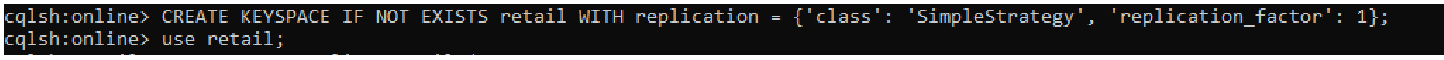
BITS ID	NAME	CONTRIBUTION	UUID
2023DA04255	Arul M	100%	DAC28570-6DA8-0000-0000-000000000000
2023DA04050	SAI SARAVAN LANKA	100%	0BEF6788-4C28-8A17-AB78-D8BBC15EBD96
2023DA04425	SOMA RAHUL LAXMAN	100%	31444335-3930-3844-4A38-383839304435

Configuration details of Cassandra installation. You may include only the lines added / modified by you in the configuration files.

- Downloaded jdk and python following steps given in the Assignment along with Cassandra and extract it.
- Python2.7 and JDK 1.8.0 – updated the environment Variables Path and JAVA_HOME and JAVA_PATH and add CASSANDRA_HOME as requested in Assignment setup.
- Converted File to UTF-8 using bash
- iconv -f ISO-8859-1 -t UTF-8 Assignment1_online_retail_dataset.csv > Assignment1_UTF-8_Converted_online_retail_dataset.csv
- Updated cassandra /conf/cassandra.yaml file as below to handle timeout issue.
  - ***read_request_timeout_in_ms: 50000***
  - ***range_request_timeout_in_ms: 50000***
  - ***write_request_timeout_in_ms: 50000***
  - ***counter_write_request_timeout_in_ms: 50000***
  - ***cas_contention_timeout_in_ms: 50000***
  - ***truncate_request_timeout_in_ms: 60000***
  - ***request_timeout_in_ms: 50000***
- Updated cassandra /conf/cassandra.yaml file as below to enable user defined functions.
  - ***enable_user_defined_functions: true***
- Updated cassandra/bin/cqlsh.py as below to handle timeout issue.
  - ***DEFAULT_CONNECT_TIMEOUT_SECONDS = 6000***
  - ***DEFAULT_REQUEST_TIMEOUT_SECONDS = 6000***
- Connected to Cassandra from cassandra/bin/
  - ***"INFO [main] 2025-01-03 16:19:50,530 StorageService.java:2408 - Node localhost/127.0.0.1 state jump to NORMAL"***
- Connected to CQLSH to perform Analytical Queries.

The definition of the Column Family (Table). Include the cqlsh create command used by you. Clearly mention the primary and partition keys.

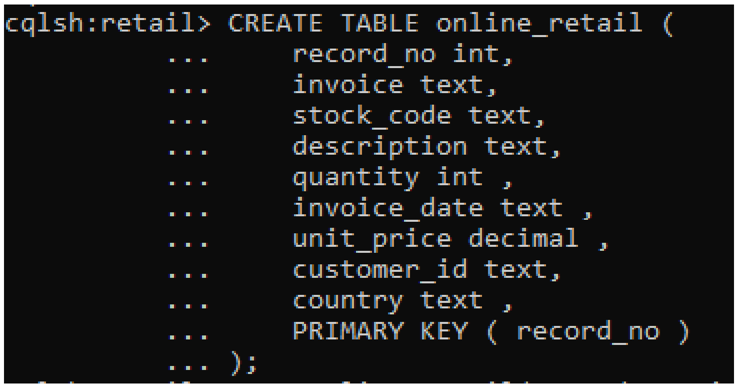
- CREATE KEYSPACE IF NOT EXISTS retail WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
- USE retail;



```
cqlsh:online> CREATE KEYSPACE IF NOT EXISTS retail WITH replication = {'class': 'SimpleStrategy', 'replication_factor': 1};
cqlsh:online> use retail;
```

- CREATE TABLE online_retail (

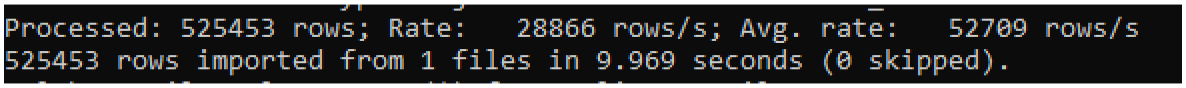
```
    record_no int,
    invoice text,
    stock_code text,
    description text,
    quantity int ,
    invoice_date text ,
    unit_price decimal ,
    customer_id text,
    country text ,
    PRIMARY KEY (record_no )
     );
```



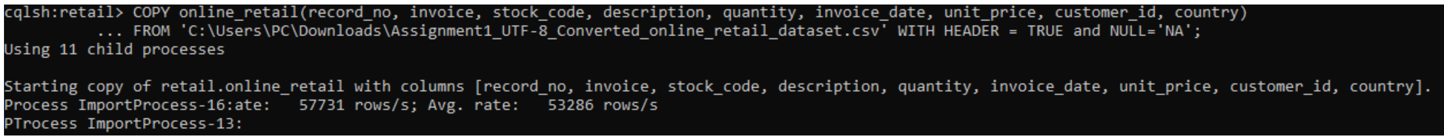
```
cqlsh:retail> CREATE TABLE online_retail (
          ...     record_no int,
          ...     invoice text,
          ...     stock_code text,
          ...     description text,
          ...     quantity int ,
          ...     invoice_date text ,
          ...     unit_price decimal ,
          ...     customer_id text,
          ...     country text ,
          ...     PRIMARY KEY ( record_no )
          ... );
```

The command used to COPY the data file into the Cassandra column family.

- COPY online_retail(record_no, invoice, stock_code, description, quantity, invoice_date, unit_price, customer_id, country)
  FROM 'C:\Users\PC\Downloads\Assignment1_UTF-8_Converted_online_retail_dataset.csv' WITH HEADER = TRUE and NULL='NA';



```
Processed: 525453 rows; Rate:   28866 rows/s; Avg. rate:   52709 rows/s
525453 rows imported from 1 files in 9.969 seconds (0 skipped).
```



```
cqlsh:retail> COPY online_retail(record_no, invoice, stock_code, description, quantity, invoice_date, unit_price, customer_id, country)
          ... FROM 'C:\Users\PC\Downloads\Assignment1_UTF-8_Converted_online_retail_dataset.csv' WITH HEADER = TRUE and NULL='NA';
Using 11 child processes

Starting copy of retail.online_retail with columns [record_no, invoice, stock_code, description, quantity, invoice_date, unit_price, customer_id, country].
Process ImportProcess-16:ate:   57731 rows/s; Avg. rate:   53286 rows/s
PTrocess ImportProcess-13:
```

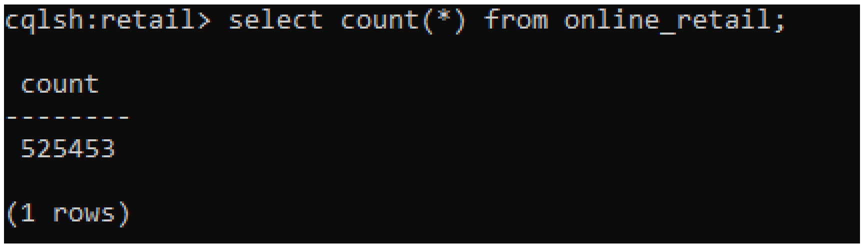
The 6 analytical queries developed by you and their results. The results should follow each query.

1. **Find total number of transactions in the given transaction file.**

   select count(*) from online_retail;

   count

   525453



```
cqlsh:retail> select count(*) from online_retail;

 count
--------
 525453

(1 rows)
```

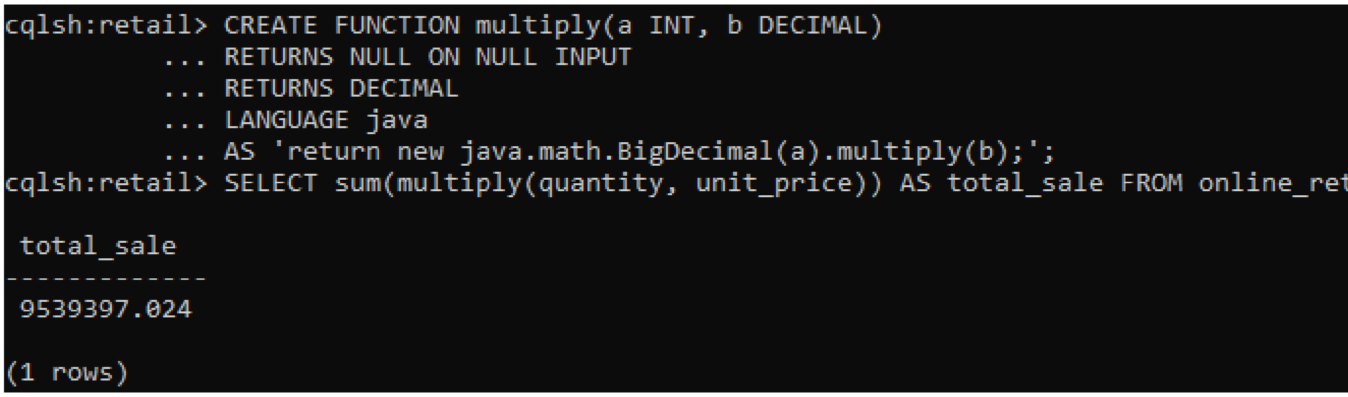
2. **Find total value of sale happened in the year 2009-2010.**
   a. **(Sale Price = Quantity*UnitPrice).**

```
CREATE FUNCTION multiply(a INT, b DECIMAL)
RETURNS NULL ON NULL INPUT
RETURNS DECIMAL
LANGUAGE java
    AS 'return new java.math.BigDecimal(a).multiply(b);';
```

SELECT sum(multiply(quantity, unit_price)) AS total_sale FROM online_retail;

total_sale

9539397.024



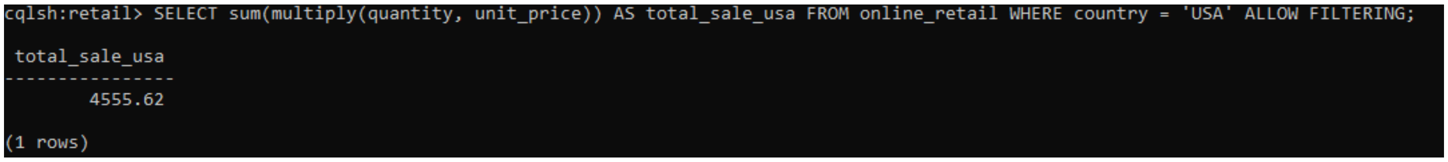
```
cqlsh:retail> CREATE FUNCTION multiply(a INT, b DECIMAL)
          ... RETURNS NULL ON NULL INPUT
          ... RETURNS DECIMAL
          ... LANGUAGE java
          ... AS 'return new java.math.BigDecimal(a).multiply(b);';
cqlsh:retail> SELECT sum(multiply(quantity, unit_price)) AS total_sale FROM online_ret

 total_sale
-------------
 9539397.024

(1 rows)
```

3. **Find total value of sale happened in USA.**

SELECT sum(multiply(quantity, unit_price)) AS total_sale_usa FROM online_retail WHERE country = 'USA' ALLOW FILTERING;

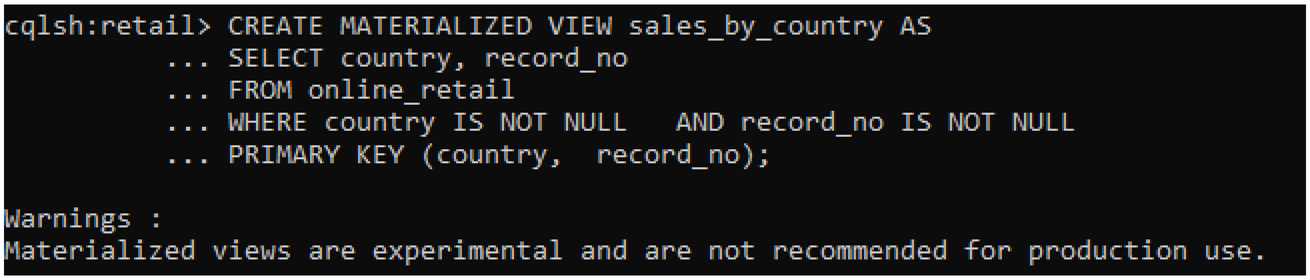
total_sale_usa

4555.62

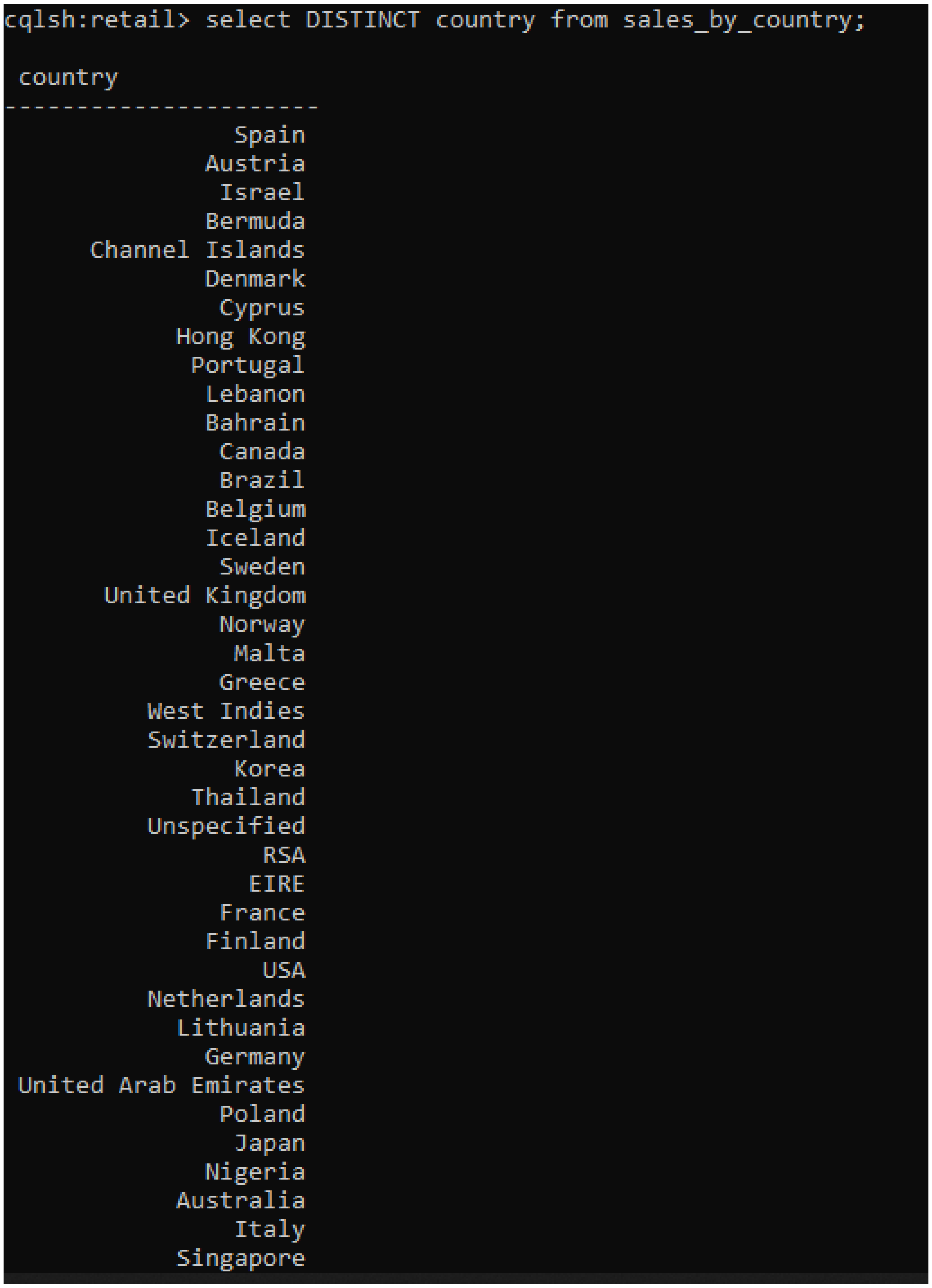


```
cqlsh:retail> SELECT sum(multiply(quantity, unit_price)) AS total_sale_usa FROM online_retail WHERE country = 'USA' ALLOW FILTERING;

 total_sale_usa
----------------
        4555.62

(1 rows)
```

4. **List of countries from which purchases were made on the online shop.**



```
cqlsh:retail> CREATE MATERIALIZED VIEW sales_by_country AS
          ... SELECT country, record_no
          ... FROM online_retail
          ... WHERE country IS NOT NULL   AND record_no IS NOT NULL
          ... PRIMARY KEY (country,  record_no);

Warnings :
Materialized views are experimental and are not recommended for production use.
```

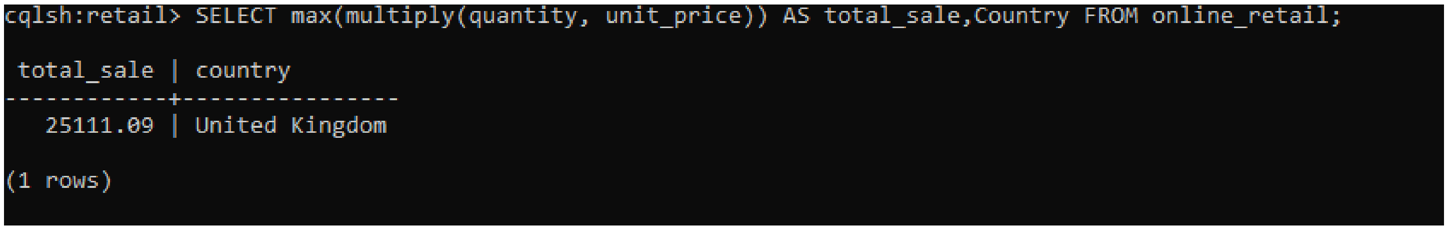


```
cqlsh:retail> select DISTINCT country from sales_by_country;

 country
----------------------
                Spain
              Austria
               Israel
              Bermuda
      Channel Islands
              Denmark
               Cyprus
            Hong Kong
             Portugal
              Lebanon
              Bahrain
               Canada
               Brazil
              Belgium
              Iceland
               Sweden
       United Kingdom
               Norway
                Malta
               Greece
          West Indies
          Switzerland
                Korea
             Thailand
          Unspecified
                  RSA
                 EIRE
               France
              Finland
                  USA
          Netherlands
            Lithuania
              Germany
 United Arab Emirates
               Poland
                Japan
              Nigeria
            Australia
                Italy
            Singapore
```

5. **Find country from which sale with the maximum value happened.**

   SELECT max(multiply(quantity, unit_price)) AS total_sale,Country FROM online_retail;

```
total_sale | country
------------+---------
25111.09 |   United Kingdom
```

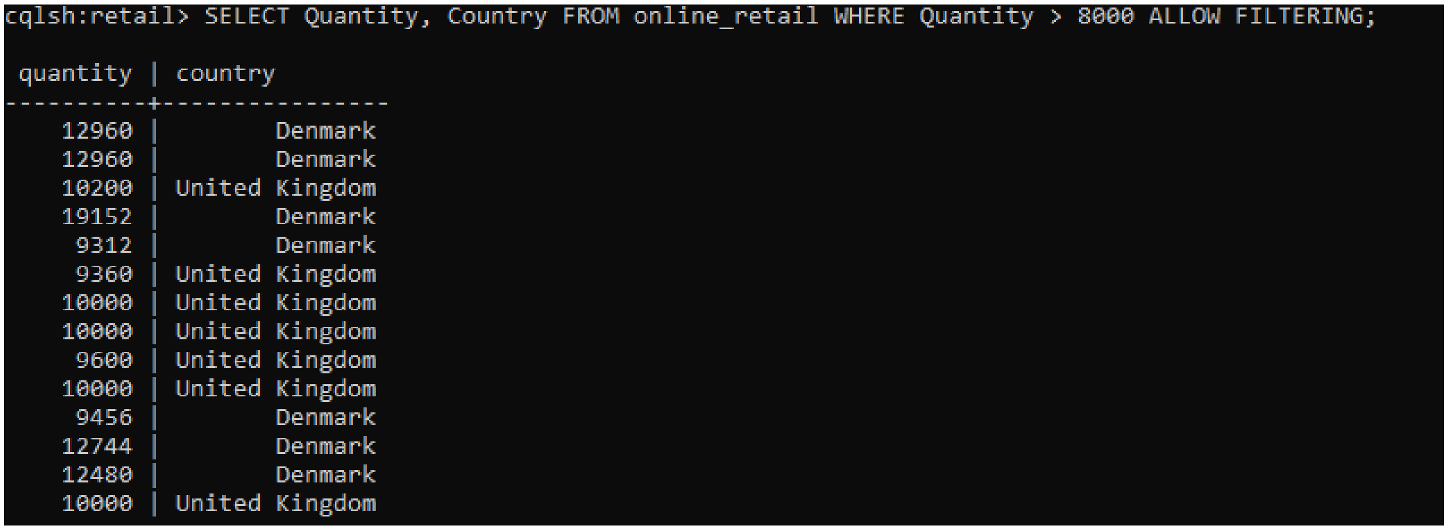


```
cqlsh:retail> SELECT max(multiply(quantity, unit_price)) AS total_sale,Country FROM online_retail;

 total_sale | country
------------+----------------
   25111.09 | United Kingdom

(1 rows)
```

6. **Find the countries from which the quantity in one sale of an item was**
   a. **more than 8000**

      SELECT Quantity, Country FROM online_retail WHERE Quantity > 8000 ALLOW FILTERING;

```
quantity | country
----------+----------------
    12960 |        Denmark
    12960 |        Denmark
    10200 | United Kingdom
    19152 |        Denmark
     9312 |        Denmark
     9360 | United Kingdom
    10000 | United Kingdom
    10000 | United Kingdom
     9600 | United Kingdom
    10000 | United Kingdom
     9456 |        Denmark
    12744 |        Denmark
    12480 |        Denmark
    10000 | United Kingdom
```



```
cqlsh:retail> SELECT Quantity, Country FROM online_retail WHERE Quantity > 8000 ALLOW FILTERING;

 quantity | country
----------+----------------
    12960 |        Denmark
    12960 |        Denmark
    10200 | United Kingdom
    19152 |        Denmark
     9312 |        Denmark
     9360 | United Kingdom
    10000 | United Kingdom
    10000 | United Kingdom
     9600 | United Kingdom
    10000 | United Kingdom
     9456 |        Denmark
    12744 |        Denmark
    12480 |        Denmark
    10000 | United Kingdom
```