SCHOOL OF COMPUTING (SOC)

CA2
Step-by-step Tutorial

SPECIALIST DIPLOMA IN DATA SCIENCE
(BIG DATA & STREAMING ANALYTICS)

IT8703 Streaming Analytics

Date of Submission: 15th February 2020

Table of Contents

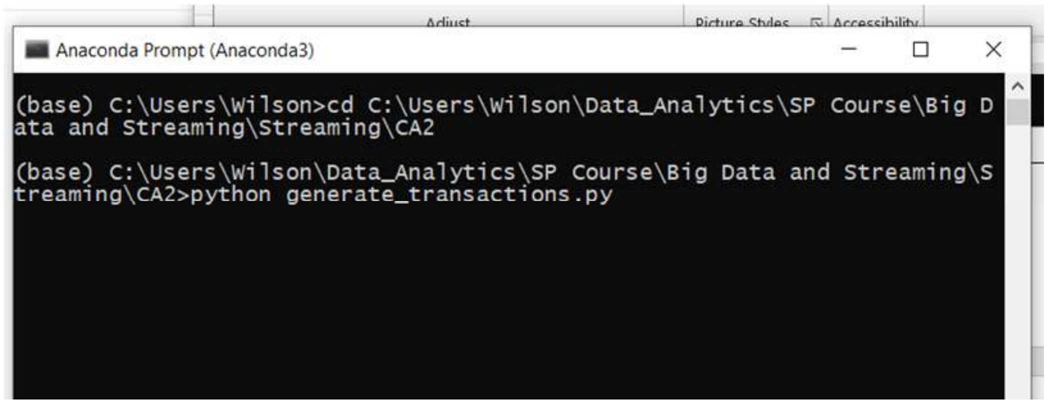
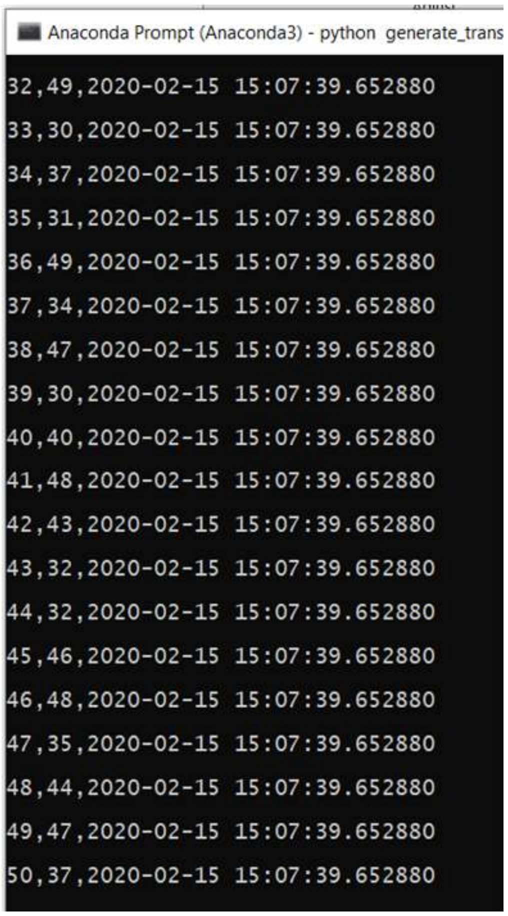
Section A – Propose your own header

Step 1 – generate_transactions.py

1. Random Transactions generations python code :

```
from datetime import datetime
import sys
import random
from time import sleep


# We will imagine that we have 8 rooms installed with light sensors
# and we constantly capture the light levels of these 8 rooms every 5 seconds
# in the subfolder iotdata

# Read data
import pandas as pd
df = pd.read_csv("bakeinc_products.csv")

product_id = df['ProductID']

while True:
    try:
        dt = datetime.now()
        # Start a new file based on current timestamp
        fn = "{}{}{}{}{}{}".format(dt.year,dt.month,dt.day,dt.hour,dt.minute,dt.second)
        f = open("data/" + fn + ".txt","w+") # Open the file for writing
        for product in product_id:
            quantity = random.randint(30,50) # Generate a random quantity purchased
            data = "{},{},{}\n".format(product,quantity,dt)
            f.write(data) # Write into the file
            print(data)
        f.close()
        sleep(5) # Wait another 5 seconds before collecting the next batch of data
    except KeyboardInterrupt:
        print('Interrupted')
        sys.exit()
    except:
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])
```

Step 2 – Output of the generate_transactions.py

1.	Console output display of the transactions :  

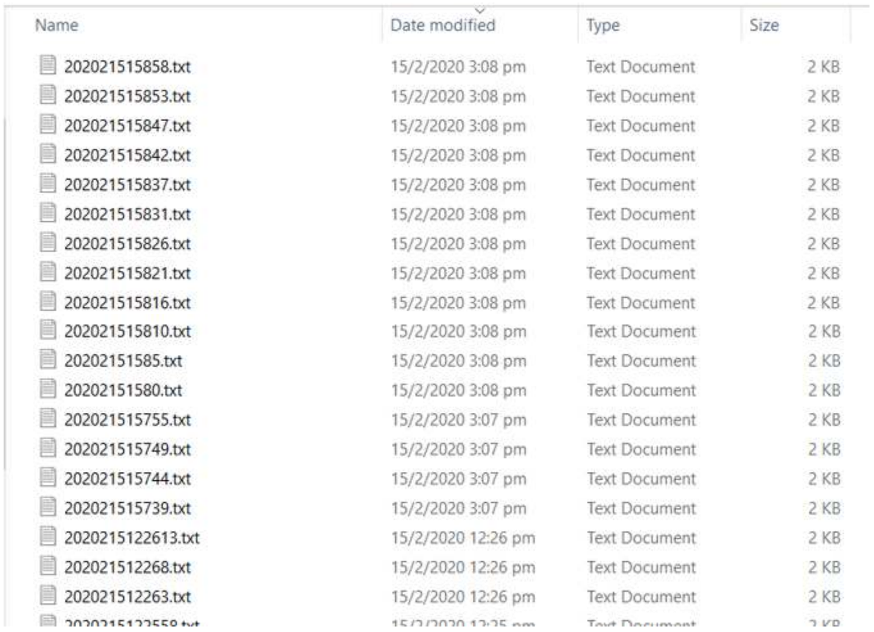
Anaconda Prompt (Anaconda3)

```
(base) C:\Users\Wilson>cd C:\Users\Wilson\Data_Analytics\SP Course\Big D
ata and Streaming\Streaming\CA2

(base) C:\Users\Wilson\Data_Analytics\SP Course\Big Data and Streaming\S
treaming\CA2>python generate_transactions.py
```

Anaconda Prompt (Anaconda3) - python generate_trans

```
32,49,2020-02-15 15:07:39.652880
33,30,2020-02-15 15:07:39.652880
34,37,2020-02-15 15:07:39.652880
35,31,2020-02-15 15:07:39.652880
36,49,2020-02-15 15:07:39.652880
37,34,2020-02-15 15:07:39.652880
38,47,2020-02-15 15:07:39.652880
39,30,2020-02-15 15:07:39.652880
40,40,2020-02-15 15:07:39.652880
41,48,2020-02-15 15:07:39.652880
42,43,2020-02-15 15:07:39.652880
43,32,2020-02-15 15:07:39.652880
44,32,2020-02-15 15:07:39.652880
45,46,2020-02-15 15:07:39.652880
46,48,2020-02-15 15:07:39.652880
47,35,2020-02-15 15:07:39.652880
48,44,2020-02-15 15:07:39.652880
49,47,2020-02-15 15:07:39.652880
50,37,2020-02-15 15:07:39.652880
```

Output the data to text file

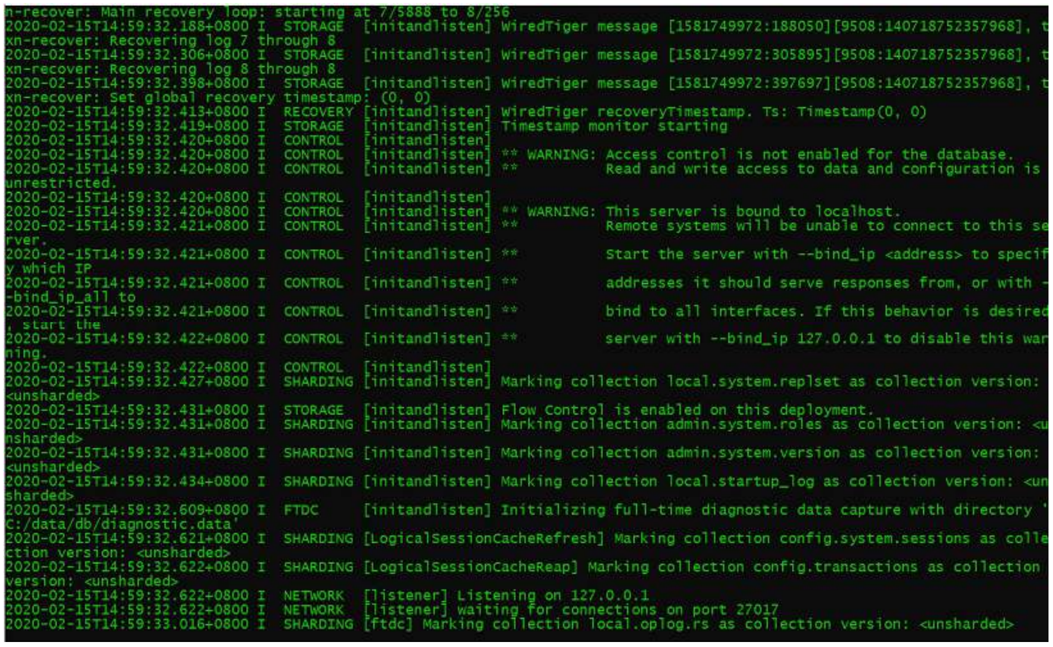


Name	Date modified	Type	Size
202021515858.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515853.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515847.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515842.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515837.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515831.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515826.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515821.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515816.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515810.txt	15/2/2020 3:08 pm	Text Document	2 KB
20202151585.txt	15/2/2020 3:08 pm	Text Document	2 KB
20202151580.txt	15/2/2020 3:08 pm	Text Document	2 KB
202021515755.txt	15/2/2020 3:07 pm	Text Document	2 KB
202021515749.txt	15/2/2020 3:07 pm	Text Document	2 KB
202021515744.txt	15/2/2020 3:07 pm	Text Document	2 KB
202021515739.txt	15/2/2020 3:07 pm	Text Document	2 KB
2020215122613.txt	15/2/2020 12:26 pm	Text Document	2 KB
202021512268.txt	15/2/2020 12:26 pm	Text Document	2 KB
202021512263.txt	15/2/2020 12:26 pm	Text Document	2 KB

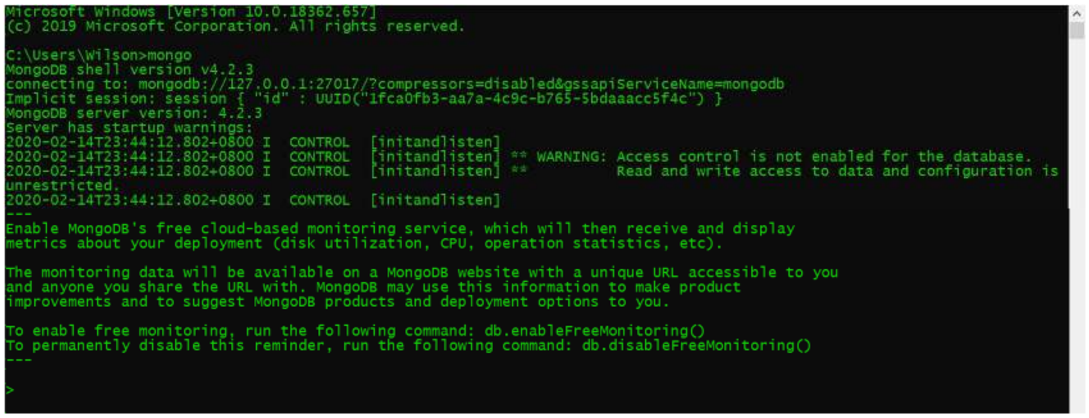
Section B – Mongo Database

Step 1 – Run Mongod and Mongo

1. Running Mongo Database Applications :



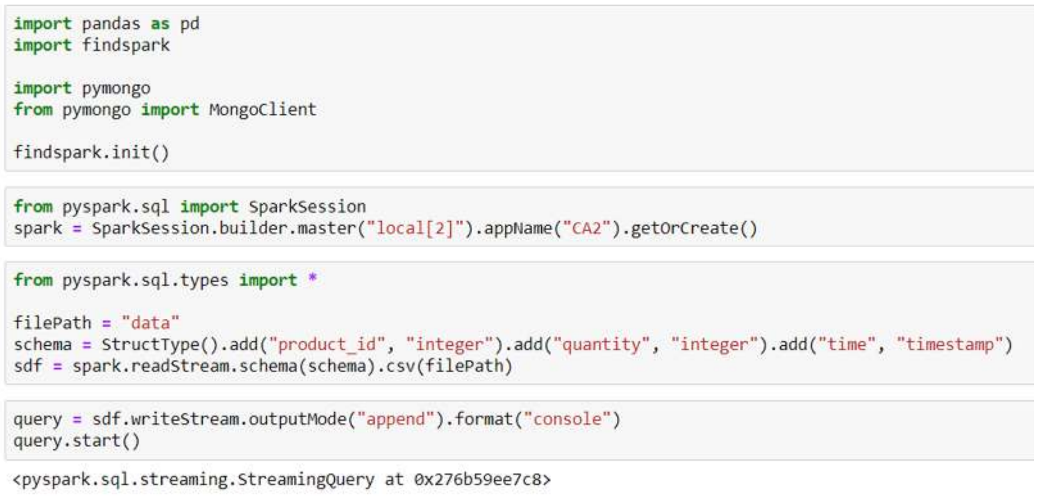
```
n-recover: Main recovery loop: starting at 7/5888 to 8/256
2020-02-15T14:59:32.188+0800 I  STORAGE  [initandlisten] WiredTiger message [1581749972:188050][9508:140718752357968], t
xn-recover: Recovering log 7 through 8
2020-02-15T14:59:32.306+0800 I  STORAGE  [initandlisten] WiredTiger message [1581749972:305895][9508:140718752357968], t
xn-recover: Recovering log 8 through 8
2020-02-15T14:59:32.398+0800 I  STORAGE  [initandlisten] WiredTiger message [1581749972:397697][9508:140718752357968], t
xn-recover: Set global recovery timestamp: (0, 0)
2020-02-15T14:59:32.413+0800 I  RECOVERY [initandlisten] WiredTiger recoveryTimestamp. Ts: Timestamp(0, 0)
2020-02-15T14:59:32.419+0800 I  STORAGE  [initandlisten] Timestamp monitor starting
2020-02-15T14:59:32.420+0800 I  CONTROL  [initandlisten]
2020-02-15T14:59:32.420+0800 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-02-15T14:59:32.420+0800 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is
unrestricted.
2020-02-15T14:59:32.420+0800 I  CONTROL  [initandlisten]
2020-02-15T14:59:32.420+0800 I  CONTROL  [initandlisten] ** WARNING: This server is bound to localhost.
2020-02-15T14:59:32.421+0800 I  CONTROL  [initandlisten] **          Remote systems will be unable to connect to this se
rver.
2020-02-15T14:59:32.421+0800 I  CONTROL  [initandlisten] **          Start the server with --bind_ip <address> to specif
y which IP
2020-02-15T14:59:32.421+0800 I  CONTROL  [initandlisten] **          addresses it should serve responses from, or with -
-bind_ip_all to
2020-02-15T14:59:32.421+0800 I  CONTROL  [initandlisten] **          bind to all interfaces. If this behavior is desired
, start the
2020-02-15T14:59:32.422+0800 I  CONTROL  [initandlisten] **          server with --bind_ip 127.0.0.1 to disable this war
ning.
2020-02-15T14:59:32.422+0800 I  CONTROL  [initandlisten]
2020-02-15T14:59:32.427+0800 I  SHARDING [initandlisten] Marking collection local.system.replset as collection version:
<unsharded>
2020-02-15T14:59:32.431+0800 I  STORAGE  [initandlisten] Flow Control is enabled on this deployment.
2020-02-15T14:59:32.431+0800 I  SHARDING [initandlisten] Marking collection admin.system.roles as collection version: <u
nsharded>
2020-02-15T14:59:32.431+0800 I  SHARDING [initandlisten] Marking collection admin.system.version as collection version:
<unsharded>
2020-02-15T14:59:32.434+0800 I  SHARDING [initandlisten] Marking collection local.startup_log as collection version: <un
sharded>
2020-02-15T14:59:32.609+0800 I  FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory '
C:/data/db/diagnostic.data'
2020-02-15T14:59:32.621+0800 I  SHARDING [LogicalSessionCacheRefresh] Marking collection config.system.sessions as colle
ction version: <unsharded>
2020-02-15T14:59:32.622+0800 I  SHARDING [LogicalSessionCacheReap] Marking collection config.transactions as collection
version: <unsharded>
2020-02-15T14:59:32.622+0800 I  NETWORK  [listener] Listening on 127.0.0.1
2020-02-15T14:59:32.622+0800 I  NETWORK  [listener] waiting for connections on port 27017
2020-02-15T14:59:33.016+0800 I  SHARDING [ftdc] Marking collection local.oplog.rs as collection version: <unsharded>
```



```
Microsoft Windows [Version 10.0.18362.657]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\Wilson>mongo
MongoDB shell version v4.2.3
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("1fca0fb3-aa7a-4c9c-b765-5bdaaacc5f4c") }
MongoDB server version: 4.2.3
Server has startup warnings:
2020-02-14T23:44:12.802+0800 I  CONTROL  [initandlisten]
2020-02-14T23:44:12.802+0800 I  CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2020-02-14T23:44:12.802+0800 I  CONTROL  [initandlisten] **          Read and write access to data and configuration is
unrestricted.
2020-02-14T23:44:12.802+0800 I  CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

Step 2 – Saving transaction data to Mongo DB

1. Reading streaming : CA2 - Streaming-MongoDB.ipynb



```
import pandas as pd
import findspark

import pymongo
from pymongo import MongoClient

findspark.init()
```

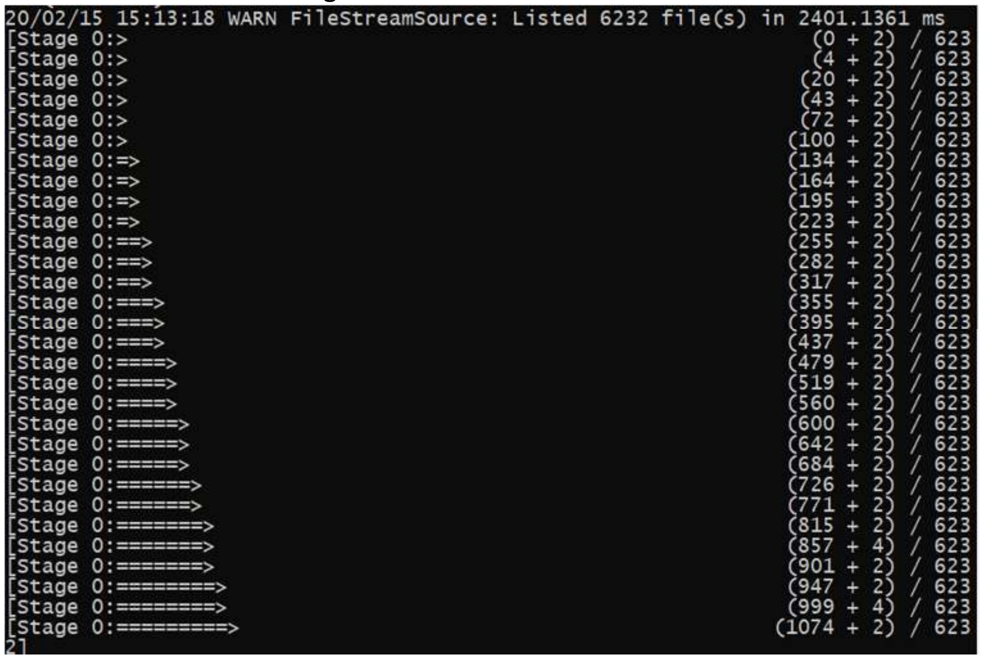
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.master("local[2]").appName("CA2").getOrCreate()
```

```
from pyspark.sql.types import *

filePath = "data"
schema = StructType().add("product_id", "integer").add("quantity", "integer").add("time", "timestamp")
sdf = spark.readStream.schema(schema).csv(filePath)
```

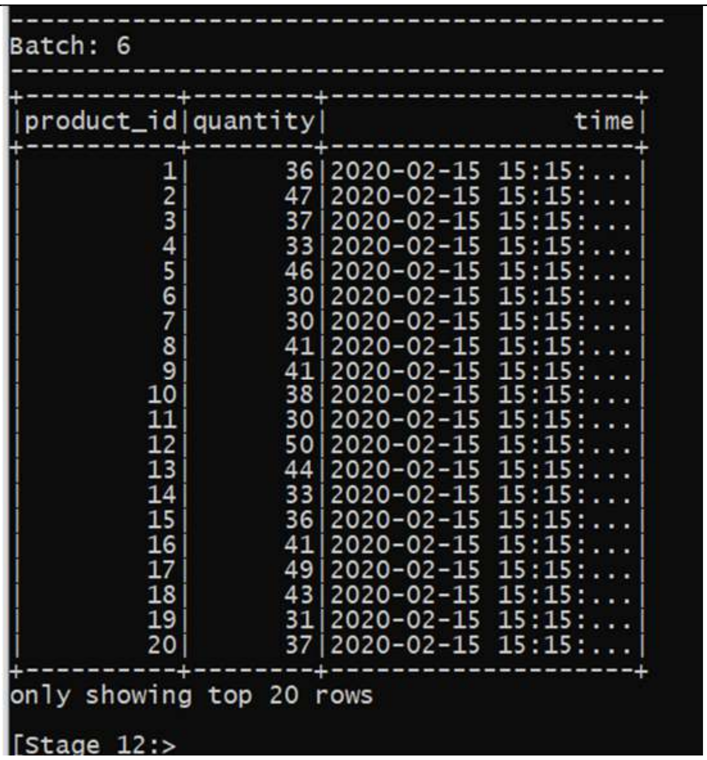
```
query = sdf.writeStream.outputMode("append").format("console")
query.start()
```

```
<pyspark.sql.streaming.StreamingQuery at 0x276b59ee7c8>
```

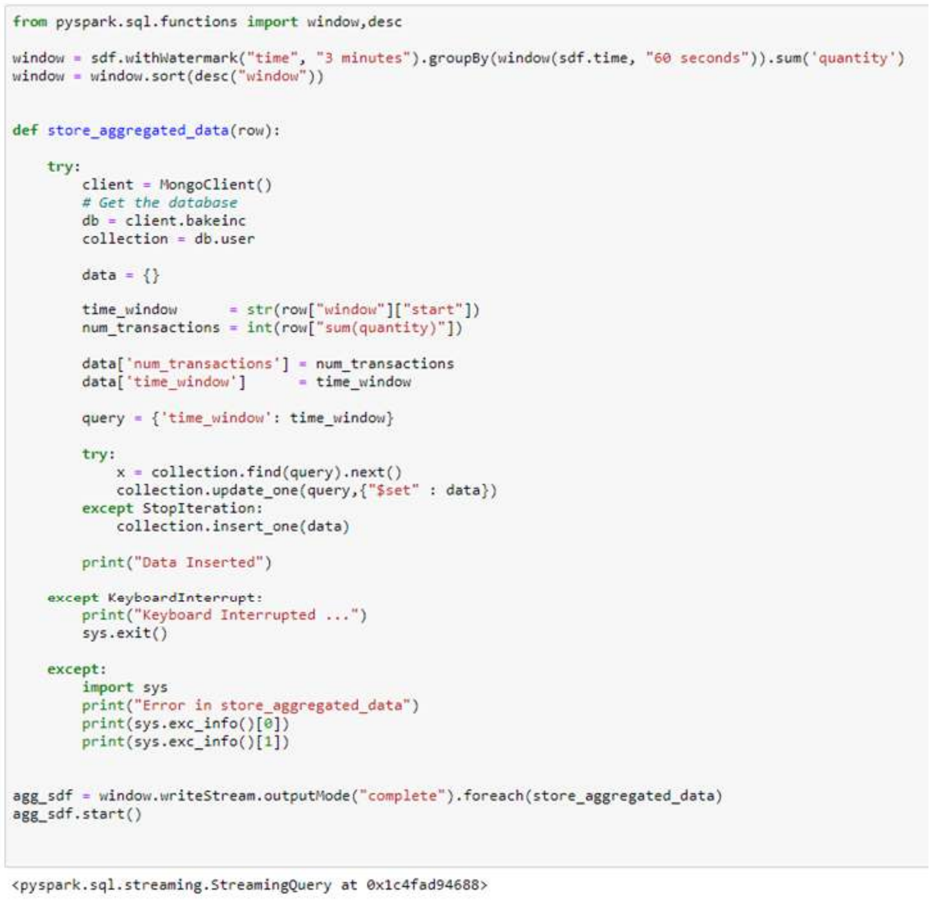
2. Output console of the streaming :



```
20/02/15 15:13:18 WARN FileStreamSource: Listed 6232 file(s) in 2401.1361 ms
[Stage 0:>                                                          (0 + 2) / 623
[Stage 0:>                                                          (4 + 2) / 623
[Stage 0:>                                                         (20 + 2) / 623
[Stage 0:>                                                         (43 + 2) / 623
[Stage 0:>                                                         (72 + 2) / 623
[Stage 0:>                                                        (100 + 2) / 623
[Stage 0:=>                                                       (134 + 2) / 623
[Stage 0:=>                                                       (164 + 2) / 623
[Stage 0:=>                                                       (195 + 3) / 623
[Stage 0:=>                                                       (223 + 2) / 623
[Stage 0:==>                                                      (255 + 2) / 623
[Stage 0:==>                                                      (282 + 2) / 623
[Stage 0:==>                                                      (317 + 2) / 623
[Stage 0:===>                                                     (355 + 2) / 623
[Stage 0:===>                                                     (395 + 2) / 623
[Stage 0:===>                                                     (437 + 2) / 623
[Stage 0:====>                                                    (479 + 2) / 623
[Stage 0:====>                                                    (519 + 2) / 623
[Stage 0:====>                                                    (560 + 2) / 623
[Stage 0:=====>                                                   (600 + 2) / 623
[Stage 0:=====>                                                   (642 + 2) / 623
[Stage 0:=====>                                                   (684 + 2) / 623
[Stage 0:======>                                                  (726 + 2) / 623
[Stage 0:======>                                                  (771 + 2) / 623
[Stage 0:=======>                                                 (815 + 2) / 623
[Stage 0:=======>                                                 (857 + 4) / 623
[Stage 0:=======>                                                 (901 + 2) / 623
[Stage 0:========>                                                (947 + 2) / 623
[Stage 0:========>                                                (999 + 4) / 623
[Stage 0:=========>                                              (1074 + 2) / 623
2]
```

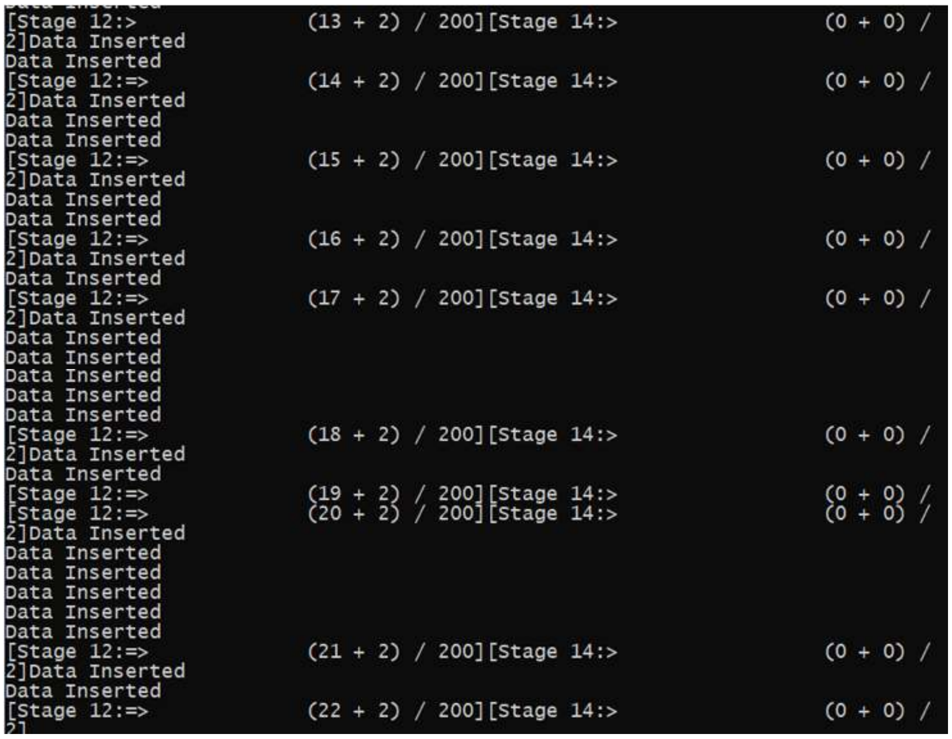
<table>
<tr><td></td><td>

```
-------------------------------------------
Batch: 6
-------------------------------------------
+----------+--------+--------------------+
|product_id|quantity|                time|
+----------+--------+--------------------+
|         1|      36|2020-02-15 15:15:...|
|         2|      47|2020-02-15 15:15:...|
|         3|      37|2020-02-15 15:15:...|
|         4|      33|2020-02-15 15:15:...|
|         5|      46|2020-02-15 15:15:...|
|         6|      30|2020-02-15 15:15:...|
|         7|      30|2020-02-15 15:15:...|
|         8|      41|2020-02-15 15:15:...|
|         9|      41|2020-02-15 15:15:...|
|        10|      38|2020-02-15 15:15:...|
|        11|      30|2020-02-15 15:15:...|
|        12|      50|2020-02-15 15:15:...|
|        13|      44|2020-02-15 15:15:...|
|        14|      33|2020-02-15 15:15:...|
|        15|      36|2020-02-15 15:15:...|
|        16|      41|2020-02-15 15:15:...|
|        17|      49|2020-02-15 15:15:...|
|        18|      43|2020-02-15 15:15:...|
|        19|      31|2020-02-15 15:15:...|
|        20|      37|2020-02-15 15:15:...|
+----------+--------+--------------------+
only showing top 20 rows

[Stage 12:>
```
</td></tr>
<tr><td>3.</td><td>Saving the data to Mongo Database</td></tr>
</table>



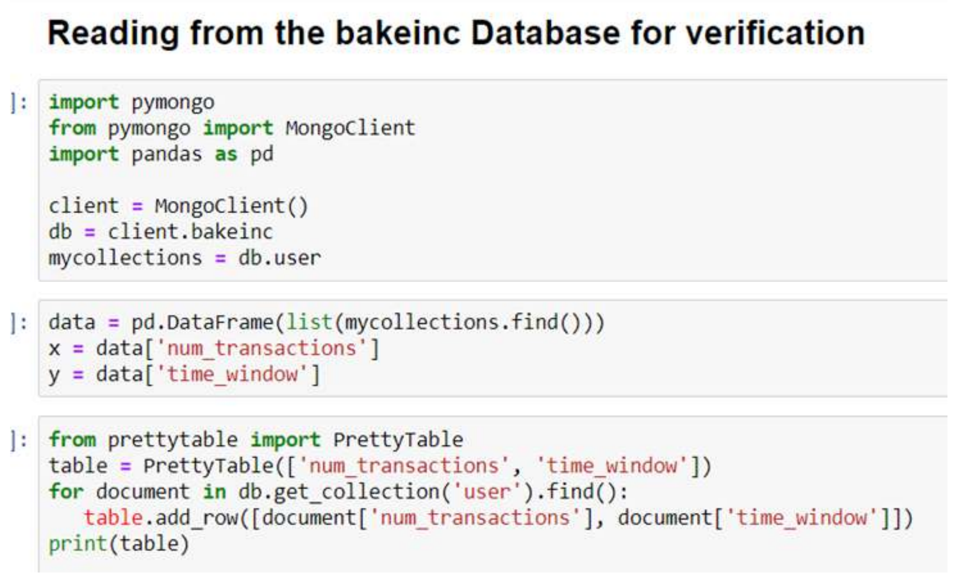
```
from pyspark.sql.functions import window,desc

window = sdf.withWatermark("time", "3 minutes").groupBy(window(sdf.time, "60 seconds")).sum('quantity')
window = window.sort(desc("window"))


def store_aggregated_data(row):

    try:
        client = MongoClient()
        # Get the database
        db = client.bakeinc
        collection = db.user

        data = {}

        time_window      = str(row["window"]["start"])
        num_transactions = int(row["sum(quantity)"])

        data['num_transactions'] = num_transactions
        data['time_window']      = time_window

        query = {'time_window': time_window}

        try:
            x = collection.find(query).next()
            collection.update_one(query,{"$set" : data})
        except StopIteration:
            collection.insert_one(data)

        print("Data Inserted")

    except KeyboardInterrupt:
        print("Keyboard Interrupted ...")
        sys.exit()

    except:
        import sys
        print("Error in store_aggregated_data")
        print(sys.exc_info()[0])
        print(sys.exc_info()[1])


agg_sdf = window.writeStream.outputMode("complete").foreach(store_aggregated_data)
agg_sdf.start()
```

```
<pyspark.sql.streaming.StreamingQuery at 0x1c4fad94688>
```



```
[Stage 12:>                (13 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
[Stage 12:=>               (14 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
Data Inserted
[Stage 12:=>               (15 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
Data Inserted
[Stage 12:=>               (16 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
[Stage 12:=>               (17 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
Data Inserted
Data Inserted
Data Inserted
Data Inserted
[Stage 12:=>               (18 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
[Stage 12:=>               (19 + 2) / 200][Stage 14:>                (0 + 0) /
[Stage 12:=>               (20 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
Data Inserted
Data Inserted
Data Inserted
Data Inserted
[Stage 12:=>               (21 + 2) / 200][Stage 14:>                (0 + 0) /
2]Data Inserted
Data Inserted
[Stage 12:=>               (22 + 2) / 200][Stage 14:>                (0 + 0) /
2]
```

4. Verification that data are inserted to Mongo Database : MongoDB_check.py

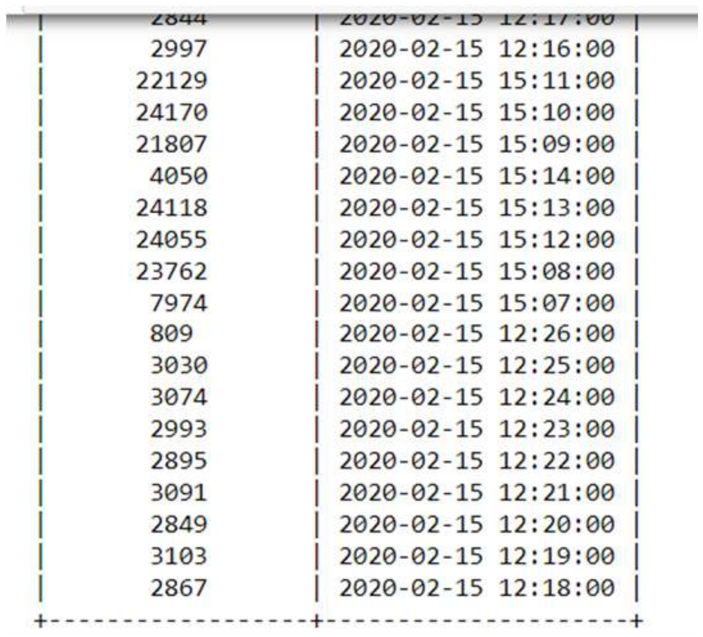


Reading from the bakeinc Database for verification

```
]: import pymongo
   from pymongo import MongoClient
   import pandas as pd

   client = MongoClient()
   db = client.bakeinc
   mycollections = db.user
```

```
]: data = pd.DataFrame(list(mycollections.find()))
   x = data['num_transactions']
   y = data['time_window']
```

```
]: from prettytable import PrettyTable
   table = PrettyTable(['num_transactions', 'time_window'])
   for document in db.get_collection('user').find():
       table.add_row([document['num_transactions'], document['time_window']])
   print(table)
```



```
|       2844        |  2020-02-15 12:17:00 |
|       2997        |  2020-02-15 12:16:00 |
|      22129        |  2020-02-15 15:11:00 |
|      24170        |  2020-02-15 15:10:00 |
|      21807        |  2020-02-15 15:09:00 |
|       4050        |  2020-02-15 15:14:00 |
|      24118        |  2020-02-15 15:13:00 |
|      24055        |  2020-02-15 15:12:00 |
|      23762        |  2020-02-15 15:08:00 |
|       7974        |  2020-02-15 15:07:00 |
|       809         |  2020-02-15 12:26:00 |
|       3030        |  2020-02-15 12:25:00 |
|       3074        |  2020-02-15 12:24:00 |
|       2993        |  2020-02-15 12:23:00 |
|       2895        |  2020-02-15 12:22:00 |
|       3091        |  2020-02-15 12:21:00 |
|       2849        |  2020-02-15 12:20:00 |
|       3103        |  2020-02-15 12:19:00 |
|       2867        |  2020-02-15 12:18:00 |
+-------------------+----------------------+
```

5.

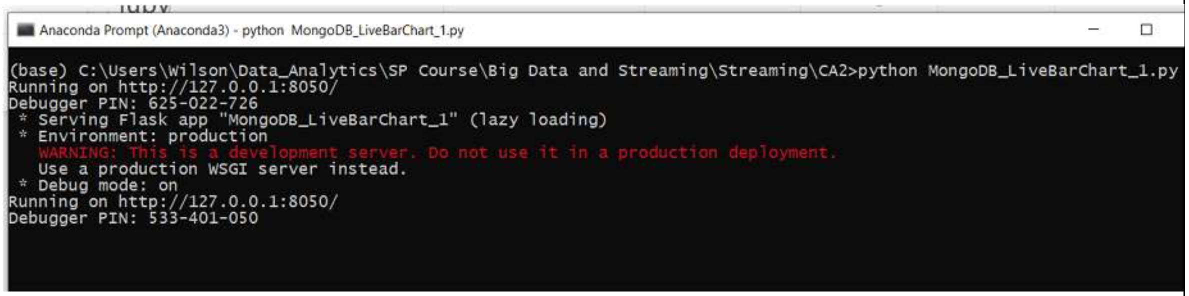
Section C – Web-app of real-time graph

Step 1 – Using Plotly Dash to display realtime Bar Chart

1 Read from MongoDB and display realtime on Dash Bar Chart : MongoDB_LiveBarChart.py

```
import dash
import dash_core_components as dcc
import dash_html_components as html
from dash.dependencies import Input, Output, State
from dash.exceptions import PreventUpdate
import random

import pymongo
from pymongo import MongoClient
import pandas as pd

external_stylesheets = ['https://codepen.io/chriddyp/pen/bWLwgP.css']

app = dash.Dash(__name__, external_stylesheets=external_stylesheets)

app.layout = html.Div(children=[
    html.H1(children='CA2 Streaming Assignment', id='first'),
    dcc.Interval(id='timer', interval=30*1000),
    html.Div(id='dummy'),
    dcc.Graph(
            id='example-graph',
        )
])

@app.callback(output=Output('example-graph', 'figure'),
              inputs=[Input('timer', 'n_intervals')])

def update_graph(n_clicks):

    client = MongoClient()
    db = client.bakeinc
    mycollections = db.user

    X = []
    Y = []

    df = pd.DataFrame(list(mycollections.find()))
    x = df['num_transactions']
    y = df['time_window']

    X = x[-20:].values
    Y = y[-20:].values

    return {
            'data': [
                {'x': Y,
                 'y': X,
                 'type': 'bar', 'name': 'SF'},

            ],
            'layout': {
                'title': 'Dash Data Visualization'
            }
        }


if __name__ == '__main__':
    app.run_server(debug=True)
```

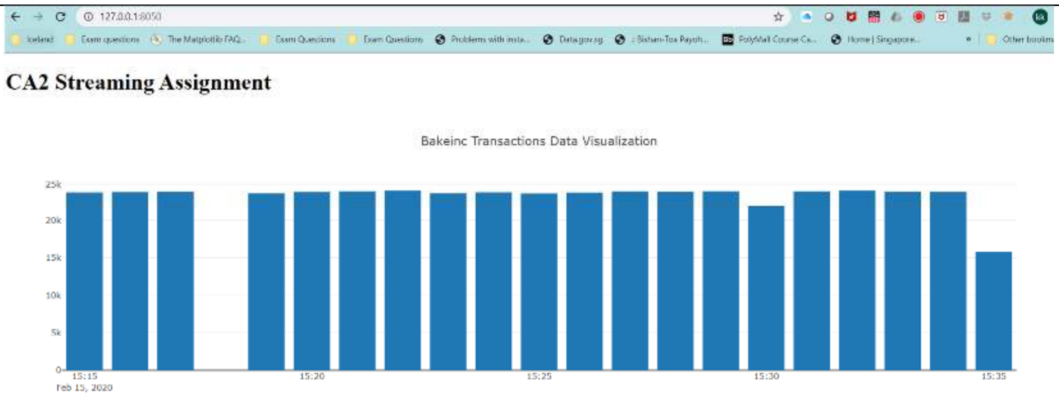
2 Running the dash realtime bar chart display : MongoDB_LiveBarChart_1.py



Anaconda Prompt (Anaconda3) - python MongoDB_LiveBarChart_1.py

```
(base) C:\Users\Wilson\Data_Analytics\SP Course\Big Data and Streaming\Streaming\CA2>python MongoDB_LiveBarChart_1.py
Running on http://127.0.0.1:8050/
Debugger PIN: 625-022-726
 * Serving Flask app "MongoDB_LiveBarChart_1" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
Running on http://127.0.0.1:8050/
Debugger PIN: 533-401-050
```

3

	 127.0.0.1:8050 CA2 Streaming Assignment Bakeinc Transactions Data Visualization 25k 20k 15k 10k 5k 0 15:15 Feb 15, 2020 15:20 15:25 15:30 15:35
4	Video submission link: https://youtu.be/mNPKHdfJAaU
5	

Appendix A
References

References to online materials used

-- End of CA2 Step-by-step tutorial --