In this activity, you will practice loading data into the Spark Docker *container* you created in [Activity 19.1](https://classroom.emeritus.org/courses/10605/assignments/246292?module_item_id=1817240) and running SQL *queries* to verify the data.

In this activity, you will be using flight data to demonstrate the capabilities of Spark. You can access the data at the following link: [Departure Delays Data](https://classroom.emeritus.org/courses/10605/files/3007512/download).

Before beginning this activity, review the submission instructions below to ensure that you collect the required screenshots as you progress through the activity.

**To complete this activity, follow these steps:**

1. Under the Module19 folder, create a folder called data and download the [departuredelays.csv](https://classroom.emeritus.org/courses/10605/files/3007512/download) file. Provide a screenshot to show that you successfully created the data folder and downloaded the departuredelays.csv file.
2. Make sure that you have successfully completed Activity 19.1 and that the Spark Docker *containers* that you created are running. Provide a screenshot to show that the Spark Docker *containers* are running.
3. From the command shell, navigate to the data folder and run a docker cp command to copy the departuredelays.csv file to the bitnami\_spark\_1 *container’s* root folder. Provide a screenshot to show that you successfully ran the docker cp command to copy the departuredelays.csv file to the Spark Docker *container’s* root folder.
4. From the Docker Desktop, select the CLI for the bitnami\_spark\_1 *container* and list the contents of the root folder to verify that the departuredelays.csv file has been copied to the Spark Docker *container*. Provide a screenshot to show that the departuredelays.csv file has been copied to the root folder.
5. From the command line interface, enter pyspark and select the Return key. This will start up the PySpark interface. Provide a screenshot to show that you successfully started the PySpark interface.
6. From the PySpark interface, run the code below to create a Spark session:

| from pyspark.sql import SparkSession  #spark session spark = (SparkSession  .builder  .appName("DepartureDelays")  .getOrCreate()) |
| --- |

1. Provide a screenshot to show that you successfully ran the command to create a Spark session.
2. From the PySpark interface, run the code below to load the departuredelays.csv file into a Python *dataframe*:

| csv\_file = "/departuredelays.csv"  df = (spark.read.format("csv")  .option("interSchema", "true")  .option("header", "true")  .load(csv\_file)) df.createOrReplaceTempView("delays\_table") |
| --- |

1. Provide a screenshot to show that you successfully ran the command to load the departuredelays.csv file into a Python *dataframe*.
2. From the PySpark interface, run an SQL *query* to find rows where the distance is greater than 1,000. Order by the distance in descending order and only show the first 10 records. Select the distance, origin, and destination columns as follows: spark.sql("""SELECT distance, origin, destination FROM delays\_table WHERE distance > 1000 ORDER BY distance DESC""").show(10)  
   Provide a screenshot to show that you successfully ran the SQL *query* and selected the correct columns (distance, origin, and destination).
3. From the PySpark interface, run an SQL *query* to find rows where the delay is > 120 AND ORIGIN = ‘SFO’ and DESTINATION = ‘ORD’ ORDER BY delay DESC. Only show the first 10 records.  
   Provide a screenshot to show that you successfully ran the SQL *query* with the correct rows.
4. From the PySpark interface, run an SQL *query* that looks at each row and categorizes the data based on the flight delay:  
   WHEN delay is > 360 then ‘Very Long Delays’  
   WHEN delay is between 120 and 360 then ‘Long Delays’  
   WHEN delay is between 60 and 120 then ‘Short Delays’  
   WHEN delay is between 0 and 60 then ‘Tolerable Delays’  
   WHEN delay is = 0 then ‘No Delays’  
   ELSE it is ‘Early’. Label the column as delay\_length.  
   Also, select the delay, origin, destination, and delay\_length columns.   
   Show the first 10 rows of data.  
   Provide a screenshot to show that you successfully ran the SQL *query*, categorized the data correctly, selected the correct columns, and showed 10 rows of data.
5. For all of the flights in the dataset, what is the maximum delay in the dataset?  
     
   Provide a screenshot showing that you successfully ran the *query* to determine the correct maximum delay for all of the flights in the dataset.
6. For all of the flights in the dataset, what is the minimum delay in the dataset?  
     
   Provide a screenshot to show that you successfully ran the *query* to determine the correct minimum delay for all of the flights in the dataset.

Congratulations. You have successfully completed Activity 19.2. Now you know how to load data into PySpark and successfully run *queries* on the data using a combination of Python and SQL.

**Submission Instructions:**

Your submission for this activity should be a Word document that includes the following screenshots, each labeled for the step that the screenshot represents:

1. Provide a screenshot showing that you successfully created the data folder and downloaded the departuredelays.csv file.
2. Provide a screenshot showing that the Spark Docker *containers* that you created in Activity 19.1 are running.
3. Provide a screenshot showing that you successfully ran the docker cp command to copy the departuredelays.csv file to the Spark Docker *container’s* root folder.
4. Provide a screenshot showing that the departuredelays.csv file has been copied to the root folder.
5. Provide a screenshot showing that you successfully started the PySpark interface.
6. Provide a screenshot showing that you successfully ran the command to create a Spark session.
7. Provide a screenshot showing that you successfully ran the command to load the departuredelays.csv file into a Python *dataframe*.
8. Provide a screenshot showing that you successfully ran the SQL *query* and selected the correct columns (distance, origin, and destination).
9. Provide a screenshot showing that you successfully ran the SQL *query* with the correct rows.
10. Provide a screenshot showing that you successfully ran the SQL *query*, categorized the data correctly, selected the correct columns, and showed 10 rows of data.
11. Provide a screenshot showing that you successfully ran the *query* to determine the correct maximum delay for all of the flights in the dataset.
12. Provide a screenshot showing that you successfully ran the *query* to determine the correct minimum delay for all of the flights in the dataset.