Project Statement for Milestone 1

Group 6

Group member: Jinyang Ruan, Rusu Wu, Brian Chan, Junqiao Mou, Yi Yao

Part 1: Project description (4%).

1. (1%) Problem statement: Answer the following questions:
a. Give a brief description of the candidate project and its tasks.

As a startup company, we need to develop a program to aid client analytics and visualize the massive scale of the datasets. The functions of the program include but are not limited to:

Airport and airline search:
- Find list of airports operating in the Country X
- Find the list of Airlines having X stops
- List of airlines operating with code share
- Find the list of active airlines in the United States - Airline aggregation:
- Which country (or) territory has the highest number of Airports.
- Top K cities with most incoming/outgoing airlines

Geospatial analytics: You must use Apache Sedona (https://sedona.apache.org/)
- Find the closest airport to a city X's geospatial coordinate
- Find the airport in each US state's geospatial boundary

b. Explain why do you want to choose this candidate project?

We chose the Airline Search Engine project by voting. 4 of us voted this topic because we think it is doable and relatively interesting.

c. Your group members and a description about the roles of each member (consider the four basic roles listed in Project Overview).

Jinyang Ruan: Infrastructure manager, visualization expert
Rusu Wu: Data analyst, ETL (Extract-Transform-Load) programmer
Brian Chan: Visualization expert, Data analyst
Junqiao Mou: ETL (Extract-Transform-Load) programmer, visualization expert
Yi Yao: Data analyst, Infrastructure manager

2. (1%) Datasets:
a. Give the link and description of the dataset.

Airports, Counties(2.66MB): http://openflights.org/data.html

Airport Codes(35MB): https://datahub.io/core/airport-codes

Airport ID, Location, Region(13.8MB): https://ourairports.com/data/

Routes(2.38MB): https://www.kaggle.com/open-flights/flight-route-database

Total: 53.84MB

b. Assume we use relational models, what are the data tables you plan to create based on the input datasets? At this stage, a rough estimation is fine.

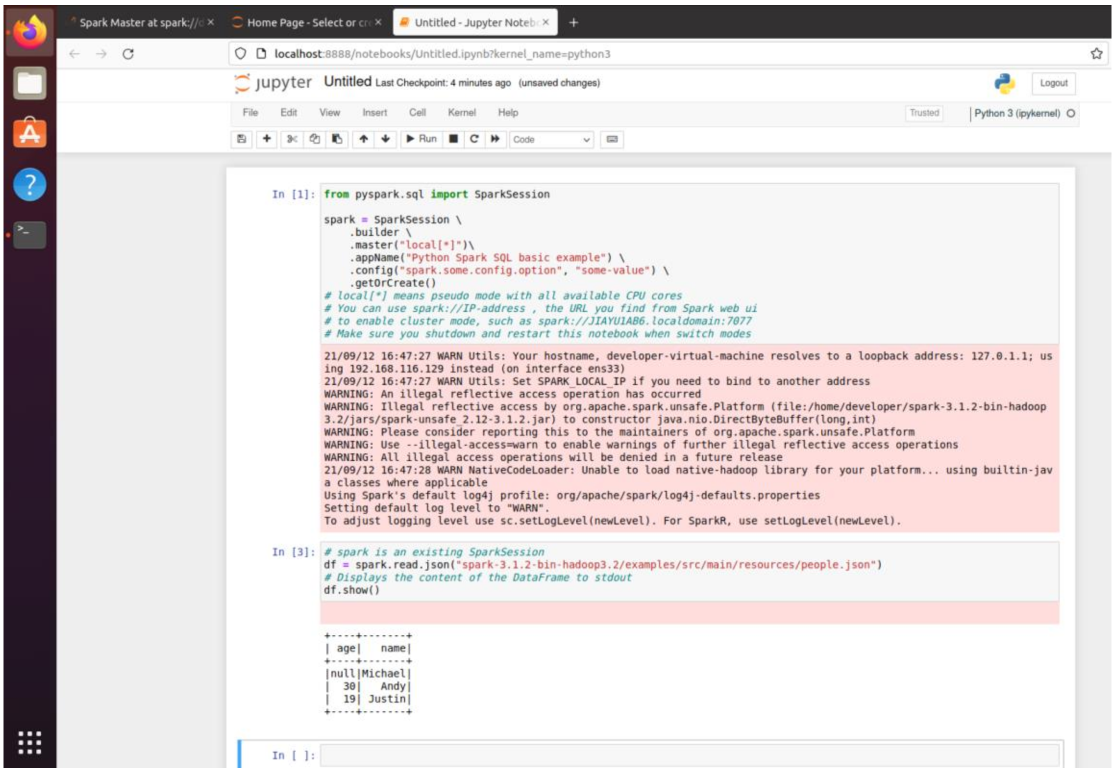
Airport ID, Name of the airport, Airline, Airline ID, Main city served by airport, Country or territory where airport is located, local code, continent, Code of Airport, Hours offset from UTC, Timezone, etc.

3. (1%) A timetable for your Milestones according to the roles of each group members. At this stage, a rough estimation is fine.

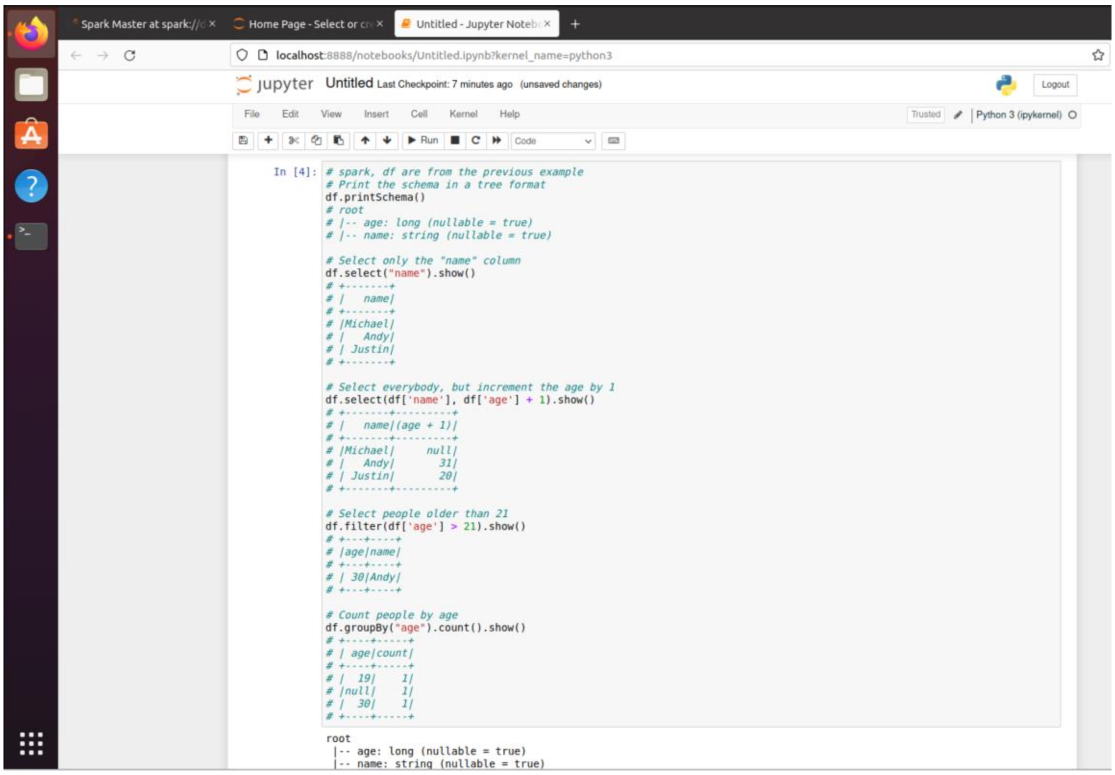
Phase	**Beginning Date**	**Deadline**
establish the project environment	20th September	26th September
prepare the large-scale datasets	20th September	10th October
data analyst	4th October	17th October
visualization expert	18th October	7th November
programming	18th October	7th November
debugging	8th November	21th November
finishing	22th November	28th November

Part 2: Set up PySpark environment (7%)

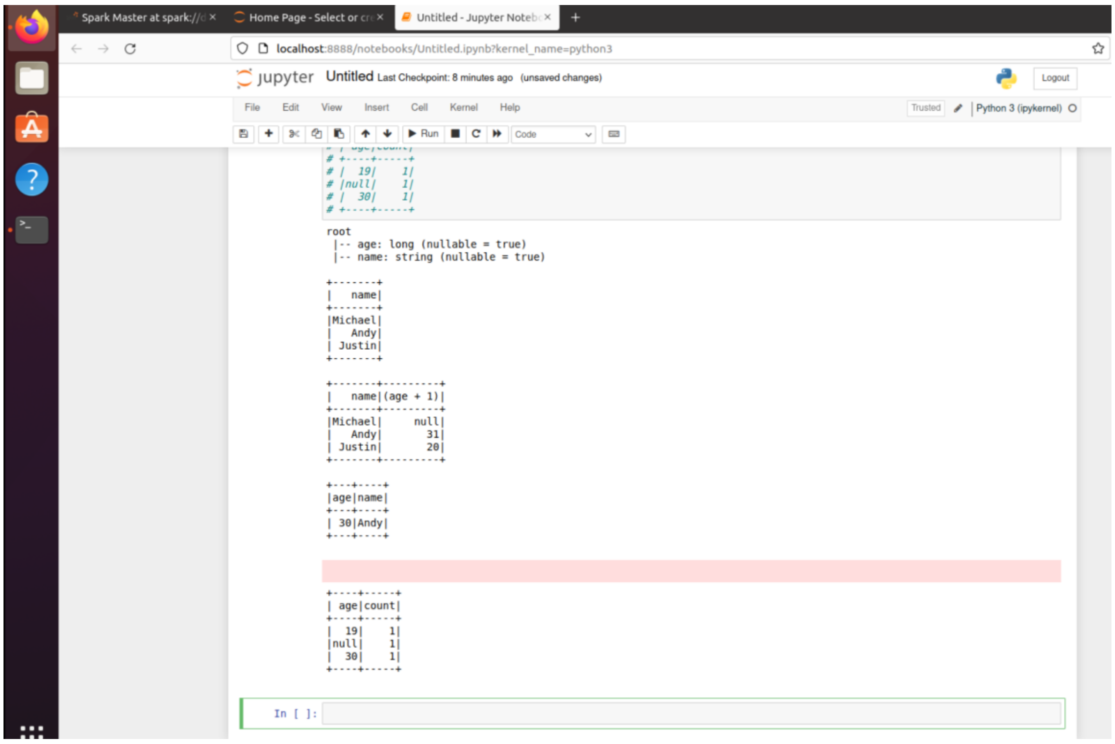
• Screenshot 1 (3%): A Jupyter notebook in a web browser that runs all commands (with outputs) listed in the given example notebook



```
In [1]: from pyspark.sql import SparkSession

        spark = SparkSession \
            .builder \
            .master("local[*]")\
            .appName("Python Spark SQL basic example") \
            .config("spark.some.config.option", "some-value") \
            .getOrCreate()
        # local[*] means pseudo mode with all available CPU cores
        # You can use spark://IP-address , the URL you find from Spark web ui
        # to enable cluster mode, such as spark://JIAYU1AB6.localdomain:7077
        # Make sure you shutdown and restart this notebook when switch modes

21/09/12 16:47:27 WARN Utils: Your hostname, developer-virtual-machine resolves to a loopback address: 127.0.1.1; us
ing 192.168.116.129 instead (on interface ens33)
21/09/12 16:47:27 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.apache.spark.unsafe.Platform (file:/home/developer/spark-3.1.2-bin-hadoop
3.2/jars/spark-unsafe_2.12-3.1.2.jar) to constructor java.nio.DirectByteBuffer(long,int)
WARNING: Please consider reporting this to the maintainers of org.apache.spark.unsafe.Platform
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
21/09/12 16:47:28 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-jav
a classes where applicable
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).

In [3]: # spark is an existing SparkSession
        df = spark.read.json("spark-3.1.2-bin-hadoop3.2/examples/src/main/resources/people.json")
        # Displays the content of the DataFrame to stdout
        df.show()

+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+

In [ ]:
```



```
In [4]: # spark, df are from the previous example
        # Print the schema in a tree format
        df.printSchema()
        # root
        # |-- age: long (nullable = true)
        # |-- name: string (nullable = true)

        # Select only the "name" column
        df.select("name").show()
        # +-------+
        # |   name|
        # +-------+
        # |Michael|
        # |   Andy|
        # | Justin|
        # +-------+

        # Select everybody, but increment the age by 1
        df.select(df['name'], df['age'] + 1).show()
        # +-------+---------+
        # |   name|(age + 1)|
        # +-------+---------+
        # |Michael|     null|
        # |   Andy|       31|
        # | Justin|       20|
        # +-------+---------+

        # Select people older than 21
        df.filter(df['age'] > 21).show()
        # +---+----+
        # |age|name|
        # +---+----+
        # | 30|Andy|
        # +---+----+

        # Count people by age
        df.groupBy("age").count().show()
        # +----+-----+
        # | age|count|
        # +----+-----+
        # |  19|    1|
        # |null|    1|
        # |  30|    1|
        # +----+-----+

root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)
```

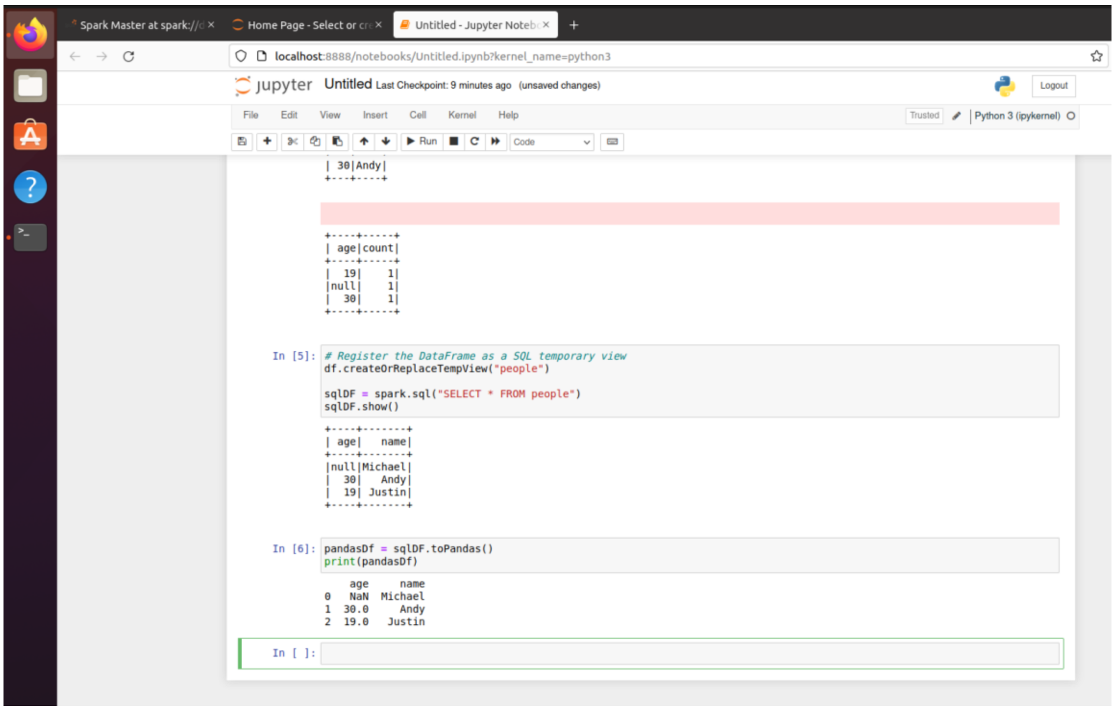


```
# +----+-----+
# |  19|    1|
# |null|    1|
# |  30|    1|
# +----+-----+
```

```
root
 |-- age: long (nullable = true)
 |-- name: string (nullable = true)

+-------+
|   name|
+-------+
|Michael|
|   Andy|
| Justin|
+-------+

+-------+---------+
|   name|(age + 1)|
+-------+---------+
|Michael|     null|
|   Andy|       31|
| Justin|       20|
+-------+---------+

+---+----+
|age|name|
+---+----+
| 30|Andy|
+---+----+
```

```
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  30|    1|
+----+-----+
```

```
In [ ]:
```



```
| 30|Andy|
+---+----+
```

```
+----+-----+
| age|count|
+----+-----+
|  19|    1|
|null|    1|
|  30|    1|
+----+-----+
```

```
In [5]: # Register the DataFrame as a SQL temporary view
        df.createOrReplaceTempView("people")

        sqlDF = spark.sql("SELECT * FROM people")
        sqlDF.show()
```

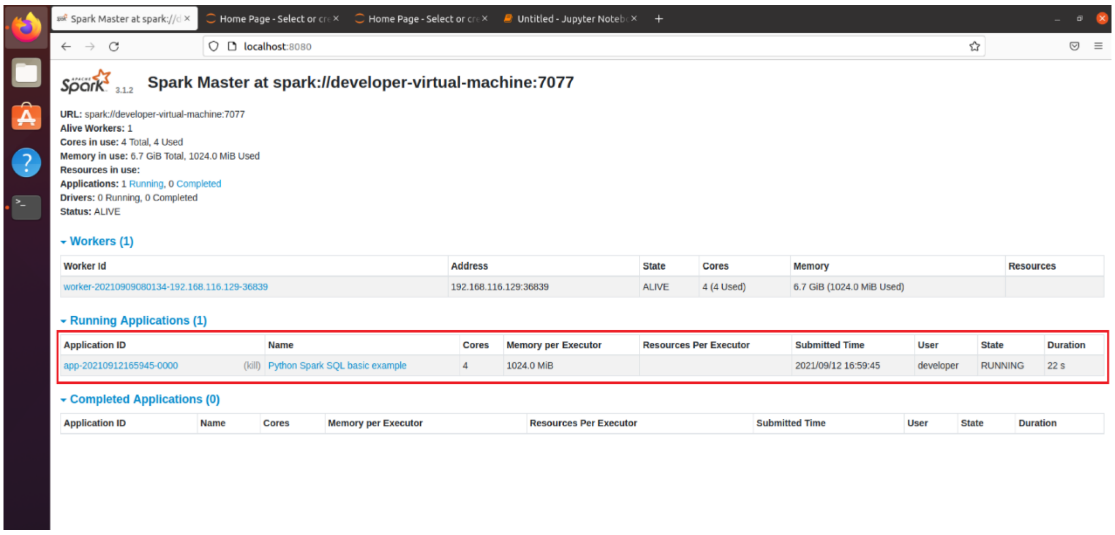
```
+----+-------+
| age|   name|
+----+-------+
|null|Michael|
|  30|   Andy|
|  19| Justin|
+----+-------+
```

```
In [6]: pandasDf = sqlDF.toPandas()
        print(pandasDf)
```

```
    age     name
0   NaN  Michael
1  30.0     Andy
2  19.0   Justin
```

```
In [ ]:
```

• Screenshot 2 (4%): Spark web UI in a web browser that shows your Jupyter notebook application connects to the cluster. The cluster only needs to have 1 worker which is the master machine itself.



Spark Master at spark://d ×   Home Page - Select or cre ×   Home Page - Select or cre ×   Untitled - Jupyter Notebo ×

localhost:8080

3.1.2 **Spark Master at spark://developer-virtual-machine:7077**

URL: spark://developer-virtual-machine:7077
Alive Workers: 1
Cores in use: 4 Total, 4 Used
Memory in use: 6.7 GiB Total, 1024.0 MiB Used
Resources in use:
Applications: 1 Running, 0 Completed
Drivers: 0 Running, 0 Completed
Status: ALIVE

Workers (1)

Worker Id	Address	State	Cores	Memory	Resources
worker-20210909080134-192.168.116.129-36839	192.168.116.129:36839	ALIVE	4 (4 Used)	6.7 GiB (1024.0 MiB Used)	

Running Applications (1)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration
app-20210912165945-0000 (kill)	Python Spark SQL basic example	4	1024.0 MiB		2021/09/12 16:59:45	developer	RUNNING	22 s

Completed Applications (0)

Application ID	Name	Cores	Memory per Executor	Resources Per Executor	Submitted Time	User	State	Duration