 (
      product_id VARCHAR,
      category VARCHAR,
      name VARCHAR,
      description VARCHAR,
      list_price DECIMAL,
      PRIMARY KEY (category, list_price, product_id)
);
```

c) Results:



```
token@cqlsh:ShopALot> SELECT name, list_price
                  ... FROM Productsq6
                  ... WHERE category = 'Meat & Seafood'
                  ... ORDER BY list_price desc
                  ... LIMIT 10;

 name                                                                                           | list_price
------------------------------------------------------------------------------------------------+------------
                                    USDA Choice Beef Ribeye Roast Bone In - 6 Lbs. (approx. weight) |      59.94
 USDA Choice Beef Top Loin New York Strip Steak Bone In Value Pack - 3.5 Lbs. (approx. weight) |      34.97
                                        Signature Farms Beef Corned Beef Brisket Flat Cut - 3.50 LB |      22.72
                                             USDA Choice Beef Filet Mignon Steak Tenderloin - 1 Lb. |      19.99
                                                     USDA Choice Beef Boneless Chuck Roast - 3 Lbs. |      17.97
                              Signature Farms Boneless Skinless Chicken Thighs Value Pack - 3 Lbs. |      13.47
                             Signature Farms Boneless Skinless Chicken Breasts Value Pack - 3 Lbs. |      13.47
                                     Open Nature Frozen Boneless Skinless Chicken Breasts - 36 Oz. |      11.99
                   Seafood Counter Fish Salmon Fresh Atlantic Salmon Fillet Color Added - 1.00 LB |       9.99
                                                        Hormel Black Label Bacon Original - 16 Oz |       8.99

(10 rows)
```

d) Explanation:

Cassandra achieves performance through the use of the clustering keys to order data. Thus, returning ordered rows in a single read. The primary key is similar to the previous question. However, we included list_price as another clustering key before product_id. Hence, Cassandra would order the data by list_price first then product_id.

7.

a) CQL Create Statement:

```
CREATE TABLE Ordersq7a (
      customer_id VARCHAR,
      order_id  VARCHAR,
      total_price DECIMAL,
      PRIMARY KEY(customer_id, total_price, order_id)
) WITH CLUSTERING ORDER BY (total_price DESC, order_id ASC);
```

b) CQL Create Statement:

```
CREATE TABLE Shoppersq7b(
      shopper_id VARCHAR,
      order_id VARCHAR,
      item_id VARCHAR,
      PRIMARY KEY (shopper_id, order_id, item_id));
```

c) CQL Create Statement:

```
CREATE TABLE Ordersq7C(
      order_id VARCHAR,
      item_id VARCHAR,
      name VARCHAR,
      category VARCHAR,
      list_price DECIMAL,
PRIMARY KEY (order_id, list_price, item_id));
```

d) CQL Create Statement:

```
CREATE TABLE Customersq7d(
      customer_id VARCHAR,
      order_id VARCHAR,
      total_price DECIMAL,
      time_placed TIMESTAMP,
      PRIMARY KEY(customer_id, time_placed, order_id));
```

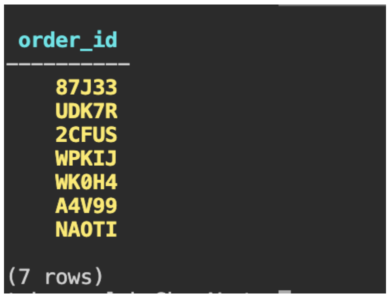
8.

a)

- CQL Query:

```
SELECT order_id FROM Ordersq7a WHERE Customer_id = '24590';
```

- Result:



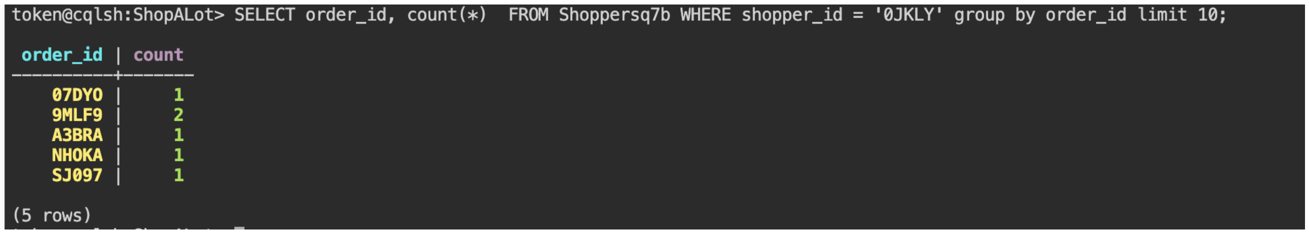
```
 order_id
----------
    87J33
    UDK7R
    2CFUS
    WPKIJ
    WK0H4
    A4V99
    NAOTI

(7 rows)
```

b)

- CQL Query:

```
SELECT order_id, count(*)  FROM Shoppersq7b WHERE shopper_id = '0JKLY'
group by order_id limit 10;
```

- Result:



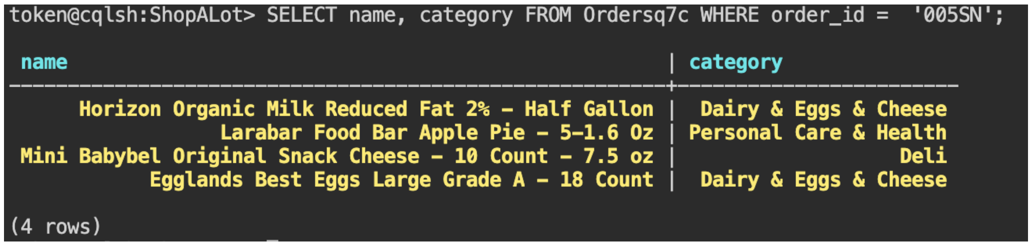
```
token@cqlsh:ShopALot> SELECT order_id, count(*)  FROM Shoppersq7b WHERE shopper_id = '0JKLY' group by order_id limit 10;

 order_id | count
----------+-------
    07DYO |     1
    9MLF9 |     2
    A3BRA |     1
    NHOKA |     1
    SJ097 |     1

(5 rows)
```

c)

- CQL Query:

```
SELECT name, category FROM Ordersq7c WHERE order_id =  '005SN';
```

- Result:



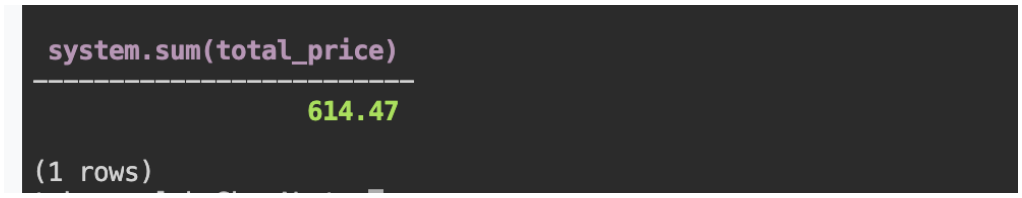
```
token@cqlsh:ShopALot> SELECT name, category FROM Ordersq7c WHERE order_id =  '005SN';

 name                                                   | category
--------------------------------------------------------+-------------------------
      Horizon Organic Milk Reduced Fat 2% - Half Gallon |  Dairy & Eggs & Cheese
                   Larabar Food Bar Apple Pie - 5-1.6 Oz | Personal Care & Health
 Mini Babybel Original Snack Cheese - 10 Count - 7.5 oz |                   Deli
             Egglands Best Eggs Large Grade A - 18 Count |  Dairy & Eggs & Cheese

(4 rows)
```

d)

- CQL Query:

```
SELECT SUM(total_price) FROM Customersq7d WHERE customer_id =  '32976' and
time_placed >= '2020-03-01 00:00:00' and time_placed <= '2020-09-01
00:00:00';
```

- Result:



```
 system.sum(total_price)
-------------------------
                   614.47

(1 rows)
```

9.

a) CQL INSERT statements:

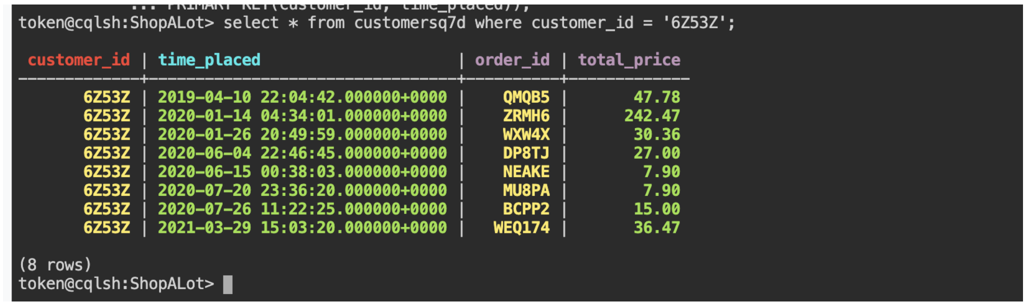
```
-- Tables from Part 2
INSERT INTO orders (order_id, total_price, pickup_time, customer_id,
shopper_id, state, license_plate, store_id, time_fulfilled) VALUES
('12MDAE', 7.14, '2021-04-10T21:01:45.000Z', '24590', 'MQD30', 'CA', 'AKM
554', 'A7ZNF', '2021-04-10T23:20:56.000Z');

INSERT INTO orderItems (item_id, order_id, product_id, qty, selling_price)
VALUES ('9B317', '12MDAE', 'GMGO5', 3, 2.38);

-- Tables from Part 7
INSERT INTO ordersq7a (customer_id, order_id, total_price) VALUES ('24590',
'12MDAE', 7.14);

INSERT INTO shoppersq7b (shopper_id, order_id, item_id) VALUES ('MQD30',
'12MDAE', '9B317');

INSERT INTO ordersq7c (order_id, item_id, name, category, list_price)
VALUES ( '12MDAE', '9B317', 'Green Giant Corn Whole Kernel Sweet - 15.25
Oz', 'Canned Goods & Soups', 1.59);

INSERT INTO customersq7d (customer_id, order_id, total_price, time_placed)
VALUES ('24590', '9B317', 7.14,  '2021-04-10T19:30:24.000Z');
```

10. [Extra Credit]

Python script:

```python
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
from datetime import datetime


# Connecting to the database: code from DataStax Astra
def connect_cassandra():
   cloud_config= {
       'secure_connect_bundle': "/Users/rew/Downloads/secure-connect-cs122d-spring.zip"
   }
   auth_provider = PlainTextAuthProvider('rsvrHPKpQDmYhjDjhBIISHJZ', 
'ENo2M,rKX4ldlPzQv2h1SvhmZex9I1Ft7d+DISg952d18TfRhlinyhSSPuKMvm3ZLMzaXvULKpO0uWGn.LBAwn.BxuU
qh8wOd86eOC8rEUPYQzHAoG.mg8E0WvL97T2h')
   cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
   global session
   session = cluster.connect()

   row = session.execute("select release_version from system.local").one()

   if row:
       print("SUCCESSFULLY CONNECTED!")
   else:
       print("An error occurred.")


# Getting all table names and column names in a keyspace
def get_tables(keyspace):
   tables = dict()
   tempTables = session.execute(f"SELECT * FROM system_schema.tables WHERE keyspace_name = 
'{keyspace}';")

   for table in tempTables:
       tempColumns = session.execute(f"SELECT * FROM system_schema.columns WHERE 
keyspace_name = '{keyspace}' AND table_name = '{table.table_name}';")
       column = {i.column_name: 0 for i in tempColumns}
       tables[table.table_name] = column

   return tables


# Creating an insert command based on the argument dictionary
def create_insert(tables, table_name, argDict):
   text = ""

   for column in tables[table_name]:
       if column in argDict:
           text += f"\"{column}\": \"{argDict[column]}\", "
       elif column in ["name", "category", "list_price"]:
           product_id = argDict["orderItems"][0]["product_id"]
           lookup = session.execute(f"SELECT {column} FROM \"ShopALot\".Products WHERE
product_id = \'{product_id}\'")
           text += f"\"{column}\": \"{lookup.one()[0]}\", "
       elif column in ["item_id", "qty", "selling_price", "product_id"]:
           orderItem = argDict["orderItems"][0][column]
           text += f"\"{column}\": \"{orderItem}\", "

   text = f"INSERT INTO \"ShopALot\".{table_name} json '{{{text[:-2]}}}';"

   insert = session.prepare(text)
   results = session.execute(insert, argDict)


def insert_new_order(tables, infoDict):
   for table in tables:
       # Check if the input table needs to add new order
       if "order_id" in tables[table]:
           create_insert(tables, table, infoDict)


def main():
   keyspace = "ShopALot"
   connect_cassandra()
   new_order = {"order_id": "WEQ174", "total_price": 36.47, "time_placed":
"2021-03-29T15:03:20.000Z",
               "pickup_time": "2021-03-23T17:54:21.000Z", "customer_id": "6Z53Z",
"shopper_id": "MQD30",
               "state": "WV", "license_plate": "0031", "store_id": "ZU9IP",
"time_fulfilled": "2021-03-23T22:43:12.000Z",
               "orderItems": [{"item_id": "PO12C", "qty": "7", "selling_price": 5.21,
"product_id": "GMGO5"}]}

   tables = get_tables(keyspace)
   insert_new_order(tables, new_order)


if __name__ == "__main__":
   main();
```

*Query to show that the new order has been added to the table.



```
        ... PRIMARY KEY(customer_id, time_placed));
token@cqlsh:ShopALot> select * from customersq7d where customer_id = '6Z53Z';

 customer_id | time_placed                     | order_id | total_price
-------------+---------------------------------+----------+-------------
       6Z53Z | 2019-04-10 22:04:42.000000+0000 |    QMQB5 |       47.78
       6Z53Z | 2020-01-14 04:34:01.000000+0000 |    ZRMH6 |      242.47
       6Z53Z | 2020-01-26 20:49:59.000000+0000 |    WXW4X |       30.36
       6Z53Z | 2020-06-04 22:46:45.000000+0000 |    DP8TJ |       27.00
       6Z53Z | 2020-06-15 00:38:03.000000+0000 |    NEAKE |        7.90
       6Z53Z | 2020-07-20 23:36:20.000000+0000 |    MU8PA |        7.90
       6Z53Z | 2020-07-26 11:22:25.000000+0000 |    BCPP2 |       15.00
       6Z53Z | 2021-03-29 15:03:20.000000+0000 |   WEQ174 |       36.47

(8 rows)
token@cqlsh:ShopALot>
```