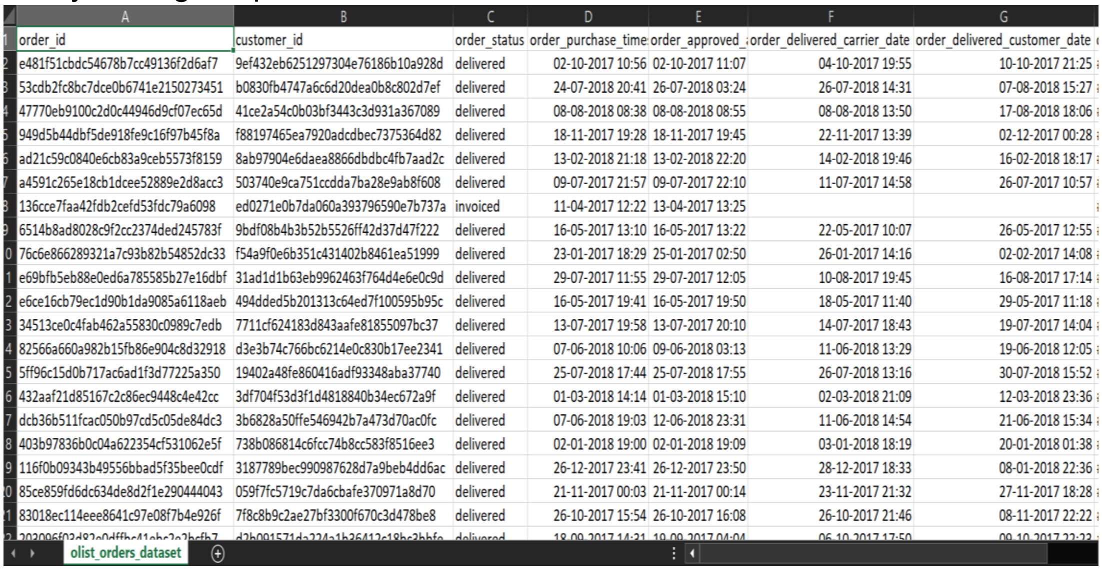
Real Time E-Commerce Data Processing

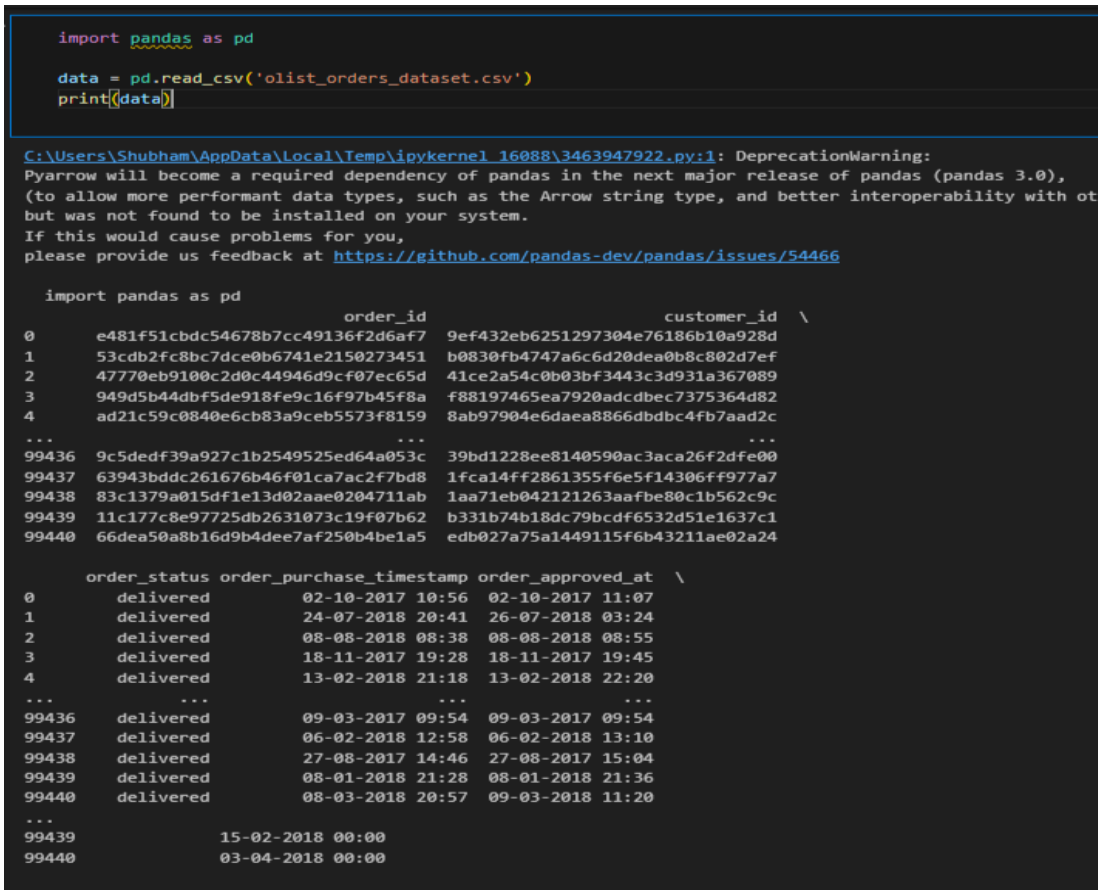
1. created a real-time data pipeline for processing e-commerce data using Apache Kafka and Apache Cassandra. I was ingesting data from a CSV file using a Kafka producer, transforming the data using a Kafka consumer, and finally storing the processed data in a Cassandra table.



	A	B	C	D	E	F	G
1	order_id	customer_id	order_status	order_purchase_time	order_approved_	order_delivered_carrier_date	order_delivered_customer_date
2	e481f51cbdc54678b7cc49136f2d6af7	9ef432eb6251297304e76186b10a928d	delivered	02-10-2017 10:56	02-10-2017 11:07	04-10-2017 19:55	10-10-2017 21:25
3	53cdb2fc8bc7dce0b6741e2150273451	b0830fb4747a6c6d20dea0b8c802d7ef	delivered	24-07-2018 20:41	26-07-2018 03:24	26-07-2018 14:31	07-08-2018 15:27
4	47770eb9100c2d0c44946d9cf07ec65d	41ce2a54c0b03bf3443c3d931a367089	delivered	08-08-2018 08:38	08-08-2018 08:55	08-08-2018 13:50	17-08-2018 18:06
5	949d5b44dbf5de918fe9c16f97b45f8a	f88197465ea7920adcdbec7375364d82	delivered	18-11-2017 19:28	18-11-2017 19:45	22-11-2017 13:39	02-12-2017 00:28
6	ad21c59c0840e6cb83a9ceb5573f8159	8ab97904e6daea8866dbdbc4fb7aad2c	delivered	13-02-2018 21:18	13-02-2018 22:20	14-02-2018 19:46	16-02-2018 18:17
7	a4591c265e18cb1dcee52889e2d8acc3	503740e9ca751ccdda7ba28e9ab8f608	delivered	09-07-2017 21:57	09-07-2017 22:10	11-07-2017 14:58	26-07-2017 10:57
8	136cce7faa42fdb2cefd53fdc79a6098	ed0271e0b7da060a393796590e7b737a	invoiced	11-04-2017 12:22	13-04-2017 13:25		
9	6514b8ad8028c9f2cc2374ded245783f	9bdf08b4b3b52b5526ff42d37d47f222	delivered	16-05-2017 13:10	16-05-2017 13:22	22-05-2017 10:07	26-05-2017 12:55
10	76c6e866289321a7c93b82b54852dc33	f54a9f0e6b351c431402b8461ea51999	delivered	23-01-2017 18:29	25-01-2017 02:50	26-01-2017 14:16	02-02-2017 14:08
11	e69bfb5eb88e0ed6a785585b27e16dbf	31ad1d1b63eb9962463f764d4e6e0c9d	delivered	29-07-2017 11:55	29-07-2017 12:05	10-08-2017 19:45	16-08-2017 17:14
12	e6ce16cb79ec1d90b1da9085a6118aeb	494dded5b201313c64ed7f100595b95c	delivered	16-05-2017 19:41	16-05-2017 19:50	18-05-2017 11:40	29-05-2017 11:18
13	34513ce0c4fab462a55830c0989c7edb	7711cf624183d843aafe81855097bc37	delivered	13-07-2017 19:58	13-07-2017 20:10	14-07-2017 18:43	19-07-2017 14:04
14	82566a660a982b15fb86e904c8d32918	d3e3b74c766bc6214e0c830b17ee2341	delivered	07-06-2018 10:06	09-06-2018 03:13	11-06-2018 13:29	19-06-2018 12:05
15	5ff96c15d0b717ac6ad1f3d77225a350	19402a48fe860416adf93348aba37740	delivered	25-07-2018 17:44	25-07-2018 17:55	26-07-2018 13:16	30-07-2018 15:52
16	432aaf21d85167c2c86ec9448c4e42cc	3df704f53d3f1d4818840b34ec672a9f	delivered	01-03-2018 14:14	01-03-2018 15:10	02-03-2018 21:09	12-03-2018 23:36
17	dcb36b511fcac050b97cd5c05de84dc3	3b6828a50ffe546942b7a473d70ac0fc	delivered	07-06-2018 19:03	12-06-2018 23:31	11-06-2018 14:54	21-06-2018 15:34
18	403b97836b0c04a622354cf531062e5f	738b086814c6fcc74b8cc583f8516ee3	delivered	02-01-2018 19:00	02-01-2018 19:09	03-01-2018 18:19	20-01-2018 01:38
19	116f0b09343b49556bbad5f35bee0cdf	3187789bec990987628d7a9beb4dd6ac	delivered	26-12-2017 23:41	26-12-2017 23:50	28-12-2017 18:33	08-01-2018 22:36
20	85ce859fd6dc634de8d2f1e290444043	059f7fc5719c7da6cbafe370971a8d70	delivered	21-11-2017 00:03	21-11-2017 00:14	23-11-2017 21:32	27-11-2017 18:28
21	83018ec114eee8641c97e08f7b4e926f	7f8c8b9c2ae27bf3300f670c3d478be8	delivered	26-10-2017 15:54	26-10-2017 16:08	26-10-2017 21:46	08-11-2017 22:22

olist_orders_dataset

2. So first loaded the 'olist_orders_dataset.csv' into a pandas dataframe and examined its structure and contents.

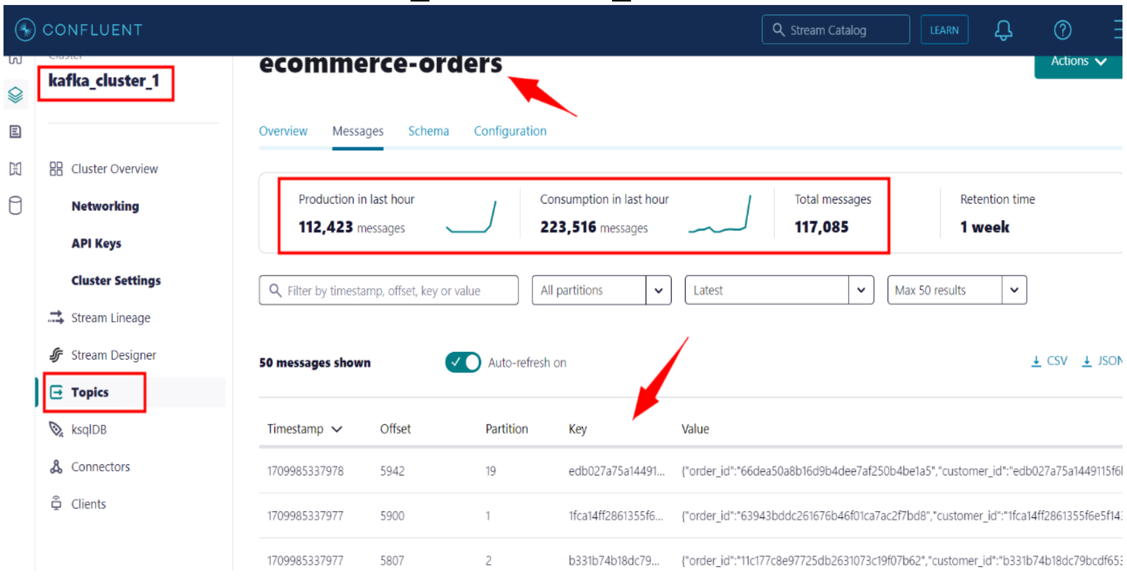


```python
import pandas as pd

data = pd.read_csv('olist_orders_dataset.csv')
print(data)
```

```
C:\Users\Shubham\AppData\Local\Temp\ipykernel_16088\3463947922.py:1: DeprecationWarning:
Pyarrow will become a required dependency of pandas in the next major release of pandas (pandas 3.0),
(to allow more performant data types, such as the Arrow string type, and better interoperability with ot
but was not found to be installed on your system.
If this would cause problems for you,
please provide us feedback at https://github.com/pandas-dev/pandas/issues/54466

  import pandas as pd
                               order_id                       customer_id  \
0      e481f51cbdc54678b7cc49136f2d6af7  9ef432eb6251297304e76186b10a928d
1      53cdb2fc8bc7dce0b6741e2150273451  b0830fb4747a6c6d20dea0b8c802d7ef
2      47770eb9100c2d0c44946d9cf07ec65d  41ce2a54c0b03bf3443c3d931a367089
3      949d5b44dbf5de918fe9c16f97b45f8a  f88197465ea7920adcdbec7375364d82
4      ad21c59c0840e6cb83a9ceb5573f8159  8ab97904e6daea8866dbdbc4fb7aad2c
...                                 ...                               ...
99436  9c5dedf39a927c1b2549525ed64a053c  39bd1228ee8140590ac3aca26f2dfe00
99437  63943bddc261676b46f01ca7ac2f7bd8  1fca14ff2861355f6e5f14306ff977a7
99438  83c1379a015df1e13d02aae0204711ab  1aa71eb042121263aafbe80c1b562c9c
99439  11c177c8e97725db2631073c19f07b62  b331b74b18dc79bcdf6532d51e1637c1
99440  66dea50a8b16d9b4dee7af250b4be1a5  edb027a75a1449115f6b43211ae02a24

      order_status order_purchase_timestamp order_approved_at  \
0        delivered         02-10-2017 10:56  02-10-2017 11:07
1        delivered         24-07-2018 20:41  26-07-2018 03:24
2        delivered         08-08-2018 08:38  08-08-2018 08:55
3        delivered         18-11-2017 19:28  18-11-2017 19:45
4        delivered         13-02-2018 21:18  13-02-2018 22:20
...            ...                      ...               ...
99436    delivered         09-03-2017 09:54  09-03-2017 09:54
99437    delivered         06-02-2018 12:58  06-02-2018 13:10
99438    delivered         27-08-2017 14:46  27-08-2017 15:04
99439    delivered         08-01-2018 21:28  08-01-2018 21:36
99440    delivered         08-03-2018 20:57  09-03-2018 11:20

...
99439                15-02-2018 00:00
99440                03-04-2018 00:00
```

3. configured Confluent Kafka & created a Kafka topic, named 'ecommerce-orders', to hold the e-commerce data.

4. developed a kafka idempotence producer(idempotence to prevent duplicates) in python that reads the data from the pandas dataframe and publishes it to the 'ecommerce-orders' Kafka topic. The key for each message was a combination of the 'customer_id' and 'order_id' fields from the dataset.



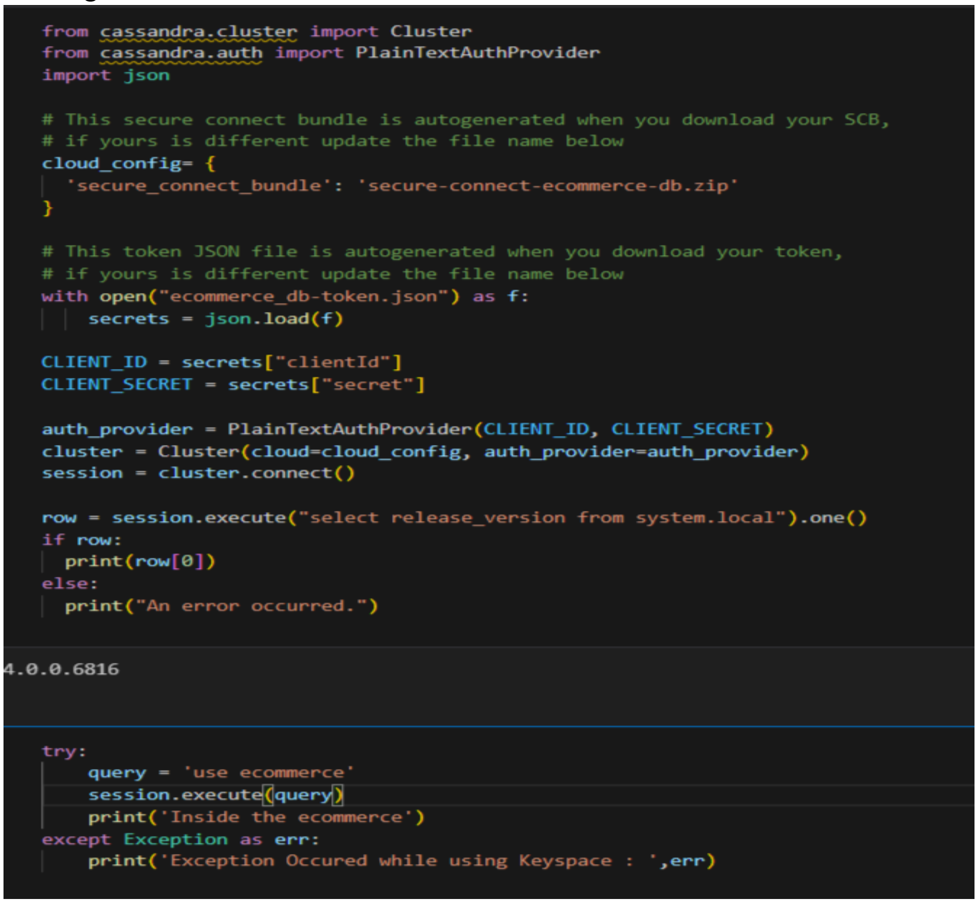
ecommerce-orders

Production in last hour: 112,423 messages
Consumption in last hour: 223,516 messages
Total messages: 117,085
Retention time: 1 week

50 messages shown

Timestamp	Offset	Partition	Key	Value
1709985337978	5942	19	edb027a75a14491...	{"order_id":"66dea50a8b16d9b4dee7af250b4be1a5","customer_id":"edb027a75a1449115f6
1709985337977	5900	1	1fca14ff2861355f6...	{"order_id":"63943bddc261676b46f01ca7ac2f7bd8","customer_id":"1fca14ff2861355f6e5f14
1709985337977	5807	2	b331b74b18dc79...	{"order_id":"11c177c8e97725db2631073c19f07b62","customer_id":"b331b74b18dc79bcdf65

5. Set up Datastax Cassandra & created a keyspace, named 'ecommerce', for storing the e-commerce data.

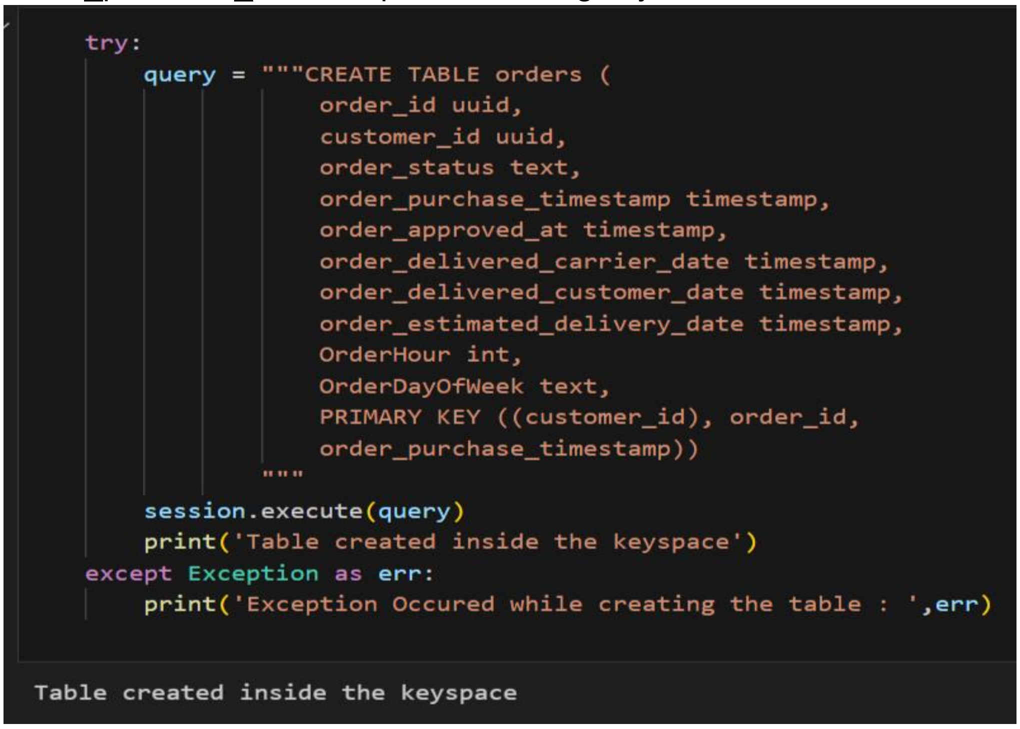


```python
from cassandra.cluster import Cluster
from cassandra.auth import PlainTextAuthProvider
import json

# This secure connect bundle is autogenerated when you download your SCB,
# if yours is different update the file name below
cloud_config= {
  'secure_connect_bundle': 'secure-connect-ecommerce-db.zip'
}

# This token JSON file is autogenerated when you download your token,
# if yours is different update the file name below
with open("ecommerce_db-token.json") as f:
    secrets = json.load(f)

CLIENT_ID = secrets["clientId"]
CLIENT_SECRET = secrets["secret"]

auth_provider = PlainTextAuthProvider(CLIENT_ID, CLIENT_SECRET)
cluster = Cluster(cloud=cloud_config, auth_provider=auth_provider)
session = cluster.connect()

row = session.execute("select release_version from system.local").one()
if row:
  print(row[0])
else:
  print("An error occurred.")
```

```
4.0.0.6816
```

```python
try:
    query = 'use ecommerce'
    session.execute(query)
    print('Inside the ecommerce')
except Exception as err:
    print('Exception Occured while using Keyspace : ',err)
```

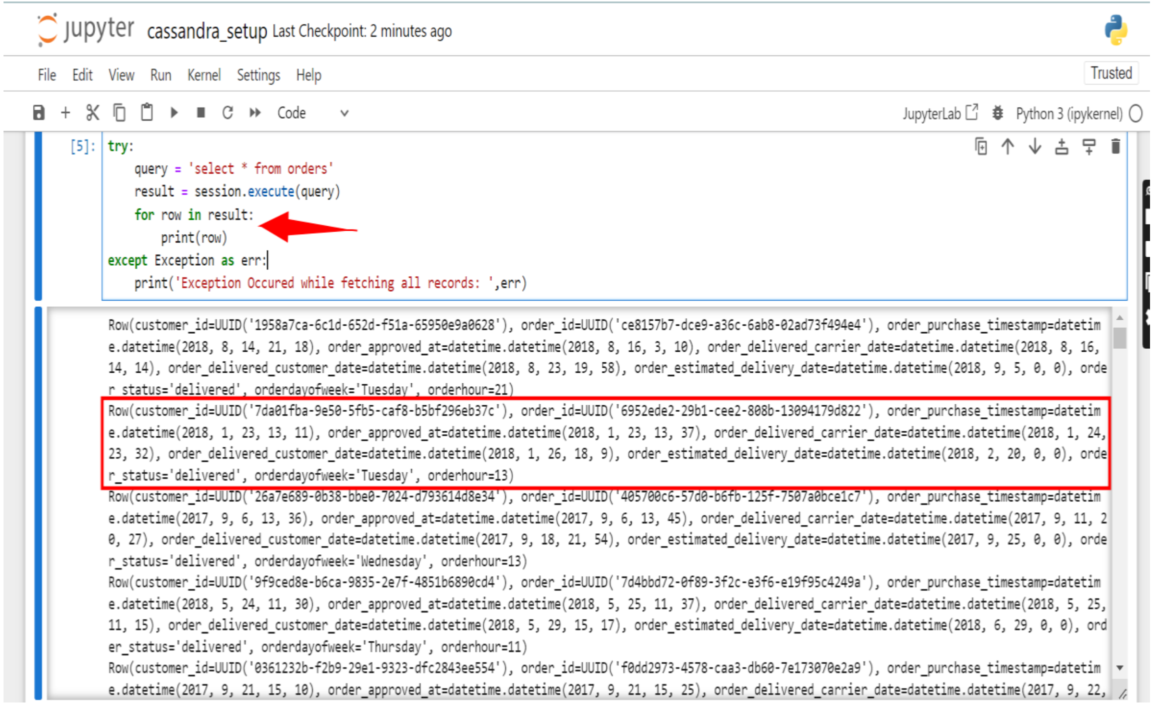
6. Then designed a table, named 'orders', within the 'ecommerce' keyspace. This table reflected the schema of the incoming data and included additional columns for the derived features: 'OrderHour' and 'OrderDayOfWeek'. The data model had 'customer_id' as the partition key and 'order_id' and 'order_purchase_timestamp' as clustering keys.



```
try:
    query = """CREATE TABLE orders (
            order_id uuid,
            customer_id uuid,
            order_status text,
            order_purchase_timestamp timestamp,
            order_approved_at timestamp,
            order_delivered_carrier_date timestamp,
            order_delivered_customer_date timestamp,
            order_estimated_delivery_date timestamp,
            OrderHour int,
            OrderDayOfWeek text,
            PRIMARY KEY ((customer_id), order_id,
            order_purchase_timestamp))
        """
    session.execute(query)
    print('Table created inside the keyspace')
except Exception as err:
    print('Exception Occured while creating the table : ',err)
```

```
Table created inside the keyspace
```

7. Developed a Kafka consumer (with having a consumer group) in Python that subscribes to the 'ecommerce-orders' topic. The consumer derived two new columns 'PurchaseHour'/'OrderHour' and 'PurchaseDayOfWeek'/'OrderDayOfWeek', then ingesting transformed data into the 'orders' table in Cassandra.

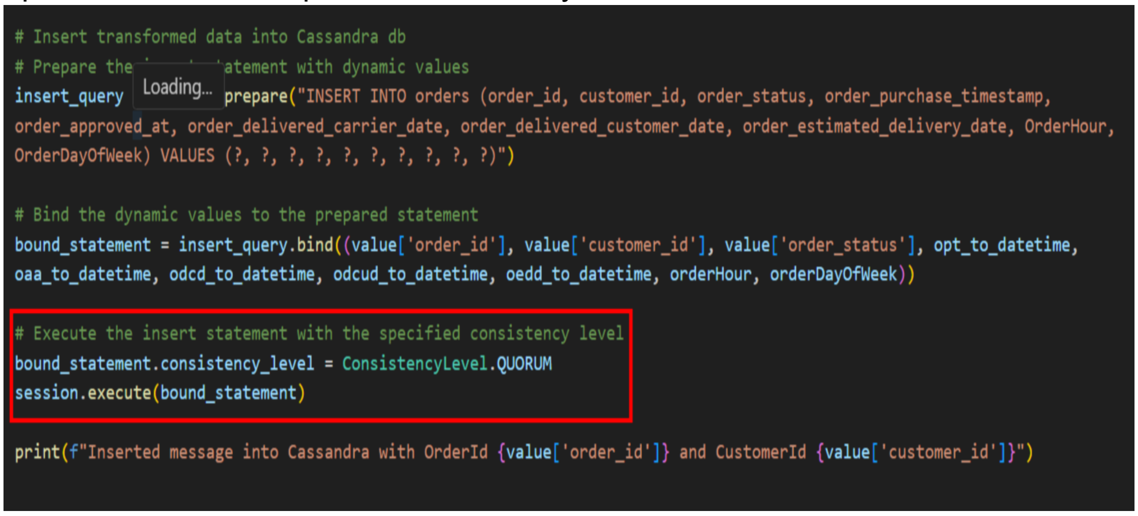


jupyter cassandra_setup Last Checkpoint: 2 minutes ago
File Edit View Run Kernel Settings Help
Trusted
Code
JupyterLab
Python 3 (ipykernel)

```
[5]: try:
         query = 'select * from orders'
         result = session.execute(query)
         for row in result:
             print(row)
     except Exception as err:
         print('Exception Occured while fetching all records: ',err)
```

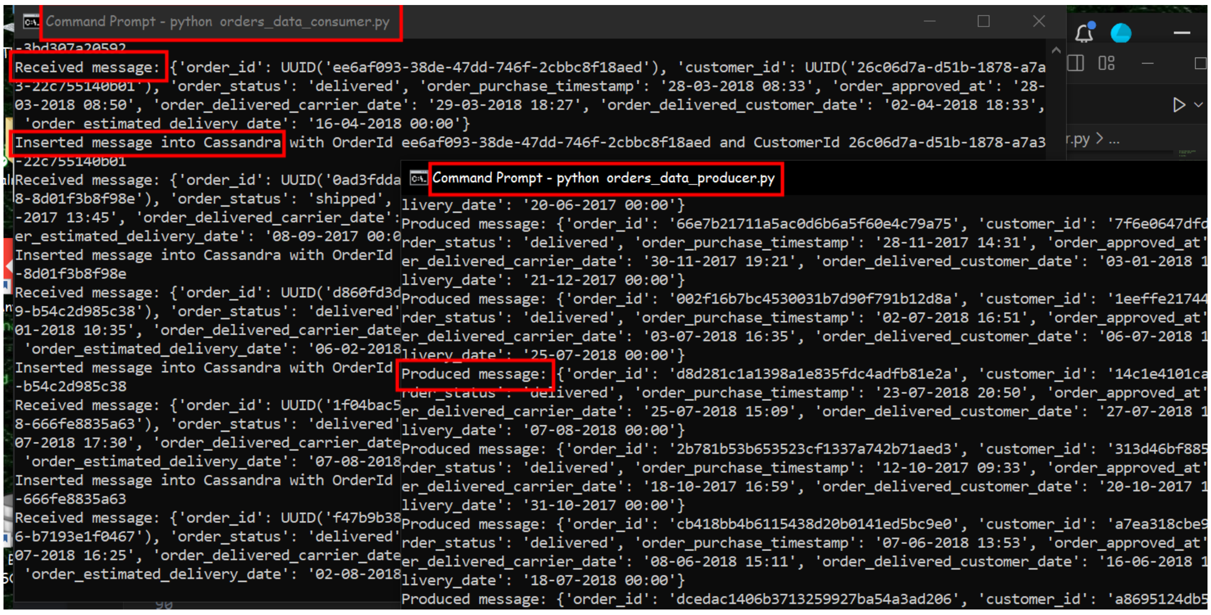
```
Row(customer_id=UUID('1958a7ca-6c1d-652d-f51a-65950e9a0628'), order_id=UUID('ce8157b7-dce9-a36c-6ab8-02ad73f494e4'), order_purchase_timestamp=datetim
e.datetime(2018, 8, 14, 21, 18), order_approved_at=datetime.datetime(2018, 8, 16, 3, 10), order_delivered_carrier_date=datetime.datetime(2018, 8, 16,
14, 14), order_delivered_customer_date=datetime.datetime(2018, 8, 23, 19, 58), order_estimated_delivery_date=datetime.datetime(2018, 9, 5, 0, 0), orde
r_status='delivered', orderdayofweek='Tuesday', orderhour=21)
Row(customer_id=UUID('7da01fba-9e50-5fb5-caf8-b5bf296eb37c'), order_id=UUID('6952ede2-29b1-cee2-808b-13094179d822'), order_purchase_timestamp=datetim
e.datetime(2018, 1, 23, 13, 11), order_approved_at=datetime.datetime(2018, 1, 23, 13, 37), order_delivered_carrier_date=datetime.datetime(2018, 1, 24,
23, 32), order_delivered_customer_date=datetime.datetime(2018, 1, 26, 18, 9), order_estimated_delivery_date=datetime.datetime(2018, 2, 20, 0, 0), orde
r_status='delivered', orderdayofweek='Tuesday', orderhour=13)
Row(customer_id=UUID('26a7e689-0b38-bbe0-7024-d793614d8e34'), order_id=UUID('405700c6-57d0-b6fb-125f-7507a0bce1c7'), order_purchase_timestamp=datetim
e.datetime(2017, 9, 6, 13, 36), order_approved_at=datetime.datetime(2017, 9, 6, 13, 45), order_delivered_carrier_date=datetime.datetime(2017, 9, 11, 2
0, 27), order_delivered_customer_date=datetime.datetime(2017, 9, 18, 21, 54), order_estimated_delivery_date=datetime.datetime(2017, 9, 25, 0, 0), orde
r_status='delivered', orderdayofweek='Wednesday', orderhour=13)
Row(customer_id=UUID('9f9ced8e-b6ca-9835-2e7f-4851b6890cd4'), order_id=UUID('7d4bbd72-0f89-3f2c-e3f6-e19f95c4249a'), order_purchase_timestamp=datetim
e.datetime(2018, 5, 24, 11, 30), order_approved_at=datetime.datetime(2018, 5, 25, 11, 37), order_delivered_carrier_date=datetime.datetime(2018, 5, 25,
11, 15), order_delivered_customer_date=datetime.datetime(2018, 5, 29, 15, 17), order_estimated_delivery_date=datetime.datetime(2018, 6, 29, 0, 0), ord
er_status='delivered', orderdayofweek='Thursday', orderhour=11)
Row(customer_id=UUID('0361232b-f2b9-29e1-9323-dfc2843ee554'), order_id=UUID('f0dd2973-4578-caa3-db60-7e173070e2a9'), order_purchase_timestamp=datetim
e.datetime(2017, 9, 21, 15, 10), order_approved_at=datetime.datetime(2017, 9, 21, 15, 25), order_delivered_carrier_date=datetime.datetime(2017, 9, 22,
```

8. While inserting data into the Cassandra 'orders' table, ensured that the write operations maintain quorum consistency.



```
# Insert transformed data into Cassandra db
# Prepare the          atement with dynamic values
insert_query  Loading...  prepare("INSERT INTO orders (order_id, customer_id, order_status, order_purchase_timestamp,
order_approved_at, order_delivered_carrier_date, order_delivered_customer_date, order_estimated_delivery_date, OrderHour,
OrderDayOfWeek) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)")

# Bind the dynamic values to the prepared statement
bound_statement = insert_query.bind((value['order_id'], value['customer_id'], value['order_status'], opt_to_datetime,
oaa_to_datetime, odcd_to_datetime, odcud_to_datetime, oedd_to_datetime, orderHour, orderDayOfWeek))

# Execute the insert statement with the specified consistency level
bound_statement.consistency_level = ConsistencyLevel.QUORUM
session.execute(bound_statement)

print(f"Inserted message into Cassandra with OrderId {value['order_id']} and CustomerId {value['customer_id']}")
```

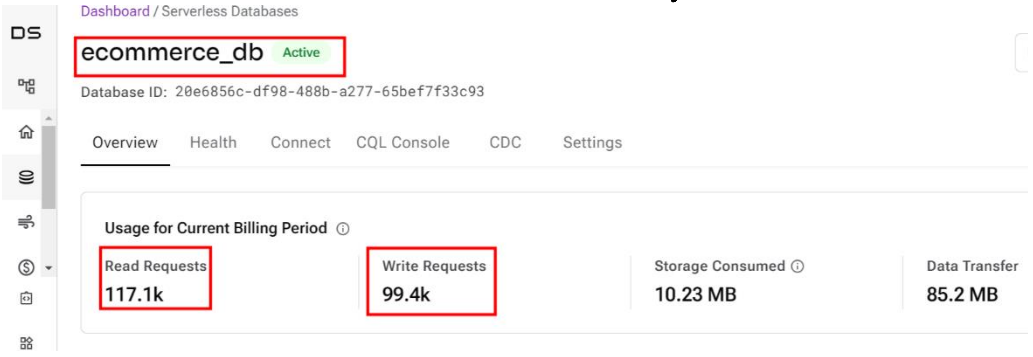
9. Tested the data pipeline end-to-end. Ran Kafka producer to ingest the data, then executed the Kafka consumer to process the data and inserted it into the Cassandra table.



```
Command Prompt - python  orders_data_consumer.py
-3bd307a20592
Received message: {'order_id': UUID('ee6af093-38de-47dd-746f-2cbbc8f18aed'), 'customer_id': UUID('26c06d7a-d51b-1878-a7a
3-22c755140b01'), 'order_status': 'delivered', 'order_purchase_timestamp': '28-03-2018 08:33', 'order_approved_at': '28-
03-2018 08:50', 'order_delivered_carrier_date': '29-03-2018 18:27', 'order_delivered_customer_date': '02-04-2018 18:33',
 'order_estimated_delivery_date': '16-04-2018 00:00'}
Inserted message into Cassandra with OrderId ee6af093-38de-47dd-746f-2cbbc8f18aed and CustomerId 26c06d7a-d51b-1878-a7a3
-22c755140b01
Received message: {'order_id': UUID('0ad3fdda
8-8d01f3b8f98e'), 'order_status': 'shipped',
-2017 13:45', 'order_delivered_carrier_date':
er_estimated_delivery_date': '08-09-2017 00:0
Inserted message into Cassandra with OrderId
-8d01f3b8f98e
Received message: {'order_id': UUID('d860fd3d
9-b54c2d985c38'), 'order_status': 'delivered'
01-2018 10:35', 'order_delivered_carrier_date
 'order_estimated_delivery_date': '06-02-2018
Inserted message into Cassandra with OrderId
-b54c2d985c38
Received message: {'order_id': UUID('1f04bac5
8-666fe8835a63'), 'order_status': 'delivered'
07-2018 17:30', 'order_delivered_carrier_date
 'order_estimated_delivery_date': '07-08-2018
Inserted message into Cassandra with OrderId
-666fe8835a63
Received message: {'order_id': UUID('f47b9b38
6-b7193e1f0467'), 'order_status': 'delivered'
07-2018 16:25', 'order_delivered_carrier_date
 'order_estimated_delivery_date': '02-08-2018
```

```
Command Prompt - python  orders_data_producer.py
livery_date': '20-06-2017 00:00'}
Produced message: {'order_id': '66e7b21711a5ac0d6b6a5f60e4c79a75', 'customer_id': '7f6e0647dfd
rder_status': 'delivered', 'order_purchase_timestamp': '28-11-2017 14:31', 'order_approved_at'
er_delivered_carrier_date': '30-11-2017 19:21', 'order_delivered_customer_date': '03-01-2018 1
livery_date': '21-12-2017 00:00'}
Produced message: {'order_id': '002f16b7bc4530031b7d90f791b12d8a', 'customer_id': '1eeffe21744
rder_status': 'delivered', 'order_purchase_timestamp': '02-07-2018 16:51', 'order_approved_at'
er_delivered_carrier_date': '03-07-2018 16:35', 'order_delivered_customer_date': '06-07-2018 1
livery_date': '25-07-2018 00:00'}
Produced message: {'order_id': 'd8d281c1a1398a1e835fdc4adfb81e2a', 'customer_id': '14c1e4101ca
rder_status': 'delivered', 'order_purchase_timestamp': '23-07-2018 20:50', 'order_approved_at'
er_delivered_carrier_date': '25-07-2018 15:09', 'order_delivered_customer_date': '27-07-2018 1
livery_date': '07-08-2018 00:00'}
Produced message: {'order_id': '2b781b53b653523cf1337a742b71aed3', 'customer_id': '313d46bf885
rder_status': 'delivered', 'order_purchase_timestamp': '12-10-2017 09:33', 'order_approved_at'
er_delivered_carrier_date': '18-10-2017 16:59', 'order_delivered_customer_date': '20-10-2017 1
livery_date': '31-10-2017 00:00'}
Produced message: {'order_id': 'cb418bb4b6115438d20b0141ed5bc9e0', 'customer_id': 'a7ea318cbe9
rder_status': 'delivered', 'order_purchase_timestamp': '07-06-2018 13:53', 'order_approved_at'
er_delivered_carrier_date': '08-06-2018 15:11', 'order_delivered_customer_date': '16-06-2018 1
livery_date': '18-07-2018 00:00'}
Produced message: {'order_id': 'dcedac1406b3713259927ba54a3ad206', 'customer_id': 'a8695124db5
```

10. Verified the data in the Cassandra table matches the processed data and that all transformations have been executed correctly.



Dashboard / Serverless Databases

ecommerce_db Active

Database ID: 20e6856c-df98-488b-a277-65bef7f33c93

Overview   Health   Connect   CQL Console   CDC   Settings

Usage for Current Billing Period

Read Requests	Write Requests	Storage Consumed	Data Transfer
117.1k	99.4k	10.23 MB	85.2 MB