

Assignment 3

BigQuery Part

Feature Analysis

- Query

```
#Feature Analysis
select * from `bigquery-public-data.epa_historical_air_quality.no2_daily_summary` limit 10;
select distinct(county_name), city_name,parameter_name
  from `bigquery-public-data.epa_historical_air_quality.no2_daily_summary`
  where parameter_name='Nitrogen dioxide (NO2)'
  limit 10;
select distinct(county_code)
  from `bigquery-public-data.epa_historical_air_quality.no2_daily_summary`
  limit 10;
select state_name, first_max_hour,aqi
  from `bigquery-public-data.epa_historical_air_quality.no2_daily_summary`
  order by aqi desc limit 20;
```

- Results



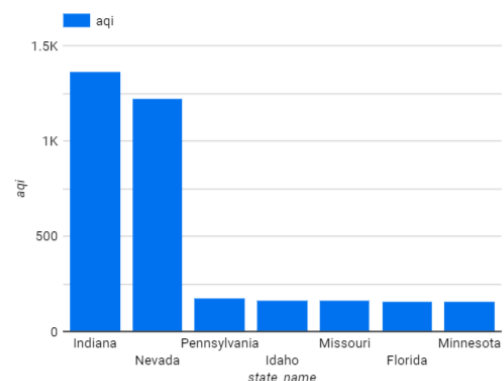
Query results

[SAVE RESULTS](#)

	JOB INFORMATION	RESULTS	JSON	EXECUTION DETAILS
Row	county_name	city_name	parameter_name	
1	Mercer	Beulah	Nitrogen dioxide (NO2)	
2	Spencer	Not in a city	Nitrogen dioxide (NO2)	
3	Greenbrier	Not in a city	Nitrogen dioxide (NO2)	
4	York	York	Nitrogen dioxide (NO2)	

- Visualization

state_name	aqi
1. Indiana	1,366
2. Nevada	1,226
3. Pennsylvania	174
4. Idaho	167
5. Missouri	166
6. Florida	161
7. Minnesota	157



Data Cleaning

- Query

```
#data cleaning
select count(city_name)
  from `bigquery-public-data.epa_historical_air_quality.no2_daily_summary`
  where event_type is not null;
select distinct(county_name),city_name
  from `bigquery-public-data.epa_historical_air_quality.no2_daily_summary`
  where local_site_name is not null;
```

- Results

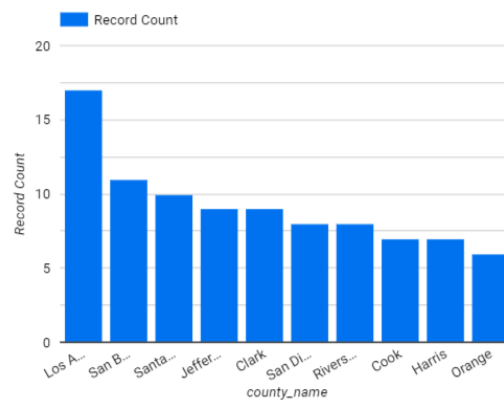
← Query results

JOB INFORMATION		RESULTS	JSON	EXECUTION DETAILS
Row	state_name	first_max_hour	aqi	
1	Nevada	3	208	
2	Indiana	11	204	
3	Nevada	2	196	
4	Indiana	13	182	

- Visualization

county_name		Record Count ▾
1.	Los Angeles	17
2.	San Bernardino	11
3.	Santa Barbara	10
4.	Clark	9
5.	Jefferson	9
6.	Riverside	8
7.	San Diego	8
8.	Harris	7
9.	Cook	7
10.	Essex	6
11.	Hillsborough	6

1 - 50 / 371 < >



Machine Learning (logistic regression)

- Creating View

```
CREATE OR REPLACE VIEW
  `nosummary.input_view` AS
SELECT
  observation_count,
  observation_percent,
  arithmetic_mean,
  aqi,
  CASE
    WHEN MOD(first_max_hour, 10) < 8 THEN 'training'
    WHEN MOD(first_max_hour, 10) = 8 THEN 'evaluation'
    WHEN MOD(first_max_hour, 10) = 9 THEN 'prediction'
  END AS dataframe
FROM
  `bigquery-public-data.epa_historical_air_quality.no2_daily_summary`
```

- Training ML Model

```
CREATE OR REPLACE MODEL
  `nosummary.aqi_model`
OPTIONS
  ( model_type='LOGISTIC_REG',
    auto_class_weights=TRUE,
    input_label_cols=['aqi']
  ) AS
SELECT
  * EXCEPT(dataframe)
FROM
  `nosummary.input_view`
WHERE
  dataframe = 'training'
```

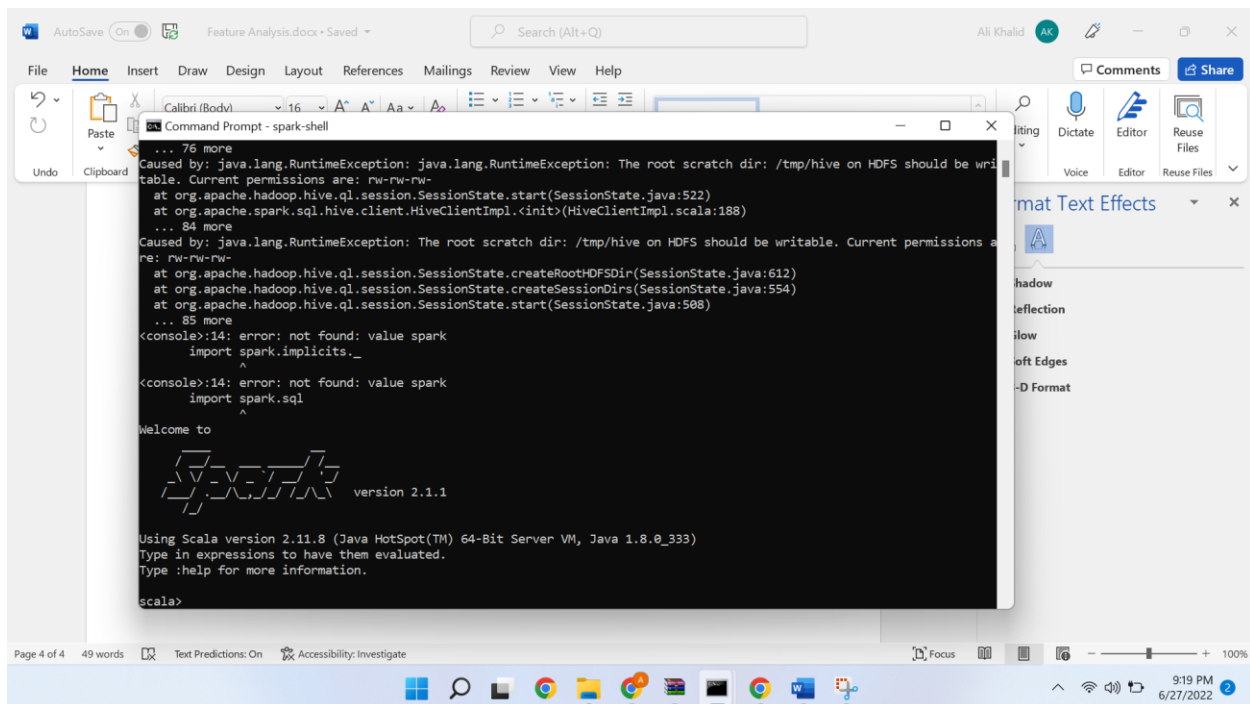
- Model Evaluation results

aqi_model

DETAILS	TRAINING	EVALUATION	SCHEMA
Mean absolute error	4.6807		
Mean squared error	39.9174		
Mean squared log error	0.178		
Median absolute error	3.7544		
R squared	0.8348		

Spark Part

Unable to run spark on PC. Spark-shell command is running, and Spark Context is initialized but it is giving some other error which I wasn't able to resolve.

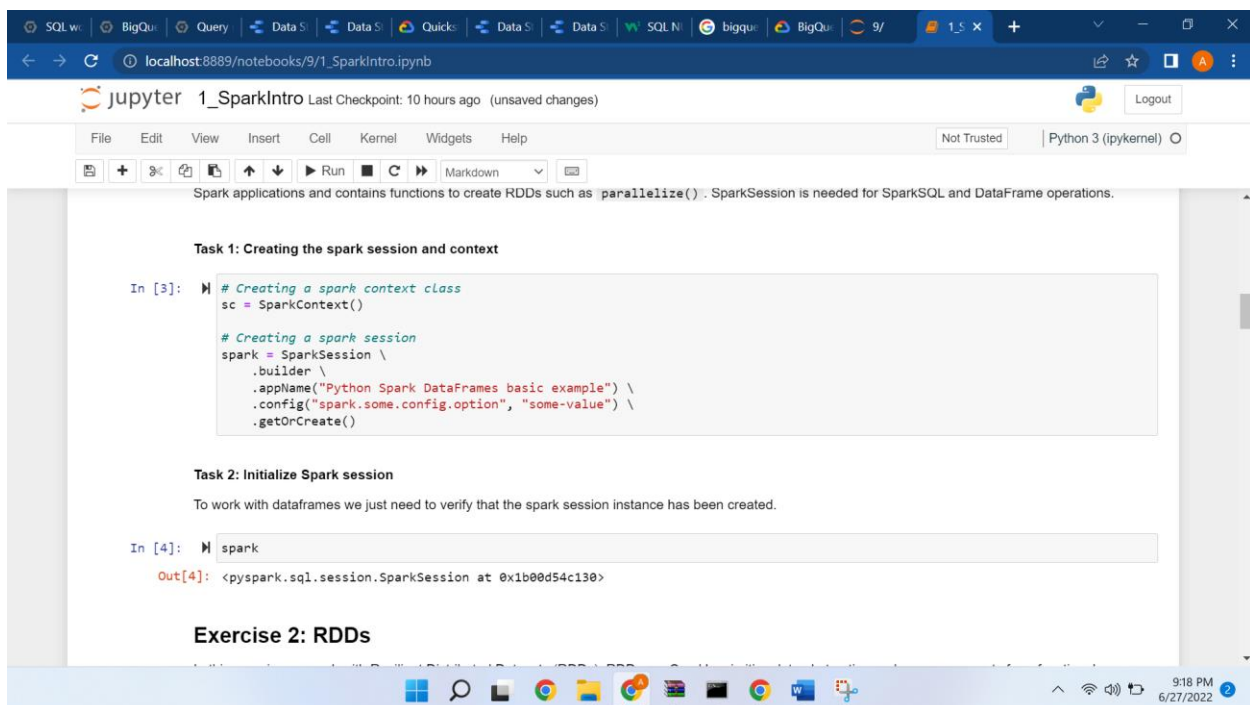


```
... 76 more
Caused by: java.lang.RuntimeException: java.lang.RuntimeException: The root scratch dir: /tmp/hive on HDFS should be writable. Current permissions are: rw-rw-rw-
at org.apache.hadoop.hive.q1.session.SessionState.start(SessionState.java:522)
at org.apache.spark.sql.hive.client.HiveClientImpl.<init>(HiveClientImpl.scala:188)
... 84 more
Caused by: java.lang.RuntimeException: The root scratch dir: /tmp/hive on HDFS should be writable. Current permissions are: rw-rw-rw-
at org.apache.hadoop.hive.q1.session.SessionState.createRootHDFSDir(SessionState.java:612)
at org.apache.hadoop.hive.q1.session.SessionState.createSessionDirs(SessionState.java:554)
at org.apache.hadoop.hive.q1.session.SessionState.start(SessionState.java:588)
... 85 more
<console>:14: error: not found: value spark
import spark.implicits._
^
<console>:14: error: not found: value spark
import spark.sql
^

Welcome to
Spark version 2.1.1

Using Scala version 2.11.8 (Java HotSpot(TM) 64-Bit Server VM, Java 1.8.0_333)
Type in expressions to have them evaluated.
Type :help for more information.

scala>
```



```
SQL wi | BigQu | Query | Data S | Data S | Quicks | Data S | Data S | SQL N | bigqu | BigQu | 9/ | 1.5 x | +
localhost:8889/notebooks/9/1_SparkIntro.ipynb
jupyter 1_SparkIntro Last Checkpoint: 10 hours ago (unsaved changes) Logout
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pykernel)
Spark applications and contains functions to create RDDs such as parallelize(). SparkSession is needed for SparkSQL and DataFrame operations.

Task 1: Creating the spark session and context

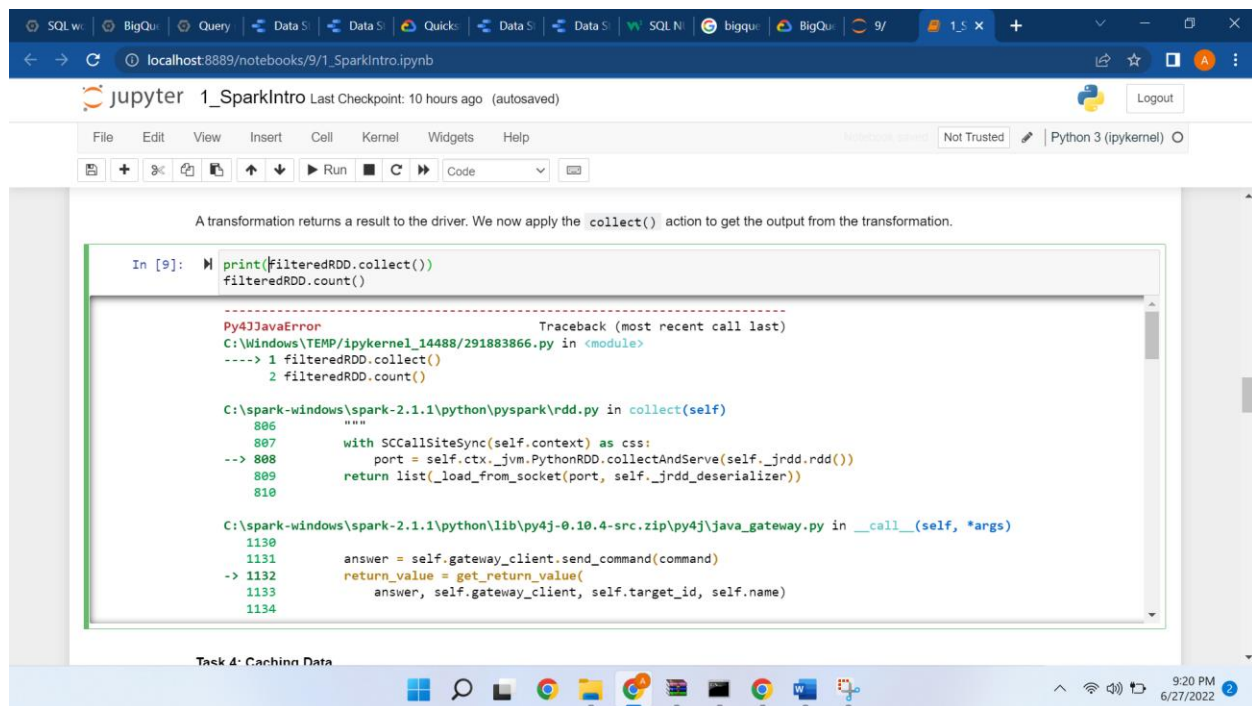
In [3]: # Creating a spark context class
sc = SparkContext()

# Creating a spark session
spark = SparkSession \
    .builder \
    .appName("Python Spark DataFrames basic example") \
    .config("spark.some.config.option", "some-value") \
    .getOrCreate()

Task 2: Initialize Spark session
To work with dataframes we just need to verify that the spark session instance has been created.

In [4]: spark
Out[4]: <pyspark.sql.session.SparkSession at 0x1b0d54c130>

Exercise 2: RDDs
```



The screenshot shows a Jupyter Notebook titled "1_SparkIntro" running on a local host. The notebook contains a code cell with the following Python code:

```
In [9]: print(filteredRDD.collect())
        filteredRDD.count()
```

The code cell has a status bar indicating it is "Not Trusted" and "Python 3 (ipykernel)". The output of the code cell is a traceback showing a `Py4JJavaError` occurring in the `collect()` method of the `filteredRDD` object. The traceback includes the following lines:

```
Py4JJavaError                                Traceback (most recent call last)
C:\Windows\TEMP\ipykernel_14488\291883866.py in <module>
----> 1 filteredRDD.collect()
      2 filteredRDD.count()

C:\spark-windows\spark-2.1.1\python\pyspark\rdd.py in collect(self)
   806     """
   807     with SCCallSiteSync(self.context) as css:
--> 808         port = self.ctx._jvm.PythonRDD.collectAndServe(self._jrdd.rdd())
   809         return list(_load_from_socket(port, self._jrdd_deserializer))
   810

C:\spark-windows\spark-2.1.1\python\lib\py4j-0.10.4-src.zip\py4j\java_gateway.py in __call__(self, *args)
  1130
  1131     answer = self.gateway_client.send_command(command)
--> 1132     return_value = get_return_value(
  1133         answer, self.gateway_client, self.target_id, self.name)
  1134
```

The task bar at the bottom of the screen shows the task "Task 4: Caching Data" and the system clock indicates 9:20 PM on 6/27/2022.