In [1]:
```
from pyspark.sql import SparkSession
import getpass
username =getpass.getuser()
spark = SparkSession. \
    builder. \
    config('spark.ui.port', '0'). \
    config("spark.sql.warehouse.dir", f"/user/{username}/warehouse"). \
    enableHiveSupport(). \
    master('yarn'). \
    getOrCreate()
```

In [2]:
```
spark
```

Out[2]:
SparkSession - hive
SparkContext

Spark UI (http://g01.itversity.com:44671)
Version
v3.1.2
Master
yarn
AppName
pyspark-shell

In [3]:
```
orders_rdd = spark.sparkContext.textFile("/public/trendytech/retail_db/orders/*")
```

In [4]:
```
orders_rdd.take(10)
```

Out[4]:
```
['1,2013-07-25 00:00:00.0,11599,CLOSED',
 '2,2013-07-25 00:00:00.0,256,PENDING_PAYMENT',
 '3,2013-07-25 00:00:00.0,12111,COMPLETE',
 '4,2013-07-25 00:00:00.0,8827,CLOSED',
 '5,2013-07-25 00:00:00.0,11318,COMPLETE',
 '6,2013-07-25 00:00:00.0,7130,COMPLETE',
 '7,2013-07-25 00:00:00.0,4530,COMPLETE',
 '8,2013-07-25 00:00:00.0,2911,PROCESSING',
 '9,2013-07-25 00:00:00.0,5657,PENDING_PAYMENT',
 '10,2013-07-25 00:00:00.0,5648,PENDING_PAYMENT']
```

1)count the orders in each category status

In [7]:
```
mapped_rdd=orders_rdd.map(lambda x: (x.split(",")[3],1))
```

In [8]:
```
mapped_rdd.take(10)
```

Out[8]:
```
[('CLOSED', 1),
 ('PENDING_PAYMENT', 1),
 ('COMPLETE', 1),
 ('CLOSED', 1),
 ('COMPLETE', 1),
 ('COMPLETE', 1),
 ('COMPLETE', 1),
 ('PROCESSING', 1),
 ('PENDING_PAYMENT', 1),
 ('PENDING_PAYMENT', 1)]
```

In [9]:
```
reduced_rdd=mapped_rdd.reduceByKey(lambda x,y : x+y)
```

In [10]:
```
reduced_rdd.collect()
```

Out[10]:
```
[('CLOSED', 7556),
 ('CANCELED', 1428),
 ('COMPLETE', 22899),
 ('PENDING_PAYMENT', 15030),
 ('SUSPECTED_FRAUD', 1558),
 ('PENDING', 7610),
 ('ON_HOLD', 3798),
 ('PROCESSING', 8275),
 ('PAYMENT_REVIEW', 729)]
```

In [13]:
```
sorted_rdd=reduced_rdd.sortBy(lambda x: x[1], False)
```

In [14]:
```
sorted_rdd.collect()
```

Out[14]:
```
[('COMPLETE', 22899),
 ('PENDING_PAYMENT', 15030),
 ('PROCESSING', 8275),
 ('PENDING', 7610),
 ('CLOSED', 7556),
 ('ON_HOLD', 3798),
 ('SUSPECTED_FRAUD', 1558),
 ('CANCELED', 1428),
 ('PAYMENT_REVIEW', 729)]
```

2) find the premium customers (top 10 who places most numbers of orders)

In [15]:
```
cust_mapped=orders_rdd.map(lambda x: (x.split(",")[2],1))
```

In [16]:
```
cust_mapped.take(5)
```

Out[16]:
```
[('11599', 1), ('256', 1), ('12111', 1), ('8827', 1), ('11318', 1)]
```

In [17]:
```
cust_agg =cust_mapped.reduceByKey(lambda x,y :x+y)
```

In [18]:
```
cust_agg.take(20)
```

Out[18]:
```
[('3066', 6),
 ('3159', 7),
 ('8135', 11),
 ('2248', 4),
 ('6117', 6),
 ('7733', 7),
 ('6540', 3),
 ('4882', 8),
 ('6060', 7),
 ('10436', 8),
 ('11478', 6),
 ('8549', 5),
 ('5834', 4),
 ('9419', 10),
 ('3478', 4),
 ('12328', 7),
 ('833', 6),
 ('5279', 7),
 ('8506', 5),
 ('8316', 5)]
```

In [20]:
```
cust_sorted=cust_agg.sortBy(lambda x :x[1], False)
```

In [21]:
```
cust_sorted.take(10)
```

Out[21]:
```
[('5897', 16),
 ('6316', 16),
 ('12431', 16),
 ('569', 16),
 ('221', 15),
 ('4320', 15),
 ('5624', 15),
 ('5283', 15),
 ('12284', 15),
 ('5654', 15)]
```

3) disticnt count of customers who placed atleast one order

In [22]:
```
dist_cust=orders_rdd.map(lambda x: x.split(",")[2]).distinct()
```

In [23]:
```
dist_cust.count()
```

Out[23]:
```
12405
```

4) which cust has the max no. of closed order

In [6]:
```
filters_orders=orders_rdd.filter(lambda x : (x.split(",")[3] =='CLOSED'))
```

In [7]:
```
filters_orders.take(20)
```

Out[7]:
```
['1,2013-07-25 00:00:00.0,11599,CLOSED',
 '4,2013-07-25 00:00:00.0,8827,CLOSED',
 '12,2013-07-25 00:00:00.0,1837,CLOSED',
 '18,2013-07-25 00:00:00.0,1205,CLOSED',
 '24,2013-07-25 00:00:00.0,11441,CLOSED',
 '25,2013-07-25 00:00:00.0,9503,CLOSED',
 '37,2013-07-25 00:00:00.0,5863,CLOSED',
 '51,2013-07-25 00:00:00.0,12271,CLOSED',
 '57,2013-07-25 00:00:00.0,7073,CLOSED',
 '61,2013-07-25 00:00:00.0,4791,CLOSED',
 '62,2013-07-25 00:00:00.0,9111,CLOSED',
 '87,2013-07-25 00:00:00.0,3065,CLOSED',
 '90,2013-07-25 00:00:00.0,9131,CLOSED',
 '101,2013-07-25 00:00:00.0,5116,CLOSED',
 '116,2013-07-26 00:00:00.0,8763,CLOSED',
 '129,2013-07-26 00:00:00.0,9937,CLOSED',
 '133,2013-07-26 00:00:00.0,10604,CLOSED',
 '191,2013-07-26 00:00:00.0,16,CLOSED',
 '201,2013-07-26 00:00:00.0,9055,CLOSED',
 '211,2013-07-26 00:00:00.0,10372,CLOSED']
```

In [8]:
```
filtered_mapped=filters_orders.map(lambda x : (x.split(",")[2],1))
```

In [9]:
```
filtered_mapped.take(10)
```

Out[9]:
```
[('11599', 1),
 ('8827', 1),
 ('1837', 1),
 ('1205', 1),
 ('11441', 1),
 ('9503', 1),
 ('5863', 1),
 ('12271', 1),
 ('7073', 1),
 ('4791', 1)]
```

In [10]:
```
agg_flitered=filtered_mapped.reduceByKey(lambda x,y : x+y )
```

In [11]:
```
agg_flitered.take(20)
```

Out[11]:
```
[('3159', 1),
 ('5834', 2),
 ('10173', 1),
 ('2101', 1),
 ('6000', 1),
 ('1352', 2),
 ('10142', 1),
 ('12210', 1),
 ('6018', 2),
 ('2252', 1),
 ('10290', 2),
 ('9117', 1),
 ('7600', 2),
 ('6482', 1),
 ('9420', 1),
 ('11673', 3),
 ('7435', 2),
 ('7879', 4),
 ('11153', 3),
 ('9771', 1)]
```

In [12]:
```
flitered_sorted=agg_flitered.sortBy(lambda x : x[1], False)
```

In [13]:
```
flitered_sorted.take(10)
```

Out[13]:
```
[('1833', 6),
 ('1363', 5),
 ('1687', 5),
 ('5493', 5),
 ('5011', 4),
 ('8974', 4),
 ('2321', 4),
 ('3736', 4),
 ('8368', 4),
 ('2236', 4)]
```

In [ ]:
```

```