!

THE UNIVERSITY OF WINNIPEG

PROFESSIONAL, APPLIED AND CONTINUING EDUCATION

Assignment 6 - Data Streaming with Kafka

by

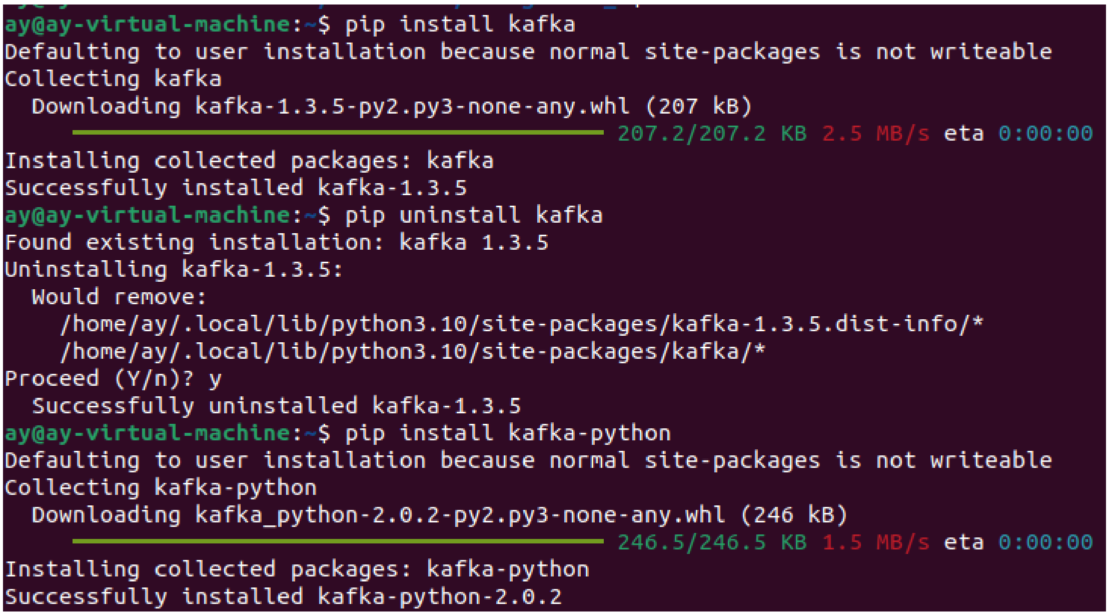
Ayotunde Oyewole

Executive Summary

This project covered basic aspects of data streaming processes. The technical stack involved include Python, postgresql, linux terminal and Apache Kafka. A producer python script was used to generate random weather data within specified ranges. This data simulates data collected from weather sensors in the specified locations (Winnipeg and Vancouver). The weather fields included temperature, wind speed and humidity. This data was encoded as json data and transmitted into a kafka topic.

Another python script - consumer.py was used to retrieve the data, decode the json data format and then load a postgres database. For ease of code regeneration, the Kafka topic, database, and the database tables were all created using python scripts. This also made it possible to explore the creation of a kafka topic using python, since creation through terminal had been explored in class. All activities in the producer and consumer scripts were logged in corresponding log files.

Erroneously installed kafka instead of kafka-python. Then correctly installed kafta.

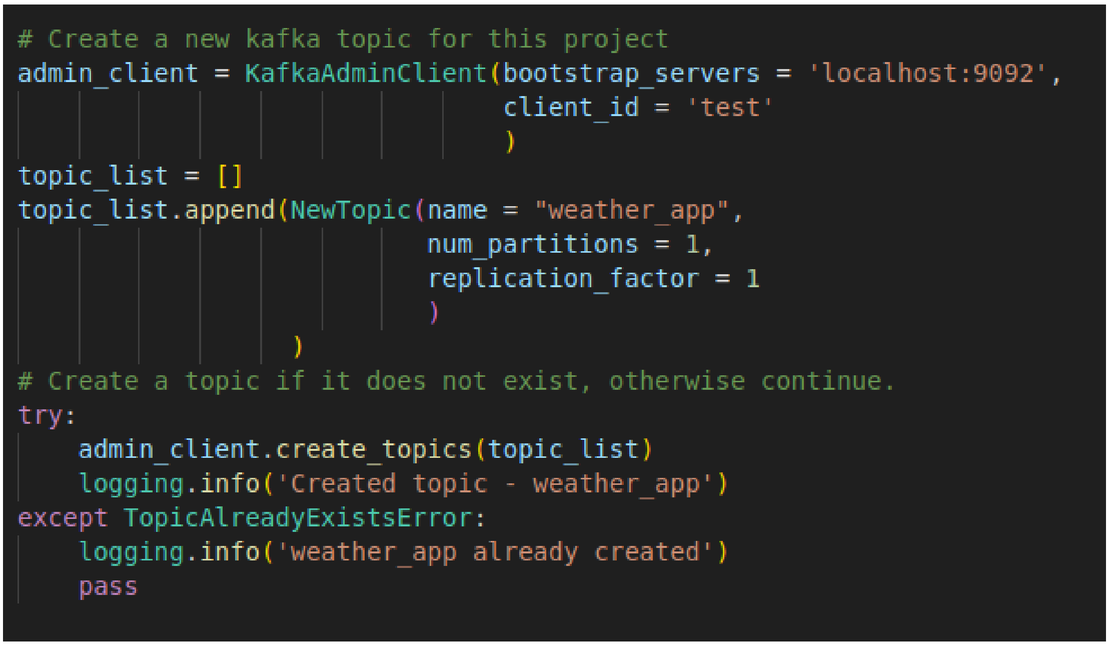


```
ay@ay-virtual-machine:~$ pip install kafka
Defaulting to user installation because normal site-packages is not writeable
Collecting kafka
  Downloading kafka-1.3.5-py2.py3-none-any.whl (207 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 207.2/207.2 KB 2.5 MB/s eta 0:00:00
Installing collected packages: kafka
Successfully installed kafka-1.3.5
ay@ay-virtual-machine:~$ pip uninstall kafka
Found existing installation: kafka 1.3.5
Uninstalling kafka-1.3.5:
  Would remove:
    /home/ay/.local/lib/python3.10/site-packages/kafka-1.3.5.dist-info/*
    /home/ay/.local/lib/python3.10/site-packages/kafka/*
Proceed (Y/n)? y
  Successfully uninstalled kafka-1.3.5
ay@ay-virtual-machine:~$ pip install kafka-python
Defaulting to user installation because normal site-packages is not writeable
Collecting kafka-python
  Downloading kafka_python-2.0.2-py2.py3-none-any.whl (246 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 246.5/246.5 KB 1.5 MB/s eta 0:00:00
Installing collected packages: kafka-python
Successfully installed kafka-python-2.0.2
```

1.0 Producer Script

1.1 Topic Creation

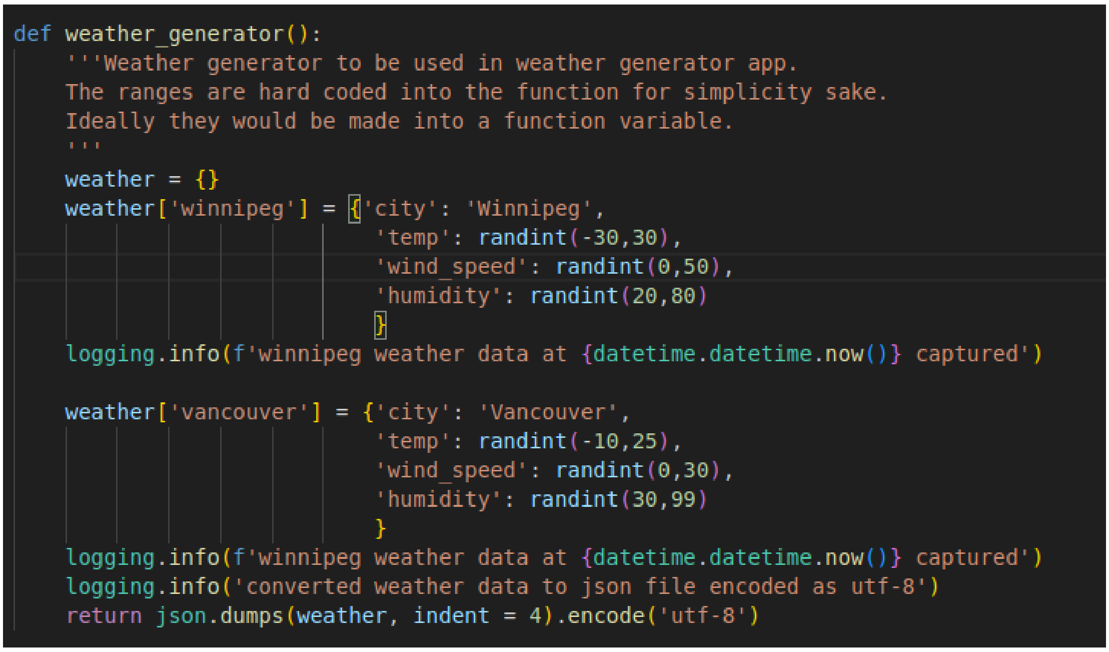
After defining log configurations, a Kafka admin_client was created to help create a new kafka topic called weather_app. Exception handling was used to capture and handle likely error of "topic already exists" if the code is executed multiple times. This method of topic definition was preferred so that I can explore topic creation from python script. Topic creation from terminal had previously been explored in a class exercise. This method was also convenient as it meant that all code was contained in this script. If the topic were to be deleted for example, I can still run this script without errors.



```python
# Create a new kafka topic for this project
admin_client = KafkaAdminClient(bootstrap_servers = 'localhost:9092',
                                client_id = 'test'
                                )
topic_list = []
topic_list.append(NewTopic(name = "weather_app",
                           num_partitions = 1,
                           replication_factor = 1
                           )
                  )
# Create a topic if it does not exist, otherwise continue.
try:
    admin_client.create_topics(topic_list)
    logging.info('Created topic - weather_app')
except TopicAlreadyExistsError:
    logging.info('weather_app already created')
    pass
```

1.2 Generator Function

A function was defined which simply uses the randint function to generate random weather data within given ranges. randint was preffered for this tax because in reality, weather data is reported as integers, not as float. This project was developed to mimic real world scenarios as closely as possible. The function generates data for Winnipeg and then for Vancouver. It returns a json data object containing the weather data. The json object is utf-8 encoded.

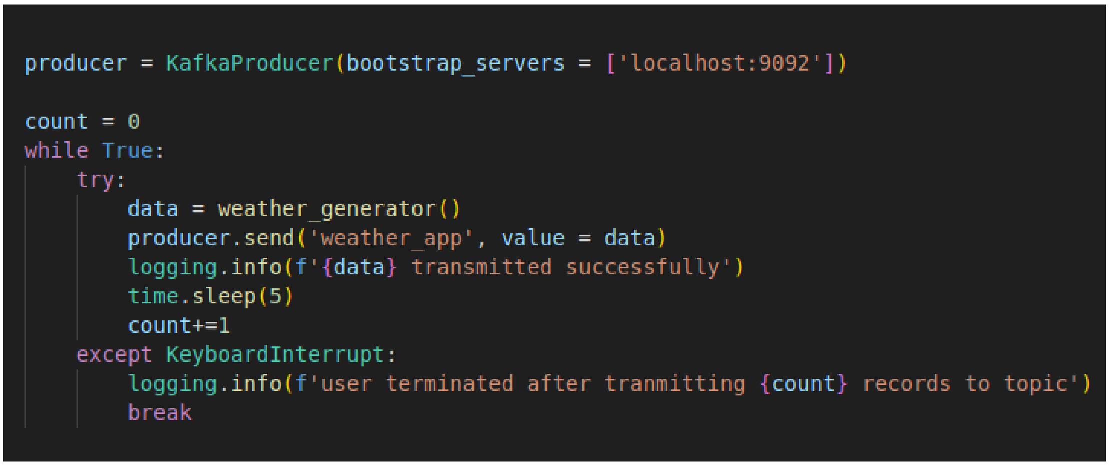


```
def weather_generator():
    '''Weather generator to be used in weather generator app.
    The ranges are hard coded into the function for simplicity sake.
    Ideally they would be made into a function variable.
    '''
    weather = {}
    weather['winnipeg'] = {'city': 'Winnipeg',
                            'temp': randint(-30,30),
                            'wind_speed': randint(0,50),
                            'humidity': randint(20,80)
                            }
    logging.info(f'winnipeg weather data at {datetime.datetime.now()} captured')

    weather['vancouver'] = {'city': 'Vancouver',
                            'temp': randint(-10,25),
                            'wind_speed': randint(0,30),
                            'humidity': randint(30,99)
                            }
    logging.info(f'winnipeg weather data at {datetime.datetime.now()} captured')
    logging.info('converted weather data to json file encoded as utf-8')
    return json.dumps(weather, indent = 4).encode('utf-8')
```

1.3 Data Transmission

The kafka producer was then instantiated and used to .send the weather data to the weather_app topic. This was accomplished by using a while loop to call the weather_generator function. It was set to a sleep time of 5 seconds, so that it generates and transmits new data every 5 seconds. Since an infinite loop was used, an exception handling rule was set to capture the keyboard interrupt error from manually terminating the process. It simply logs the number of successful records transmitted before interruption, and then breaks without crashing.

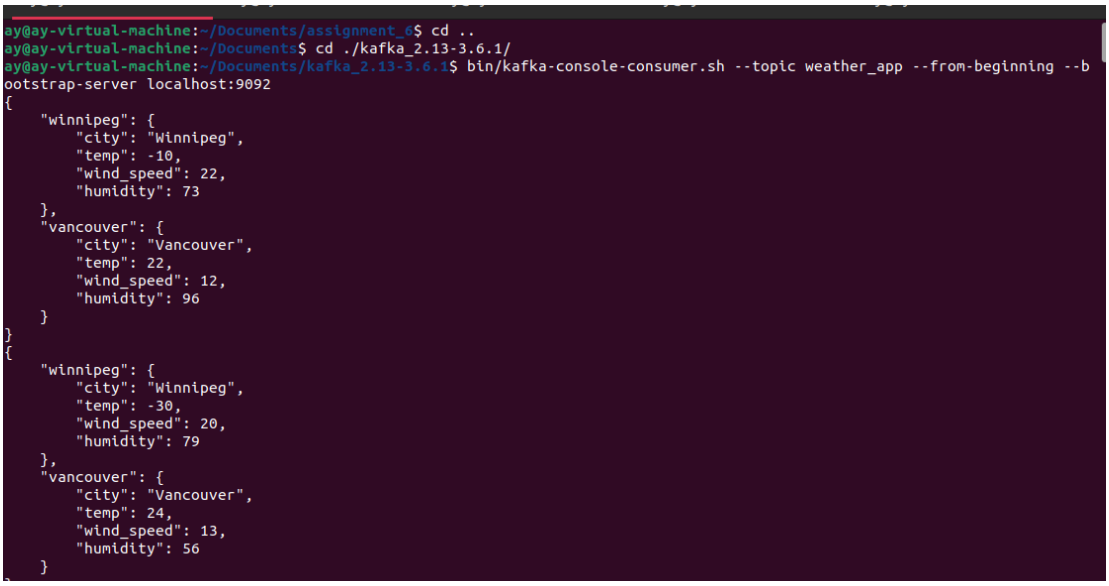


```
producer = KafkaProducer(bootstrap_servers = ['localhost:9092'])

count = 0
while True:
    try:
        data = weather_generator()
        producer.send('weather_app', value = data)
        logging.info(f'{data} transmitted successfully')
        time.sleep(5)
        count+=1
    except KeyboardInterrupt:
        logging.info(f'user terminated after tranmitting {count} records to topic')
        break
```

1.4 Sample Json Tranmitted

To test the successful transmission of the data, a consumer instance was set in the terminal to listen to the json data produced. Below is a screenshot of what some of the data looked like.



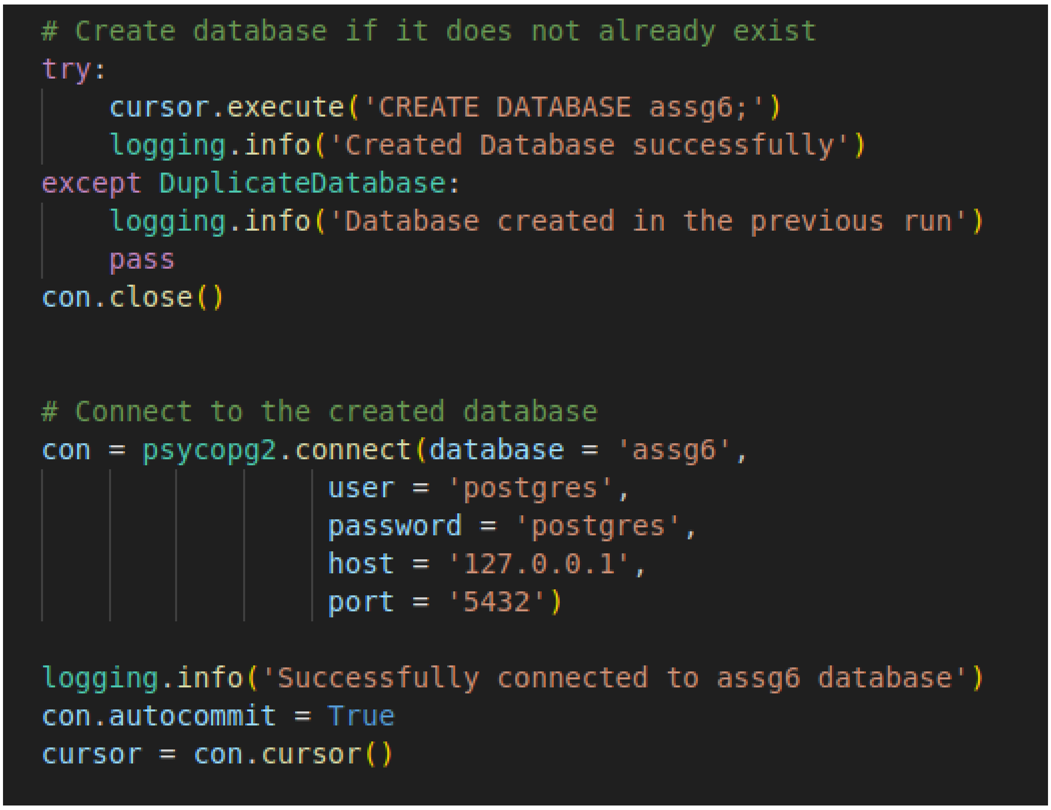
```
ay@ay-virtual-machine:~/Documents/assignment_6$ cd ..
ay@ay-virtual-machine:~/Documents$ cd ./kafka_2.13-3.6.1/
ay@ay-virtual-machine:~/Documents/kafka_2.13-3.6.1$ bin/kafka-console-consumer.sh --topic weather_app --from-beginning --b
ootstrap-server localhost:9092
{
    "winnipeg": {
        "city": "Winnipeg",
        "temp": -10,
        "wind_speed": 22,
        "humidity": 73
    },
    "vancouver": {
        "city": "Vancouver",
        "temp": 22,
        "wind_speed": 12,
        "humidity": 96
    }
}
{
    "winnipeg": {
        "city": "Winnipeg",
        "temp": -30,
        "wind_speed": 20,
        "humidity": 79
    },
    "vancouver": {
        "city": "Vancouver",
        "temp": 24,
        "wind_speed": 13,
        "humidity": 56
    }
```

2.0 Consumer Script

Having successfully executed and tested the producer script, the consumer script was written to fetch every instance of the streaming data from the topic to a postgres database in realtime.

2.1 Database Creation and Connection

Again, after necessary logging definitions, a connection was instantiated to the postgres server. A database was created therein, and then a connection was set to that database. For convenience, autocommit was set to True.



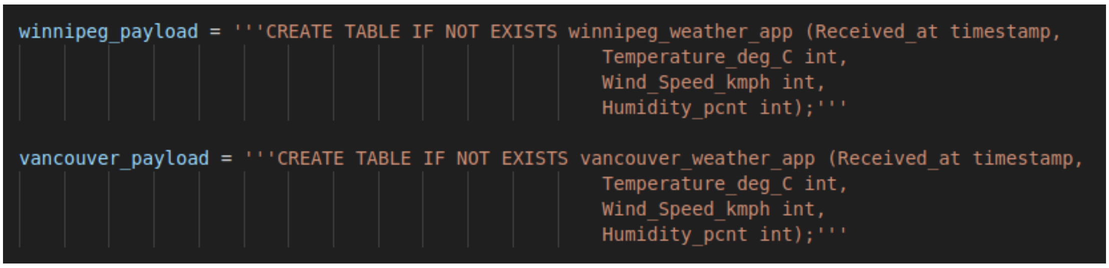
```python
# Create database if it does not already exist
try:
    cursor.execute('CREATE DATABASE assg6;')
    logging.info('Created Database successfully')
except DuplicateDatabase:
    logging.info('Database created in the previous run')
    pass
con.close()


# Connect to the created database
con = psycopg2.connect(database = 'assg6',
                        user = 'postgres',
                        password = 'postgres',
                        host = '127.0.0.1',
                        port = '5432')

logging.info('Successfully connected to assg6 database')
con.autocommit = True
cursor = con.cursor()
```

2.2 Table Creation

Two tables were created for this project. While the project requirement seemed to be for one table containing the weather information for both cities, I preferred to use two tables because that appeared more practical. In real world situations, seperate tables would contain weather information for seperate cities. With a mindset of real world applicability, the table for winnipeg weather data was created seperately from the table for vancouver weather data. Furthermore, streaming into two different tables provided an opportunity to test this capability in kafka.

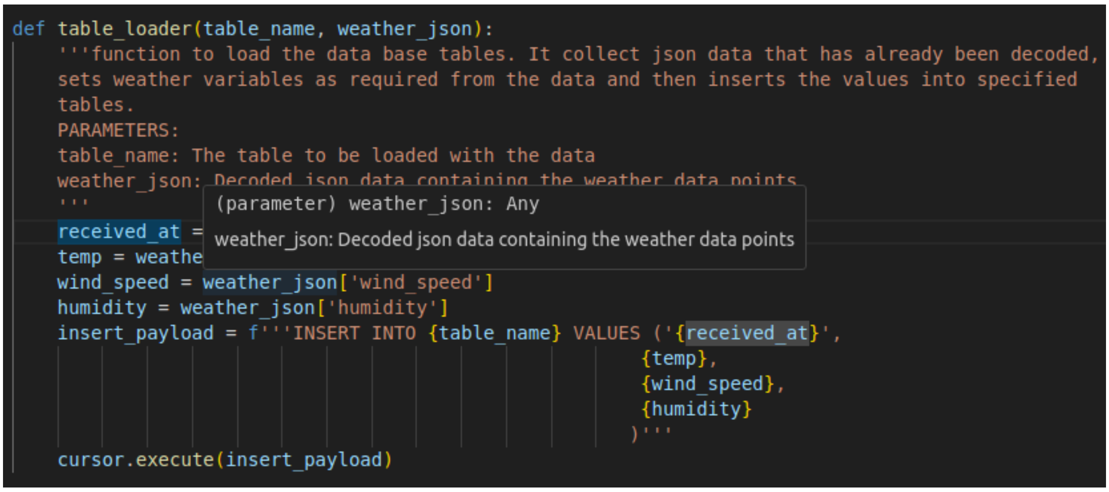


```
winnipeg_payload = '''CREATE TABLE IF NOT EXISTS winnipeg_weather_app (Received_at timestamp,
                                                    Temperature_deg_C int,
                                                    Wind_Speed_kmph int,
                                                    Humidity_pcnt int);'''

vancouver_payload = '''CREATE TABLE IF NOT EXISTS vancouver_weather_app (Received_at timestamp,
                                                    Temperature_deg_C int,
                                                    Wind_Speed_kmph int,
                                                    Humidity_pcnt int);'''
```

2.3 Table Loader Function

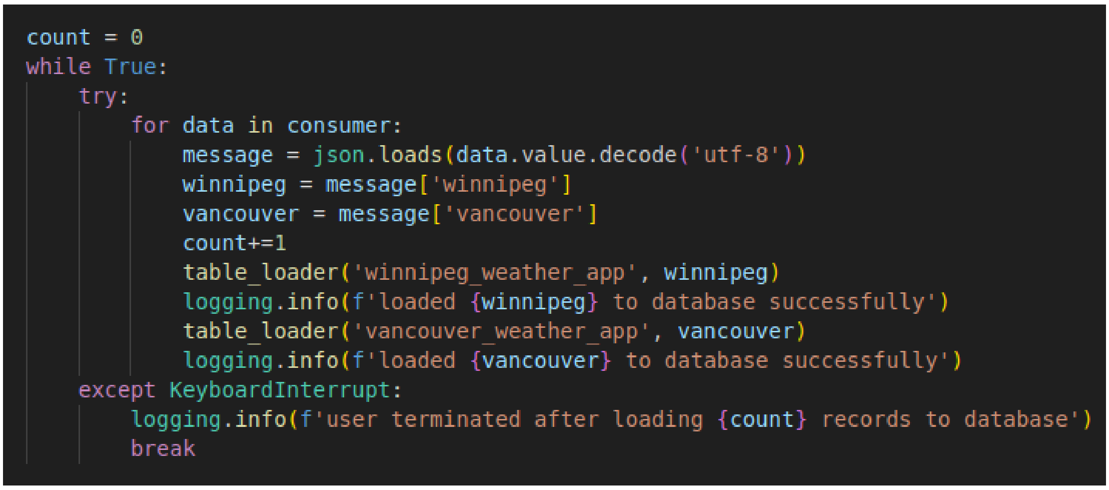
A function was created to load the tables. It takes the table name and the json data as variables. The function seperates the json data into its constituent variables and then sets those as values to be inserted into the named table. This function does not return anything but simply executes an sql payload.



```
def table_loader(table_name, weather_json):
    '''function to load the data base tables. It collect json data that has already been decoded,
    sets weather variables as required from the data and then inserts the values into specified
    tables.
    PARAMETERS:
    table_name: The table to be loaded with the data
    weather_json: Decoded json data containing the weather data points
    '''        (parameter) weather_json: Any
    received_at =
                   weather_json: Decoded json data containing the weather data points
    temp = weathe
    wind_speed = weather_json['wind_speed']
    humidity = weather_json['humidity']
    insert_payload = f'''INSERT INTO {table_name} VALUES ('{received_at}',
                                                    {temp},
                                                    {wind_speed},
                                                    {humidity}
                                                   )'''
    cursor.execute(insert_payload)
```

2.4 Data Loading

The actual data loading is accomplished using the trio of a while loop, a for loop, and a try-except exception handling. After the kafka consumer is intantiated, a for loop extract the value and the timestamp from the contents of this consumer. The timestamp from the consumer is used because it represents the true time the data was received into the topic. Using datetime.now() will return another timeframe entirely (time when the data was loaded to the database). The timestamp referenced in the assignment instructions seemed to relate more with the time the data was received by the consumer. TThe formatting of this timestamp to datetime string had been previously setup in the table loader funciton. The message is decoded and the data for the respective cities is extracted and saved to their variables. The table loader function is called for each city and then a keyboard interrupt exception handler is put in place to prevent crashing when the while loop is manually terminated. The count of successful records loaded to database is logged.

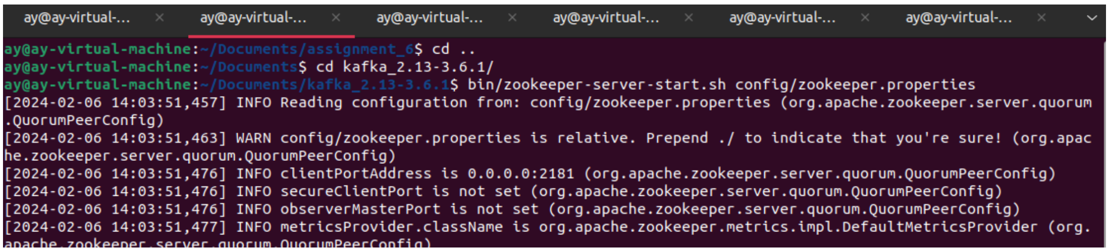


```
count = 0
while True:
    try:
        for data in consumer:
            message = json.loads(data.value.decode('utf-8'))
            winnipeg = message['winnipeg']
            vancouver = message['vancouver']
            count+=1
            table_loader('winnipeg_weather_app', winnipeg)
            logging.info(f'loaded {winnipeg} to database successfully')
            table_loader('vancouver_weather_app', vancouver)
            logging.info(f'loaded {vancouver} to database successfully')
    except KeyboardInterrupt:
        logging.info(f'user terminated after loading {count} records to database')
        break
```

3.0 Script Execution from Terminal

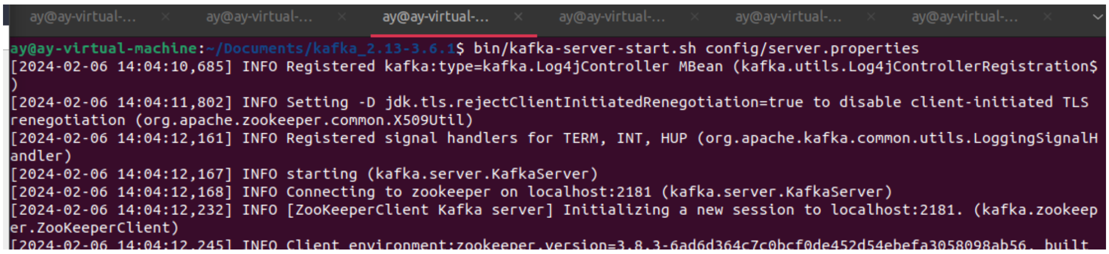
To complete the project, the script is executed from terminal. First the zookeeper and the kafka broker services were started. While these were running, a new terminal was used to run the producer script. Finally the consumer script was executed in yet another terminal. The log folder and the corresponding files were thus created. The database and the required tables were also created. The producer and consumer scripts were allowed to run for about ten minutes before terminating. This resulted in the loading of 134 records in each of Winnipeg_weather_app and vancouver_weather_app tables. The following screenshots shows these steps in succession. A screenshot of some of the table contents is also presented below.

3.1 Starting the Zookeeper Service



```
ay@ay-virtual-machine:~/Documents/assignment_6$ cd ..
ay@ay-virtual-machine:~/Documents$ cd kafka_2.13-3.6.1/
ay@ay-virtual-machine:~/Documents/kafka_2.13-3.6.1$ bin/zookeeper-server-start.sh config/zookeeper.properties
[2024-02-06 14:03:51,457] INFO Reading configuration from: config/zookeeper.properties (org.apache.zookeeper.server.quorum
.QuorumPeerConfig)
[2024-02-06 14:03:51,463] WARN config/zookeeper.properties is relative. Prepend ./ to indicate that you're sure! (org.apac
he.zookeeper.server.quorum.QuorumPeerConfig)
[2024-02-06 14:03:51,476] INFO clientPortAddress is 0.0.0.0:2181 (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-02-06 14:03:51,476] INFO secureClientPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-02-06 14:03:51,476] INFO observerMasterPort is not set (org.apache.zookeeper.server.quorum.QuorumPeerConfig)
[2024-02-06 14:03:51,477] INFO metricsProvider.className is org.apache.zookeeper.metrics.impl.DefaultMetricsProvider (org.
apache.zookeeper.server.quorum.QuorumPeerConfig)
```

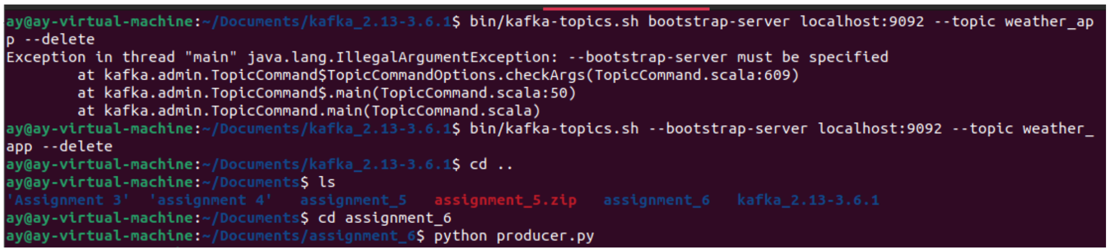
3.2 Starting the Kafka service



```
ay@ay-virtual-machine:~/Documents/kafka_2.13-3.6.1$ bin/kafka-server-start.sh config/server.properties
[2024-02-06 14:04:10,685] INFO Registered kafka:type=kafka.Log4jController MBean (kafka.utils.Log4jControllerRegistration$
)
[2024-02-06 14:04:11,802] INFO Setting -D jdk.tls.rejectClientInitiatedRenegotiation=true to disable client-initiated TLS
renegotiation (org.apache.zookeeper.common.X509Util)
[2024-02-06 14:04:12,161] INFO Registered signal handlers for TERM, INT, HUP (org.apache.kafka.common.utils.LoggingSignalH
andler)
[2024-02-06 14:04:12,167] INFO starting (kafka.server.KafkaServer)
[2024-02-06 14:04:12,168] INFO Connecting to zookeeper on localhost:2181 (kafka.server.KafkaServer)
[2024-02-06 14:04:12,232] INFO [ZooKeeperClient Kafka server] Initializing a new session to localhost:2181. (kafka.zookeep
er.ZooKeeperClient)
[2024-02-06 14:04:12,245] INFO Client environment:zookeeper.version=3.8.3-6ad6d364c7c0bcf0de452d54ebefa3058098ab56, built
```

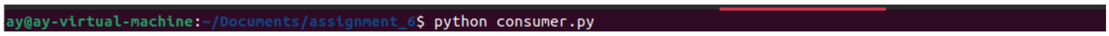
3.3 Running the Producer.py

The weather_app topic was first deleted. This is because it contained all the data that had been used to test the scripts. This was not a problem since the script itself is configured to re-create the topic if it doesnt exist.



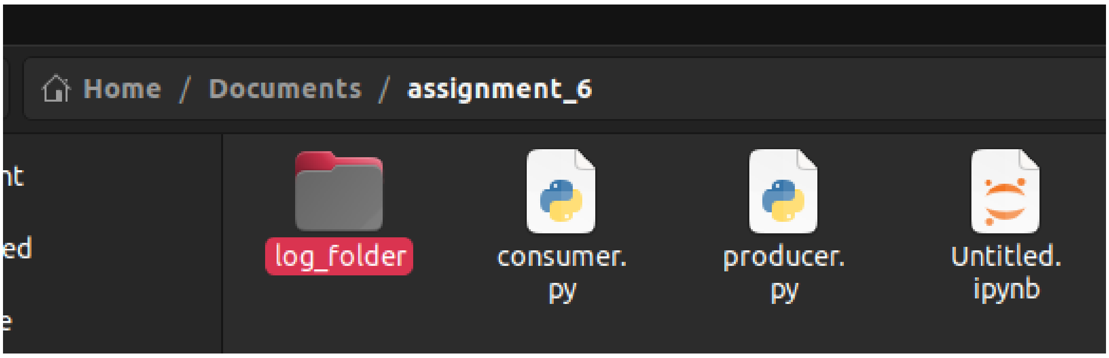
```
ay@ay-virtual-machine:~/Documents/kafka_2.13-3.6.1$ bin/kafka-topics.sh bootstrap-server localhost:9092 --topic weather_ap
p --delete
Exception in thread "main" java.lang.IllegalArgumentException: --bootstrap-server must be specified
        at kafka.admin.TopicCommand$TopicCommandOptions.checkArgs(TopicCommand.scala:609)
        at kafka.admin.TopicCommand$.main(TopicCommand.scala:50)
        at kafka.admin.TopicCommand.main(TopicCommand.scala)
ay@ay-virtual-machine:~/Documents/kafka_2.13-3.6.1$ bin/kafka-topics.sh --bootstrap-server localhost:9092 --topic weather_
app --delete
ay@ay-virtual-machine:~/Documents/kafka_2.13-3.6.1$ cd ..
ay@ay-virtual-machine:~/Documents$ ls
'Assignment 3'  'assignment 4'   assignment_5   assignment_5.zip   assignment_6   kafka_2.13-3.6.1
ay@ay-virtual-machine:~/Documents$ cd assignment_6
ay@ay-virtual-machine:~/Documents/assignment_6$ python producer.py
```

3.4 Running the Consumer.py

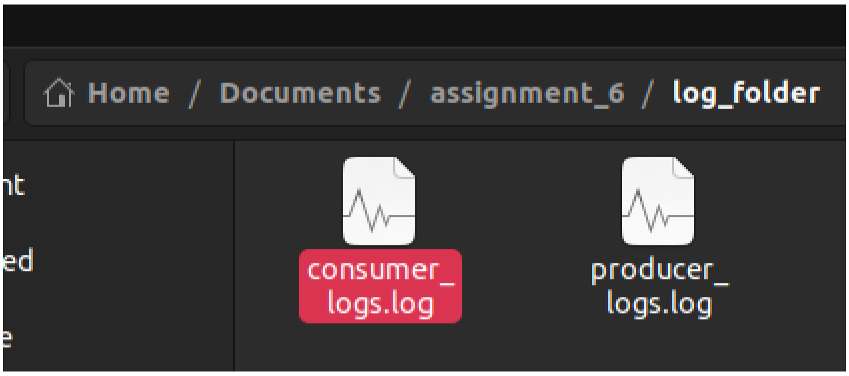


```
ay@ay-virtual-machine:~/Documents/assignment_6$ python consumer.py
```

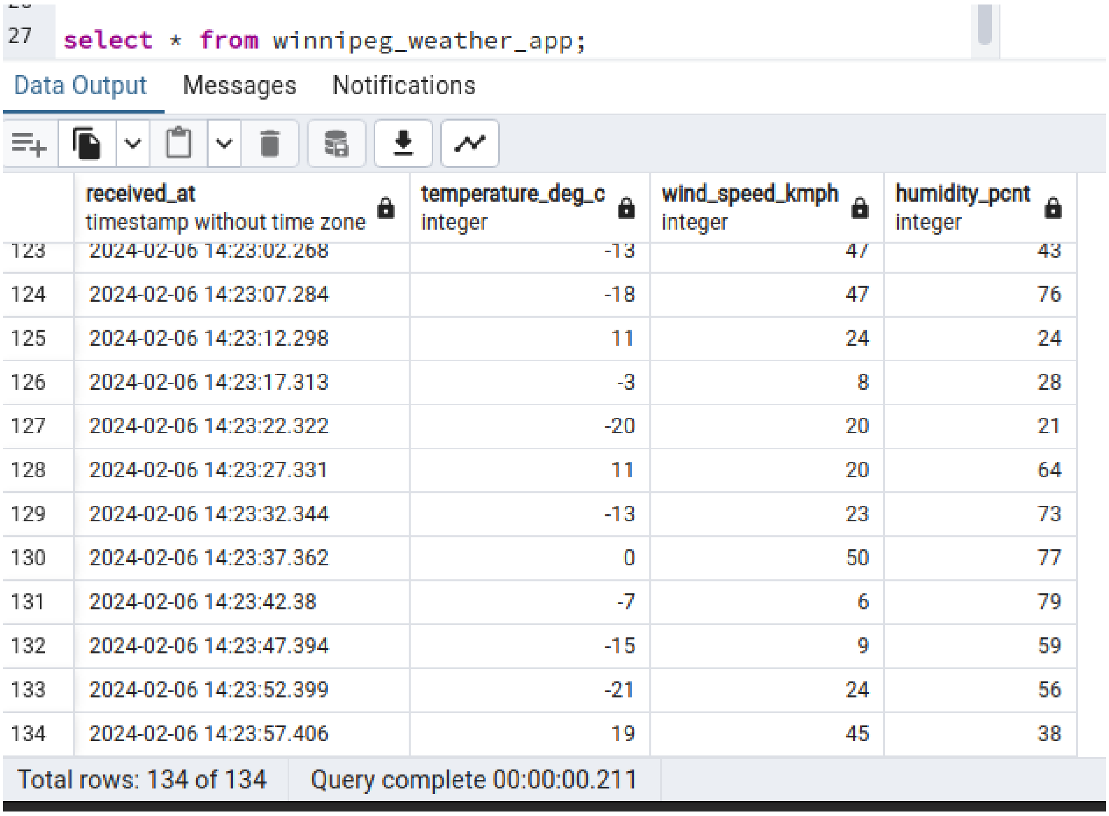
3.4 Log Folder Created



3.5 Log Files Created



3.6 Winnipeg Database Table

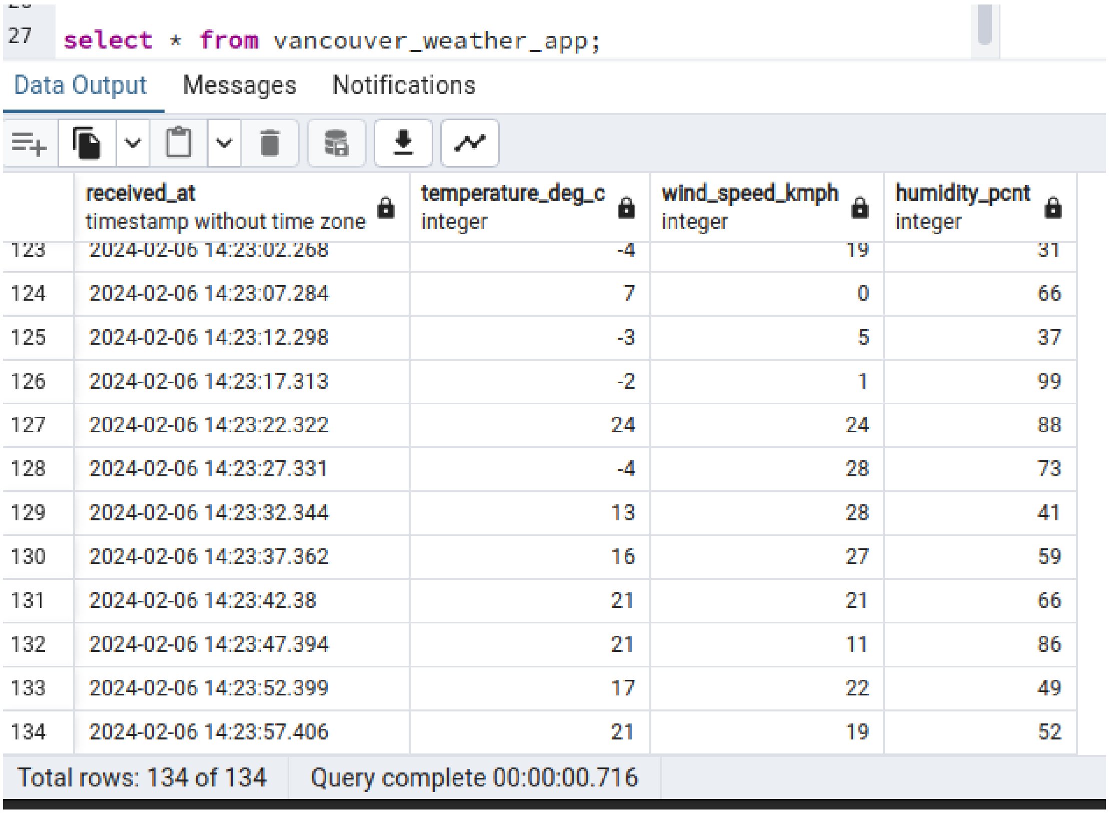


```
select * from winnipeg_weather_app;
```

	received_at timestamp without time zone	temperature_deg_c integer	wind_speed_kmph integer	humidity_pcnt integer
123	2024-02-06 14:23:02.268	-13	47	43
124	2024-02-06 14:23:07.284	-18	47	76
125	2024-02-06 14:23:12.298	11	24	24
126	2024-02-06 14:23:17.313	-3	8	28
127	2024-02-06 14:23:22.322	-20	20	21
128	2024-02-06 14:23:27.331	11	20	64
129	2024-02-06 14:23:32.344	-13	23	73
130	2024-02-06 14:23:37.362	0	50	77
131	2024-02-06 14:23:42.38	-7	6	79
132	2024-02-06 14:23:47.394	-15	9	59
133	2024-02-06 14:23:52.399	-21	24	56
134	2024-02-06 14:23:57.406	19	45	38

Total rows: 134 of 134 Query complete 00:00:00.211

3.7 Vancouver Database Table

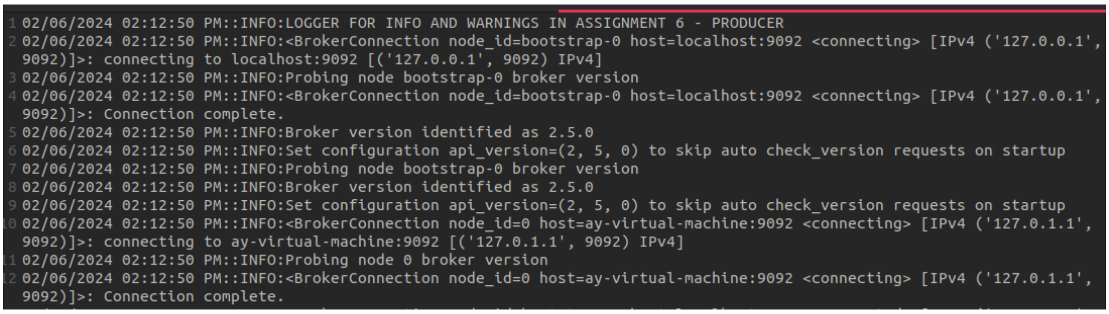


```
select * from vancouver_weather_app;
```

	received_at timestamp without time zone	temperature_deg_c integer	wind_speed_kmph integer	humidity_pcnt integer
123	2024-02-06 14:23:02.268	-4	19	31
124	2024-02-06 14:23:07.284	7	0	66
125	2024-02-06 14:23:12.298	-3	5	37
126	2024-02-06 14:23:17.313	-2	1	99
127	2024-02-06 14:23:22.322	24	24	88
128	2024-02-06 14:23:27.331	-4	28	73
129	2024-02-06 14:23:32.344	13	28	41
130	2024-02-06 14:23:37.362	16	27	59
131	2024-02-06 14:23:42.38	21	21	66
132	2024-02-06 14:23:47.394	21	11	86
133	2024-02-06 14:23:52.399	17	22	49
134	2024-02-06 14:23:57.406	21	19	52

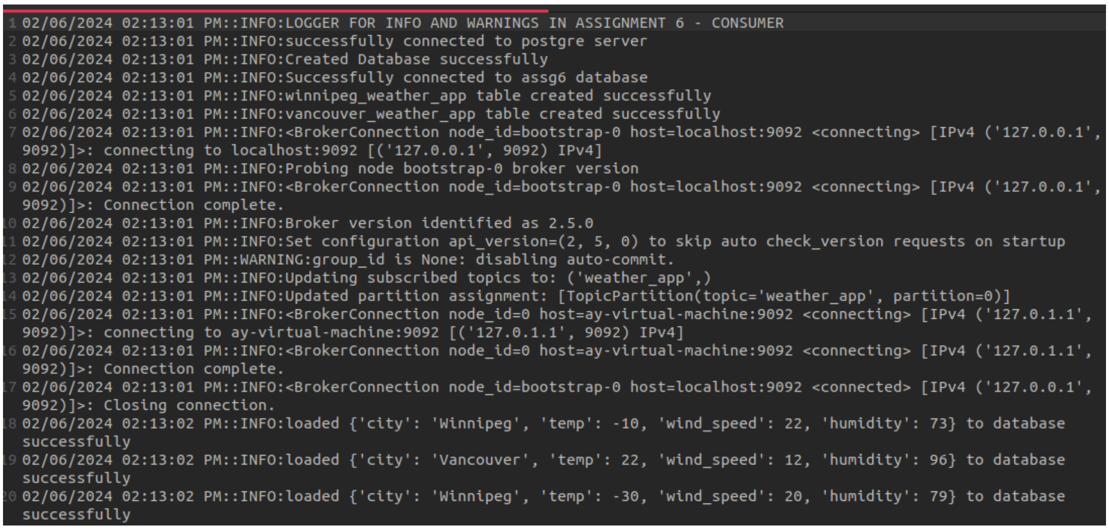
Total rows: 134 of 134 Query complete 00:00:00.716

3.8 Sample Producer Logs



```
02/06/2024 02:12:50 PM::INFO:LOGGER FOR INFO AND WARNINGS IN ASSIGNMENT 6 - PRODUCER
02/06/2024 02:12:50 PM::INFO:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv4 ('127.0.0.1',
9092)]>: connecting to localhost:9092 [('127.0.0.1', 9092) IPv4]
02/06/2024 02:12:50 PM::INFO:Probing node bootstrap-0 broker version
02/06/2024 02:12:50 PM::INFO:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv4 ('127.0.0.1',
9092)]>: Connection complete.
02/06/2024 02:12:50 PM::INFO:Broker version identified as 2.5.0
02/06/2024 02:12:50 PM::INFO:Set configuration api_version=(2, 5, 0) to skip auto check_version requests on startup
02/06/2024 02:12:50 PM::INFO:Probing node bootstrap-0 broker version
02/06/2024 02:12:50 PM::INFO:Broker version identified as 2.5.0
02/06/2024 02:12:50 PM::INFO:Set configuration api_version=(2, 5, 0) to skip auto check_version requests on startup
02/06/2024 02:12:50 PM::INFO:<BrokerConnection node_id=0 host=ay-virtual-machine:9092 <connecting> [IPv4 ('127.0.1.1',
9092)]>: connecting to ay-virtual-machine:9092 [('127.0.1.1', 9092) IPv4]
02/06/2024 02:12:50 PM::INFO:Probing node 0 broker version
02/06/2024 02:12:50 PM::INFO:<BrokerConnection node_id=0 host=ay-virtual-machine:9092 <connecting> [IPv4 ('127.0.1.1',
9092)]>: Connection complete.
```

3.9 Sample Consumer Logs



```
02/06/2024 02:13:01 PM::INFO:LOGGER FOR INFO AND WARNINGS IN ASSIGNMENT 6 - CONSUMER
02/06/2024 02:13:01 PM::INFO:successfully connected to postgre server
02/06/2024 02:13:01 PM::INFO:Created Database successfully
02/06/2024 02:13:01 PM::INFO:Successfully connected to assg6 database
02/06/2024 02:13:01 PM::INFO:winnipeg_weather_app table created successfully
02/06/2024 02:13:01 PM::INFO:vancouver_weather_app table created successfully
02/06/2024 02:13:01 PM::INFO:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv4 ('127.0.0.1',
9092)]>: connecting to localhost:9092 [('127.0.0.1', 9092) IPv4]
02/06/2024 02:13:01 PM::INFO:Probing node bootstrap-0 broker version
02/06/2024 02:13:01 PM::INFO:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connecting> [IPv4 ('127.0.0.1',
9092)]>: Connection complete.
02/06/2024 02:13:01 PM::INFO:Broker version identified as 2.5.0
02/06/2024 02:13:01 PM::INFO:Set configuration api_version=(2, 5, 0) to skip auto check_version requests on startup
02/06/2024 02:13:01 PM::WARNING:group_id is None: disabling auto-commit.
02/06/2024 02:13:01 PM::INFO:Updating subscribed topics to: ('weather_app',)
02/06/2024 02:13:01 PM::INFO:Updated partition assignment: [TopicPartition(topic='weather_app', partition=0)]
02/06/2024 02:13:01 PM::INFO:<BrokerConnection node_id=0 host=ay-virtual-machine:9092 <connecting> [IPv4 ('127.0.1.1',
9092)]>: connecting to ay-virtual-machine:9092 [('127.0.1.1', 9092) IPv4]
02/06/2024 02:13:01 PM::INFO:<BrokerConnection node_id=0 host=ay-virtual-machine:9092 <connecting> [IPv4 ('127.0.1.1',
9092)]>: Connection complete.
02/06/2024 02:13:01 PM::INFO:<BrokerConnection node_id=bootstrap-0 host=localhost:9092 <connected> [IPv4 ('127.0.0.1',
9092)]>: Closing connection.
02/06/2024 02:13:02 PM::INFO:loaded {'city': 'Winnipeg', 'temp': -10, 'wind_speed': 22, 'humidity': 73} to database
successfully
02/06/2024 02:13:02 PM::INFO:loaded {'city': 'Vancouver', 'temp': 22, 'wind_speed': 12, 'humidity': 96} to database
successfully
02/06/2024 02:13:02 PM::INFO:loaded {'city': 'Winnipeg', 'temp': -30, 'wind_speed': 20, 'humidity': 79} to database
successfully
```

4.0 Conclusion

The procedure was streaming data from the source to a destination was demonstrated in this project using kafka. Two different tables were loaded with the data streams in real time by setting up a consumer script. The data generate from a producer script was transmitted to a kafka topic from where the consumer script picks it. At the end of the project, 134 records were loaded into each table. The script and logs for this project are submitted along with this report in a zipped folder.