DSC 530 Final Project summary

|  |  |
| --- | --- |
| Title | nyc taxi eda |
| Author | Rajdeep Biswas |
| Date | august 8th , 2020 |



##### Focus AREA

* Exploratory data analytics of the NYC Taxi & Limousine Commission - yellow taxi trip records. Time series, seasonality, and outlier analysis by using the Worldwide public holiday data and Worldwide hourly weather history data.
* Prediction of whether a taxi trip will result in a tip or not using logistic regression.

##### guiding principles

The work that will be subsequently done as part of this paper will have at the very least embody the following principles (ai/responsible-ai, n.d.):

* Fair - AI must maximize efficiencies without destroying dignity and guard against bias.
* Accountable - AI must have algorithmic accountability.
* Transparent - AI systems must be transparent and understandable.
* Ethical - AI must assist humanity and be designed for intelligent privacy.

##### source dataset

For this paper the source are the 3 datasets as mentioned blow from Azure Open Datasets (docs.microsoft.com, n.d.) which are hosted in the Azure Blob storage (storage/blobs/, n.d.) which provides a scalable, cost-efficient object storage in the cloud.

1. NYC Taxi & Limousine Commission - yellow taxi trip records. This dataset is stored in Parquet format. There are about 1.5B rows (50GB) in total as of 2018.This dataset contains historical records accumulated from 2009 to 2018. View data example [here](https://github.com/RajdeepBiswas/NYC_Taxi/blob/master/Dataset/NYC_Taxi_Data.csv)

* Blob location\*: [https://azureopendatastorage.blob.core.windows.net/nyctlc/yellow/puYear=\*/puMonth=\*/\*.parquet](https://azureopendatastorage.blob.core.windows.net/nyctlc/yellow/puYear=*/puMonth=*/*.parquet)

1. Worldwide public holiday data sourced from PyPI holidays package and Wikipedia, covering 38 countries or regions from 1970 to 2099. This dataset is stored in Parquet format. It is a snapshot with holiday information from 1970-01-01 to 2099-01-01. The data size is about 500KB. View data example [here](https://github.com/RajdeepBiswas/NYC_Taxi/blob/master/Dataset/Public_Holidays_Data.csv)

* Blob location\*: [https://azureopendatastorage.blob.core.windows.net/holidaydatacontainer/Processed/\*.parquet](https://azureopendatastorage.blob.core.windows.net/holidaydatacontainer/Processed/*.parquet)

1. Worldwide hourly weather history data (example: temperature, precipitation, wind) sourced from the National Oceanic and Atmospheric Administration (NOAA).This dataset is stored in Parquet format. It is updated daily and contains about 400M rows (20GB) in total as of 2019.This dataset contains historical records accumulated from 2008 to the present. View data example [here](https://github.com/RajdeepBiswas/NYC_Taxi/blob/master/Dataset/Hourly_Weather_Data.csv)

* Blob location\*: [https://azureopendatastorage.blob.core.windows.net/isdweatherdatacontainer/ISDWeather/year=\*/month=\*/\*.parquet](https://azureopendatastorage.blob.core.windows.net/isdweatherdatacontainer/ISDWeather/year=*/month=*/*.parquet)

\*Note: the blob location is not accessible from browser.

##### Technology and platform

##### Synapse analytics

Azure Synapse (synapse-analytics) is Azure SQL Data Warehouse evolved—blending big data, data warehousing, and data integration into a single service for end-to-end analytics at cloud scale. Synapse has SQL and Spark built in along with pipelines and a studio GUI experience.

##### PYSpark

PySpark is the Python API written in python to support Apache Spark . Apache Spark is a distributed framework that can handle Big Data analysis. Apache Spark is written in Scala and can be integrated with Python, Scala, Java, R, SQL languages. Spark is basically a computational engine, which (python) works with huge sets of data by processing them in parallel and batch systems.

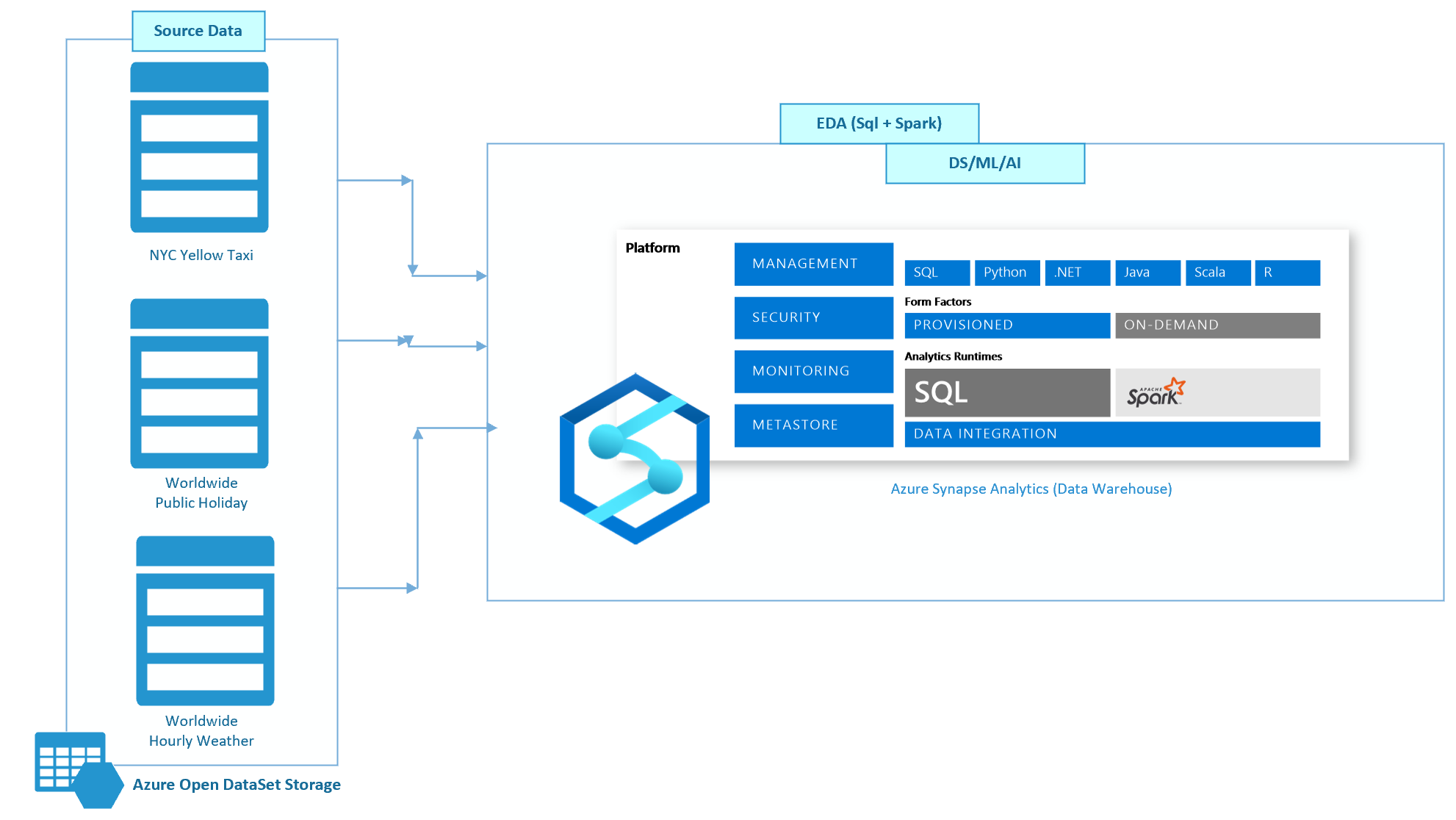
##### SparkSQL

Spark SQL (spark.apache.org, n.d.) is a Spark module for structured data processing. Unlike the basic Spark RDD API, the interfaces provided by Spark SQL provide Spark with more information about the structure of both the data and the computation being performed. Internally, Spark SQL uses this extra information to perform extra optimizations. There are several ways to interact with Spark SQL including SQL and the Dataset API. When computing a result, the same execution engine is used, independent of which API/language you are using to express the computation.

##### Azure blob storage

Azure Blob Storage helps you create data lakes for your analytics needs and provides storage to build powerful cloud-native and mobile apps. Optimize costs with tiered storage for your long-term data, and flexibly scale up for high-performance computing and machine learning workloads (docs.microsoft.com, n.d.).

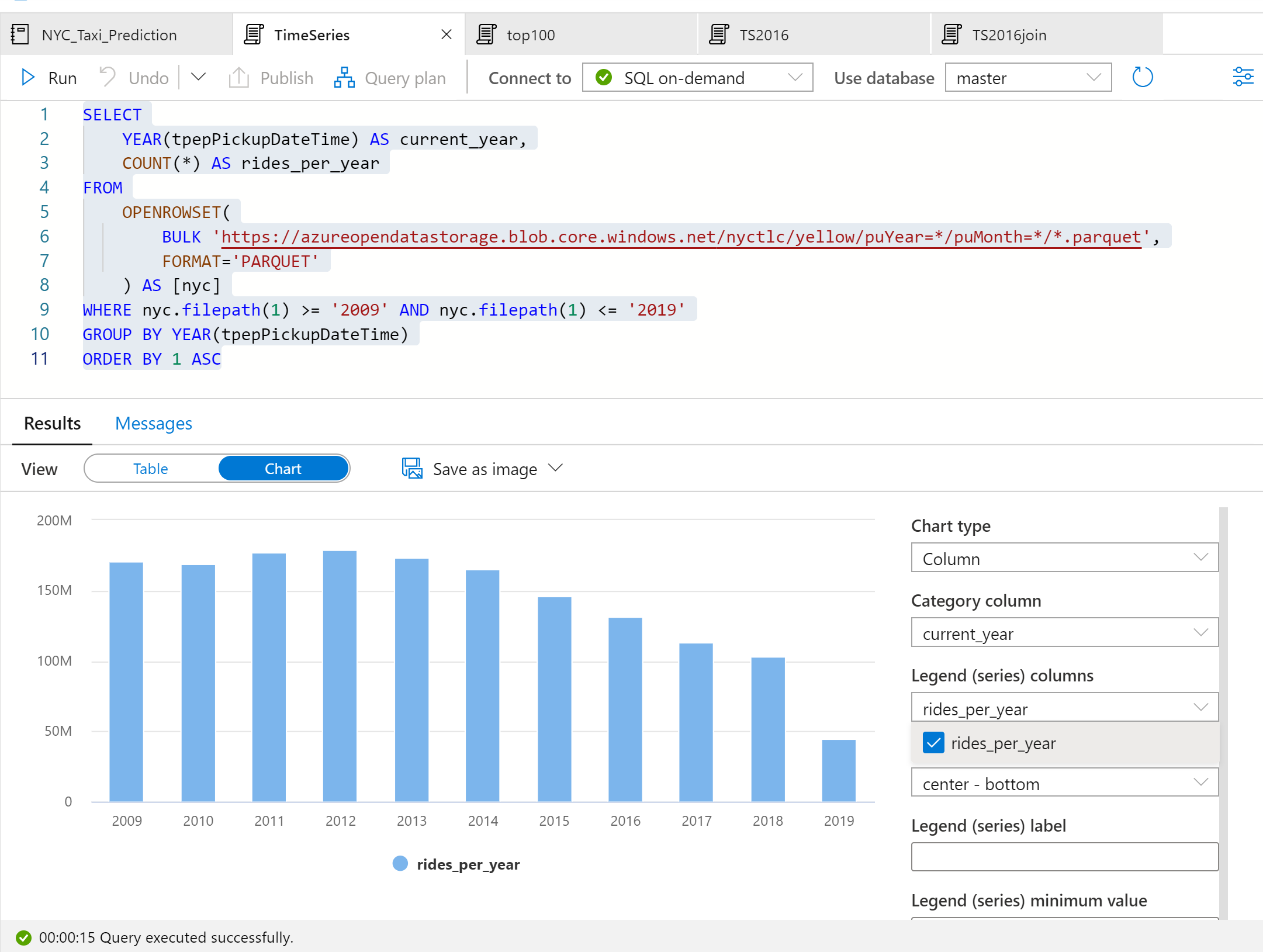
##### architecture



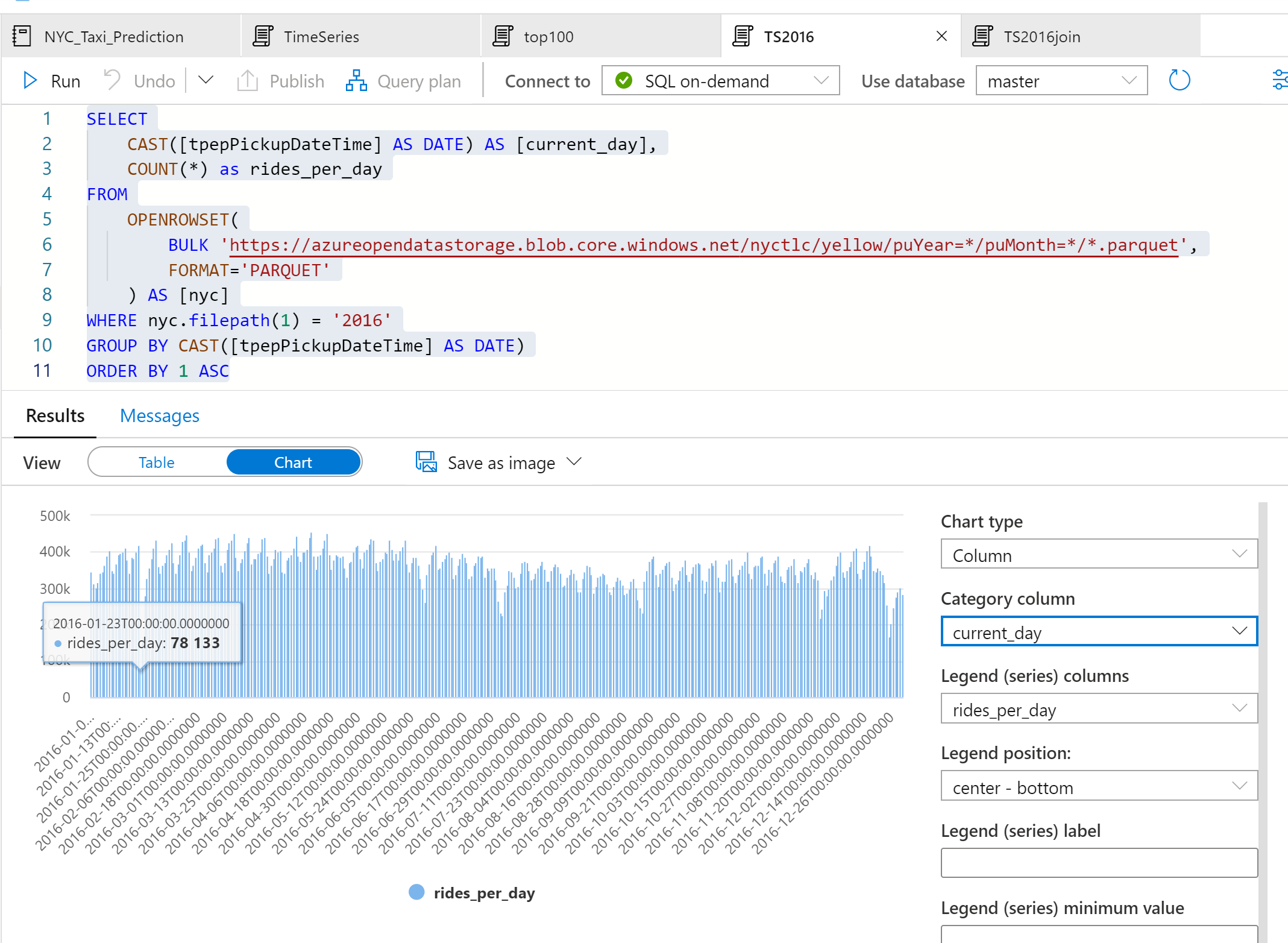
##### EDA highlights

##### Time series, seasonality, and outlier analysis

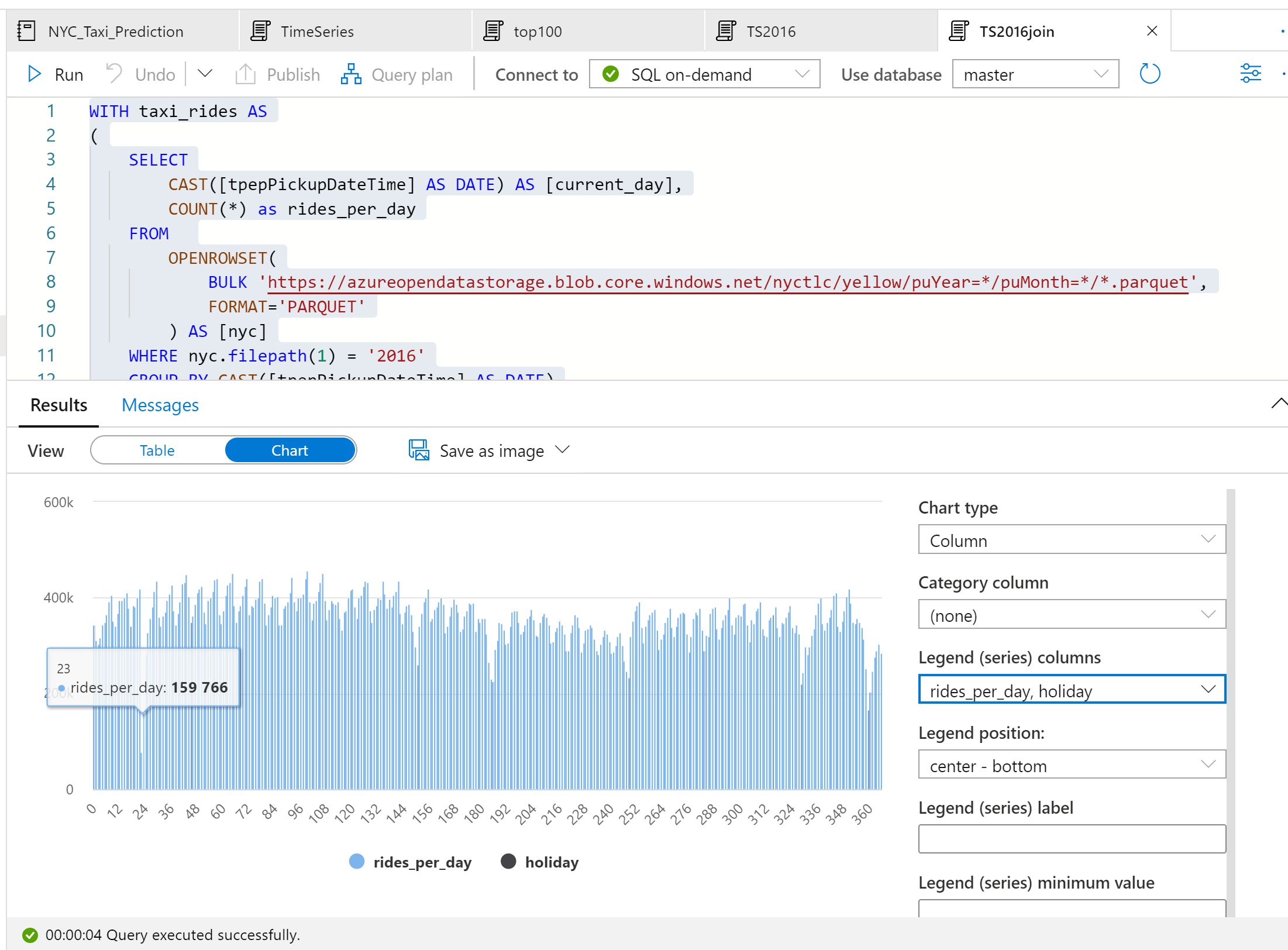
### Summarize the yearly number of taxi rides by using the following query:



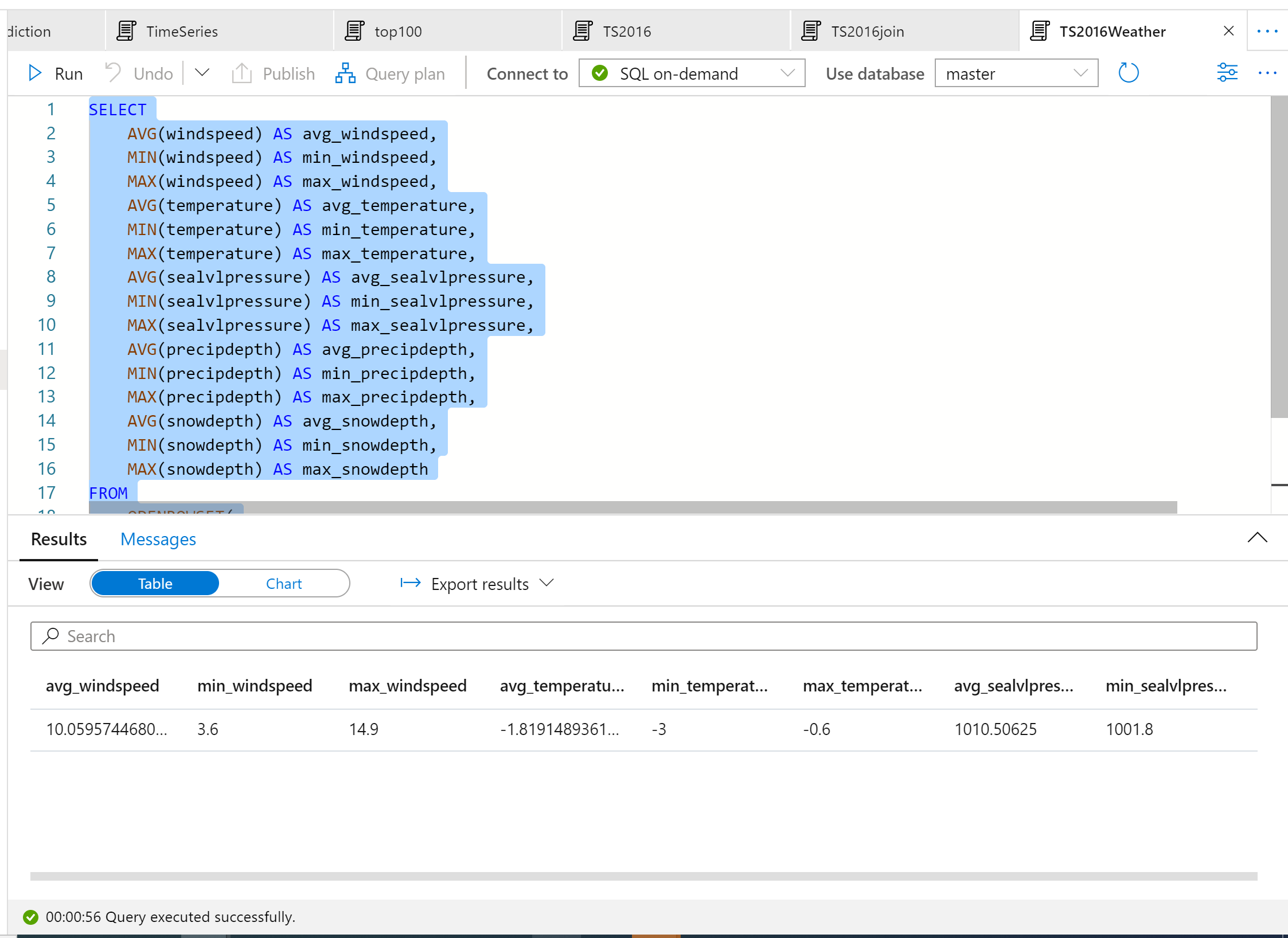
Note:  data for 2019 is incomplete. As a result, there is a huge drop in the number of rides for that year. For 2016, the following query returns the daily number of rides during that year. Visualize data by plotting the Column chart with the Category column set to current\_day and the Legend (series) column set to rides\_per\_day.



### From the plot chart, we can see that there's a weekly pattern, with Saturdays as the peak day. During summer months, there are fewer taxi rides because of vacations. There are also some significant drops in the number of taxi rides without a clear pattern of when and why they occur. Next, let us see if the drops correlate with public holidays by joining the NYC Taxi rides dataset with the Public Holidays dataset. This time, we want to highlight the number of taxi rides during public holidays. For that purpose, we choose none for the Category column and rides\_per\_day and holiday as the Legend (series) columns.



From the plot chart, we can see that during public holidays the number of taxi rides is lower. There is still one unexplained large drop on January 23. Let us check the weather in NYC on that day by querying the Weather Data dataset.

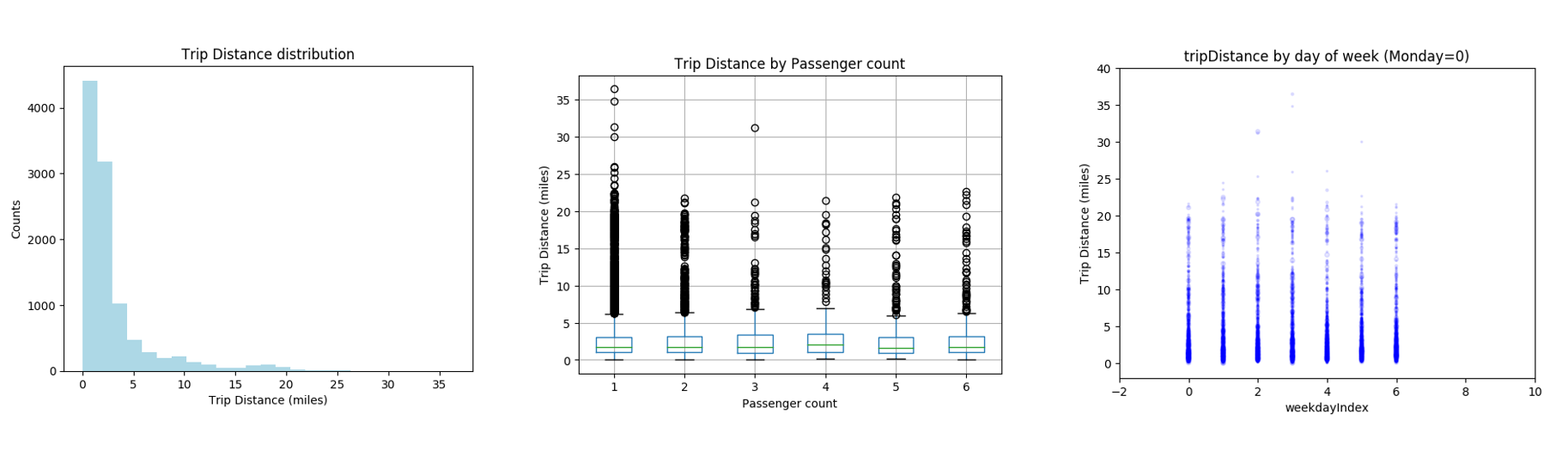


The results of the query indicate that the drop in the number of taxi rides occurred because:

* There was a blizzard on that day in NYC with heavy snow (~30 cm).
* It was cold (temperature was below zero degrees Celsius).
* It was windy (~10 m/s).

##### EDA & Predictive analysis on NYC Yellow Taxi data using PySpark and SparkSQL

The code shows three different visualizations of the data related to tips that lead to conclusions about the state and quality of the data.

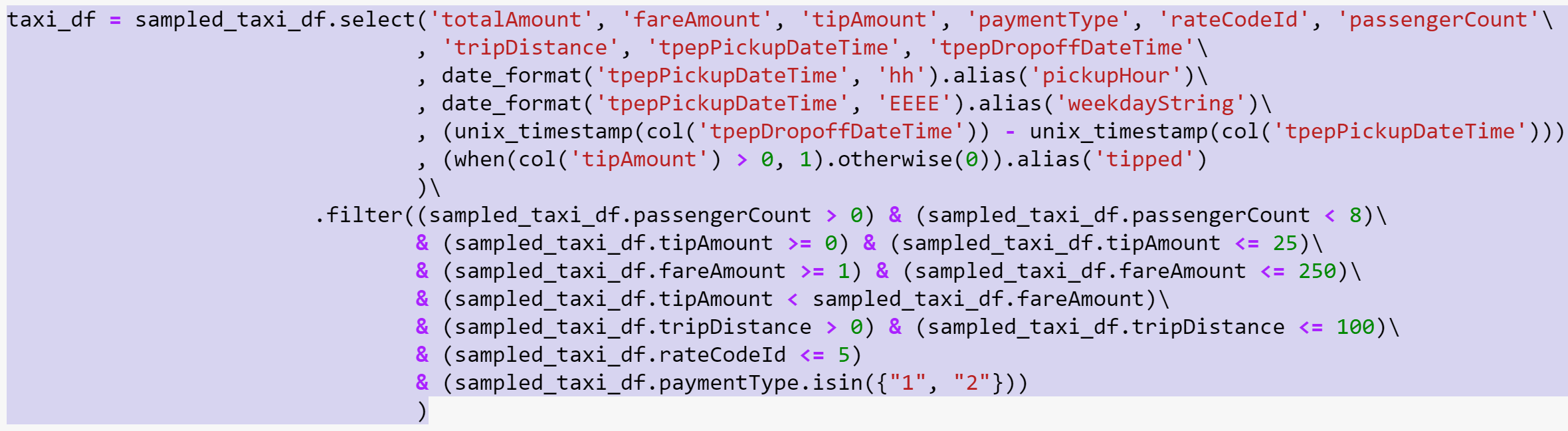


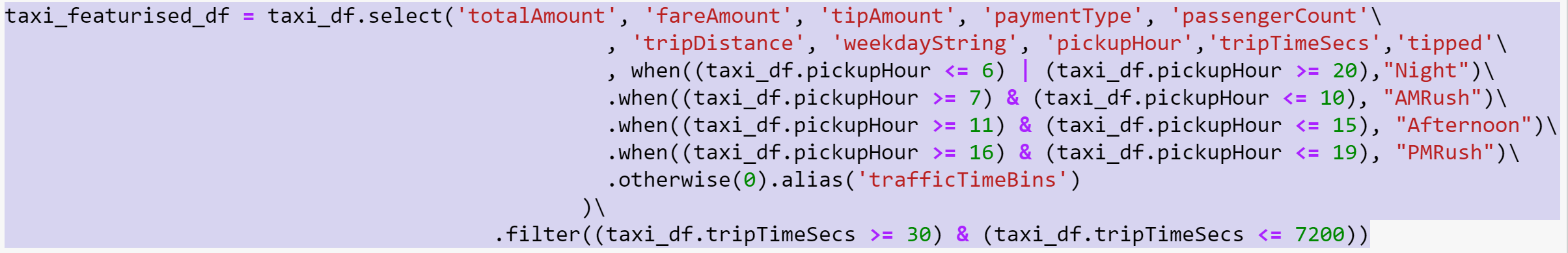
##### Data Preparation

The data in its raw form is frequently not suitable for passing directly to a model. A series of actions must be performed on the data to get it into a state where the model can consume it.

In the code below four classes of operations are performed:

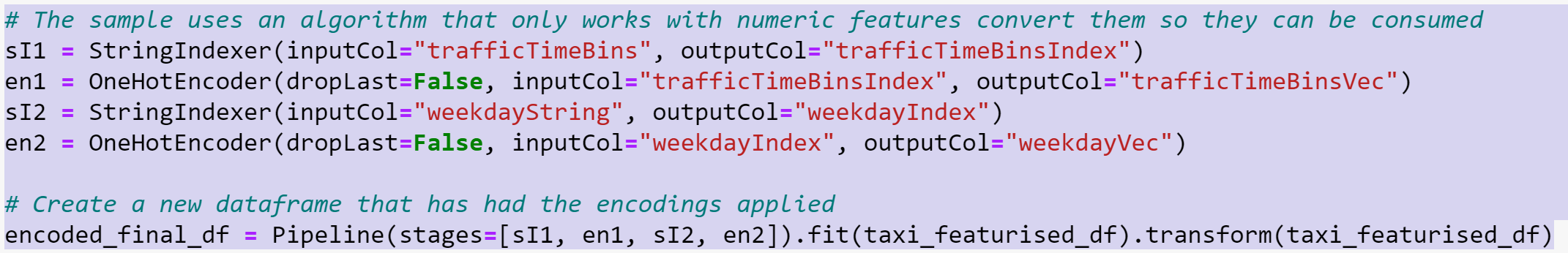
* The removal of outliers/incorrect values through filtering.
* The removal of columns, which are not needed.
* The creation of new columns derived from the raw data to make the model work more effectively, sometimes called featurization.
* Labeling, as we are undertaking binary classification (will there be a tip or not on a given trip) there is a need to convert the tip amount into a 0 or 1 value.



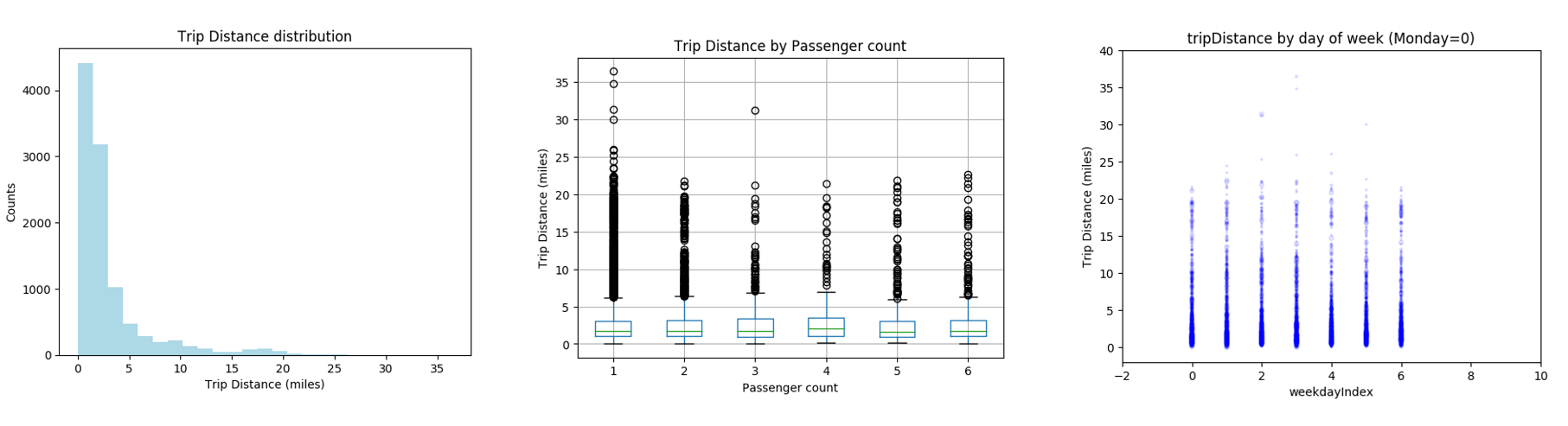


##### OneHotEncoding

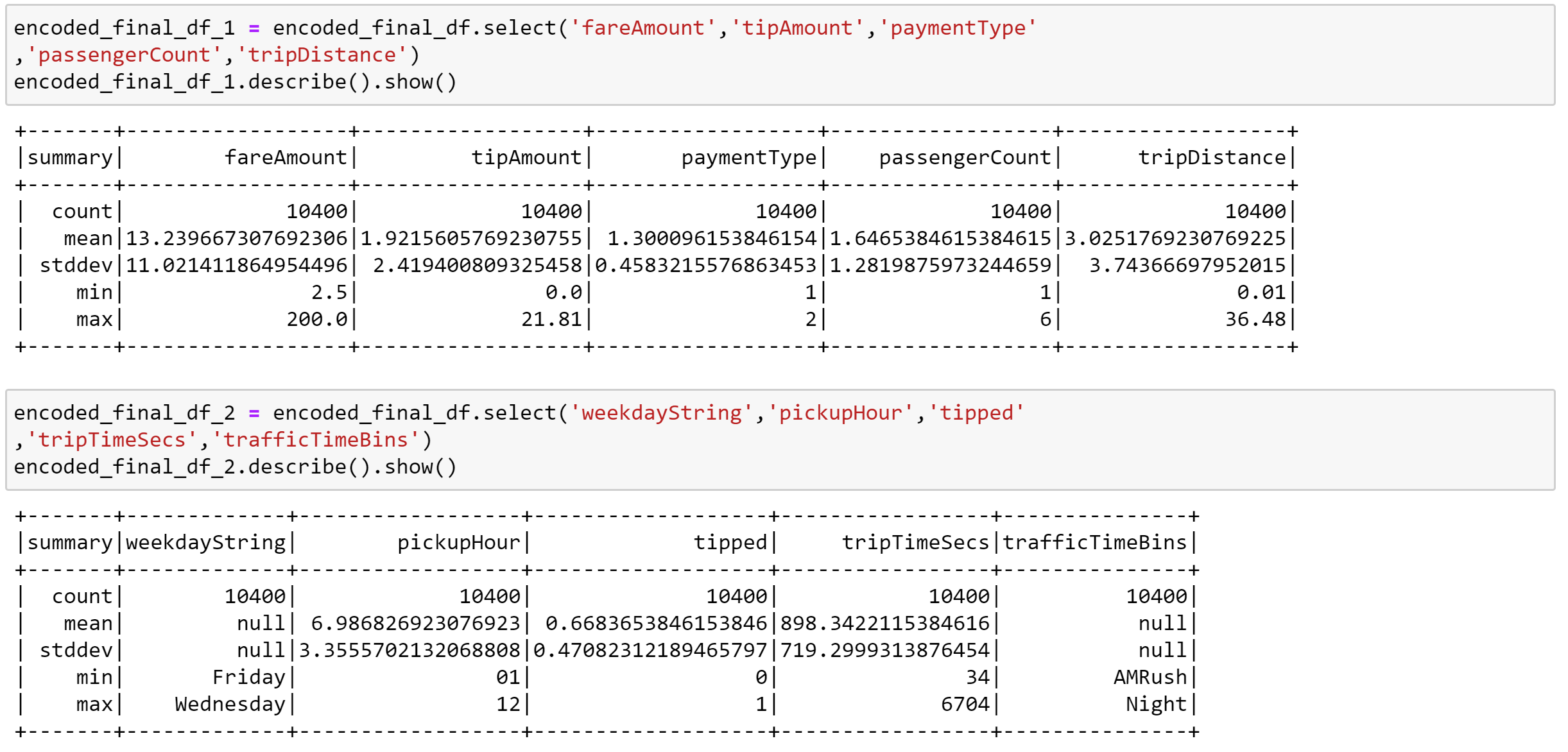
The final task is to convert the labeled data into a format that can be analyzed by logistic regression. The input to a logistic regression algorithm needs to be a set of label-feature vector pairs, where the feature vector is a vector of numbers representing the input point. So, we need to convert the categorical columns into numbers. The trafficTimeBins and weekdayString columns need to be converted into integer representations. There are multiple approaches to performing the conversion, however the approach taken in this example is OneHotEncoding, a common approach.



##### Visualization with trip distance



##### Summary Statitics



##### Build a logistic regression model

##### Step1: Create Train Test Data set

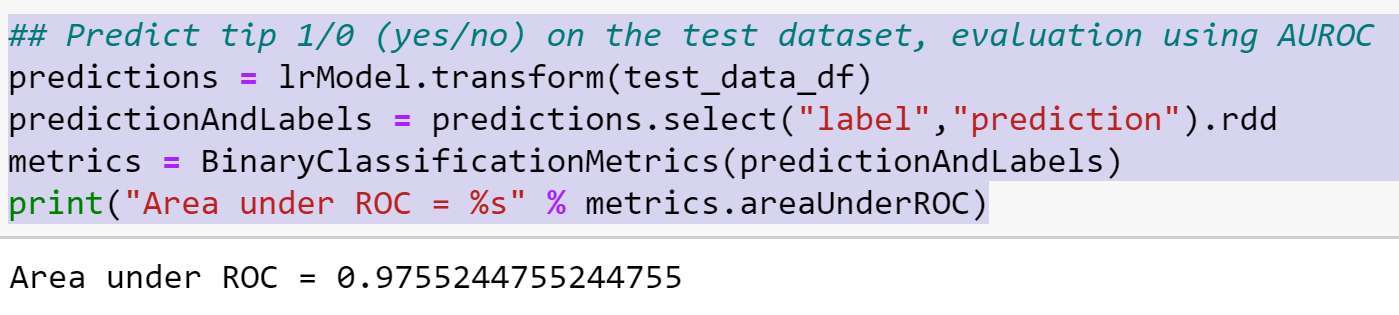
The first task was to split the dataset into a training set and a testing or validation set. I took a 70:30 split for training to testing.

##### Step2: Create Model

The second task was to select a formula and create the model. Logistic regression is an algorithm that I used for classification. Spark's logistic regression API is useful for binary classification or classifying input data into one of two groups. The process of logistic regression produces a logistic function that can be used to predict the probability that an input vector belongs in one group or the other.

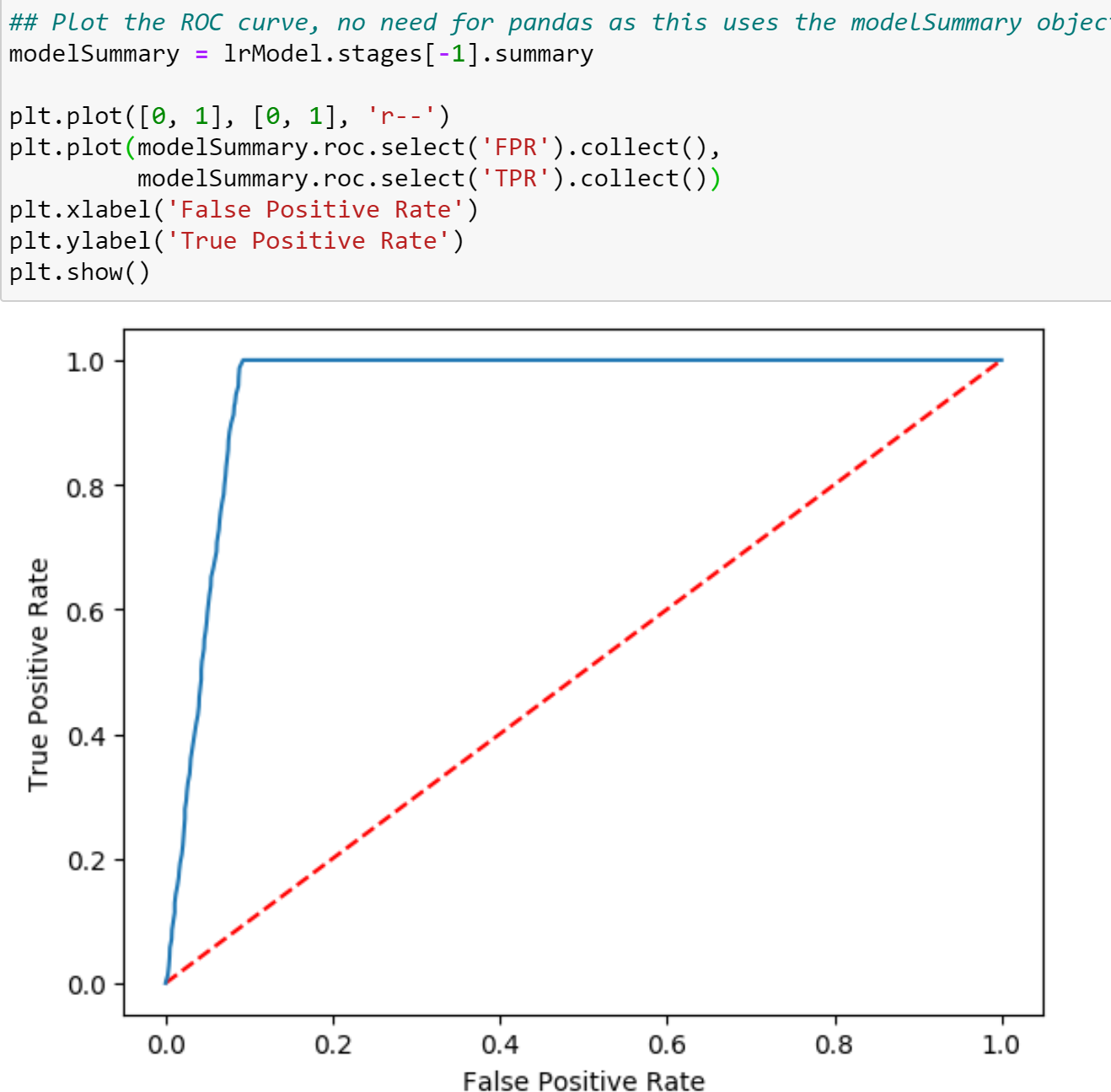
##### Step3: Test Model

The third task is to test the model. AUROC  (Area Under the Receiver Operating Characteristics) can be computed using the trapezoidal rule. In general, an AUC of 0.5 suggests no discrimination (i.e., ability to diagnose patients with and without the disease or condition based on the test), 0.7 to 0.8 is considered acceptable, 0.8 to 0.9 is considered excellent, and more than 0.9 is considered outstanding.



##### Step4: Visualize the prediction

When we need to check or visualize the performance of classification problem, we use AUC (Area Under the Curve) ROC (Receiver Operating Characteristics) curve. It is one of the most important evaluation metrics for checking any classification model’s performance. It is also written as AUROC (Area Under the Receiver Operating Characteristics)



##### References

*ai/responsible-ai*. (n.d.). Retrieved from microsoft.com: https://www.microsoft.com/en-us/ai/responsible-ai

*docs.microsoft.com*. (n.d.). Retrieved from https://docs.microsoft.com/en-us/azure/open-datasets/overview-what-are-open-datasets

*python.* (n.d.). Retrieved from spark.apache.org: https://spark.apache.org/docs/latest/api/python/index.html

*spark.apache.org*. (n.d.). Retrieved from https://spark.apache.org/docs/latest/sparkr.html

*spark.apache.org*. (n.d.). Retrieved from https://spark.apache.org/docs/latest/sql-programming-guide.html

*storage/blobs/*. (n.d.). Retrieved from microsoft.com: https://azure.microsoft.com/en-us/services/storage/blobs/

*synapse-analytics.* (n.d.). Retrieved from azure.microsoft.com: https://azure.microsoft.com/en-us/services/synapse-analytics/