# **Connecting to Blb SA from Azure Databricks cluster for mounting the directory:**

**Step1:** Create a Blob Storage Account & container inside the storage account.

**Step2:** Create an Azure Databricks>>Launch the workspace>>and create a cluster (by clicking on compute left side).

**Step3:** Get the Storage account; Access keys; & container name inside the storage account as shown below as an example

**Storage Account Name:** 1961mysa

**Access Keys:** I/4UBW2dm+Cl1XX2i2N9Y5LA3d1VCQB6WbX64p+fRpXxQPcfDG/DLKbcwAgPbigoE0cufB+4TIH6+ASt9xzTnA==

**Container name:** mycon

**Step4:** Now from the above link copy the entire code (To set up the file access, you need to do this:) and paste it in Pyspark notebook cell and make the changes accordingly as per the SA, Container name & Access keys.

**Step5:** click on extreme right top nob in the cell body and run the python code inside the cell by clicking on **Run cell** and we see output as below.

Mounting: /mnt/mycon

=> Directory /mnt/mycon already mounted

# **Reading .csv file from Blob SA with Windows Azure Storage Blob Service(WASBS) METHOD from Azure Databricks cluster:**

**Step1:** Create a Blob Storage Account & container inside the storage account and upload the .csv file inside the SA container.



**Step2:** Create an Azure Databricks>>Launch the workspace>>and create a cluster (by clicking on compute left side).

**Step3:** Create a notebook and paste the code (Get the code from above link and also as shown below) inside the cell and run the code.

**Step4:** Then will see the output as shown below

A screenshot of a computer

Description automatically generated

**Note:** These commands are absolute case sensitive, so while typing these commands in cluster notebook we have to give **100% attentions with upper case and lower case.**

**Step5:** click on + as shown below in image and here we can do the data visualization and multiple types of charts(Line chart, Bar chart, Area chart, Pie chart, Scatter chart, bubble chart…etc. etc.)and can also apply various filters on it.

A screenshot of a computer

Description automatically generated

**Step6:** create a new cell and type the below commands which gives different results

**df.printSchema()>>**this command shows the schema of csv file

**df.describe().show()>>**this command shows the aggregates values of the file records(like count, mean, min, max, stddev…etc)

**df.head(5)>>**this command shows only top 5(whatever the No we pass here that many records will be displayed)records.

**Step7:** create a new cell and type the below commands which helps us to create the temporary view and to convert or replace the code from Python to SQL

df.createOrReplaceTempView("xrate")>>hold the result in temp view and to convert from Python to SQL

df = spark.sql("select \* from xrate")>>Typing SQL query with spark.sql method

display(df)>>method to display the output.

**Step8:** Create a new cell and paste the below command to get the output displayed as Group by year and order by year Desc from xrate(temp view)

df = spark.sql("SELECT YEAR(Date) as year, COUNT(Date) as count, MEAN(Rate) as mean From xrate GROUP BY YEAR(Date) ORDER BY year DESC")>>command to get the data from xrate

display(df)**>>**command to display the output. And the result is shown below

A screenshot of a computer

Description automatically generated

**Step9:**

Create a new cell and if we want to write the SQL query directly then first select SQL on to right inside the cell and then directly, we can write the SQL Queries as shown below.

%sql

SELECT YEAR(Date) as year, COUNT(Date) as count, MEAN(Rate) as mean From xrate GROUP BY YEAR(Date) ORDER BY year DESC

A screenshot of a computer

Description automatically generated

Here on top if we see **%sql>>this is called as Magic command**, when we select SQL on top right then we can see this magic command will automatically be printed in our cell body as shown above.

# Pyspark:

When we are integrating with Python library and with this spark we can able to call it in our coding. Hence wit this we can Pyspark.

Here in the above methods, groupBy, agg, sort..etc.. these are all methods and we are applying these methods on top of dataframe(df).

# Importing Apache Spark libraries & writing the code in Databricks cluster in %Scala:

Create a new cell in same cluster Notebook and paste the below code

|  |
| --- |
| %scala |
|  |  |
|  | import org.apache.spark.sql.functions.\_ |
|  | var df = spark.table("xrate") |
|  | // or |
|  | // df = spark.sql("select \* from xrate") |
|  | var Row(minValue, maxValue) = df.select(min("Rate"), max("Rate")).head |
|  |  |
|  | println(s"Min: ${minValue}, Max: ${maxValue}") |

Here in above code, we are importing Apache Spark function to get the Min & Max values for Rate column

Hence, like this we can write the code in Azure Data bricks cluster notebooks on either Python, Sql or Scala by mentioning the magic command in cell body.