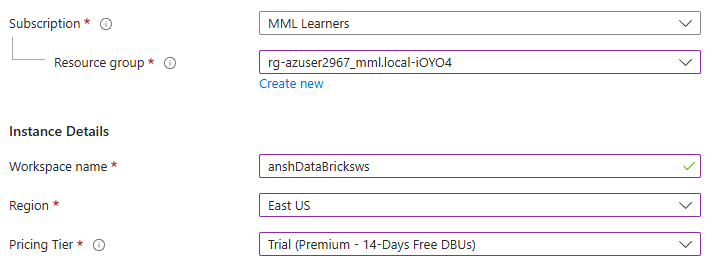
**Ansh Ranjan  
Azure Databricks – All Exercises**

**Exercise 1 – Settings up DataBricks and Spark Basics**

**TASK 1: Create a new Azure DataBricks workspace**

1. Go to Azure Portal > Azure DataBricks > Create > Enter details > Review and Create  
   

**TASK 2: Launch a spark cluster and explore databricks interface**

1. Launch your Databricks workspace > Computer side tab > Create Compute  
   A screenshot of a computer

   AI-generated content may be incorrect.
2. Now click on New > Notebook and you will have your notebook ready in your workspace  
   A screenshot of a computer

   AI-generated content may be incorrect.

The **Azure Databricks workspace** provides a unified environment for data engineering, data science, and machine learning. The main parts of the interface include:

**1. Workspace**

* Organize notebooks, libraries, and workflows.
* Create folders and share them with users or groups.

**2. Notebooks**

* Interactive notebooks supporting **Python**, **SQL**, **Scala**, and **R**.
* Run code in cells and visualize data easily.

**3. Clusters**

* Spin up Spark clusters for running jobs or interactive analysis.
* Choose autoscaling and runtime version (with Delta, ML, or GPU support).

**4. Jobs**

* Schedule notebooks or workflows.
* Automate ETL, ML training, or batch jobs.

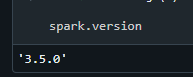
**5. Data**

* Browse databases, tables, and files.
* Supports Unity Catalogue (if enabled) for secure, centralized governance.

**6. Repos (Git Integration)**

* Connect to GitHub or Azure DevOps to version-control notebooks and code.

**TASK 3: Run Basic Spark commands**

1. Running spark.version command in first cell  
    A screen shot of a computer

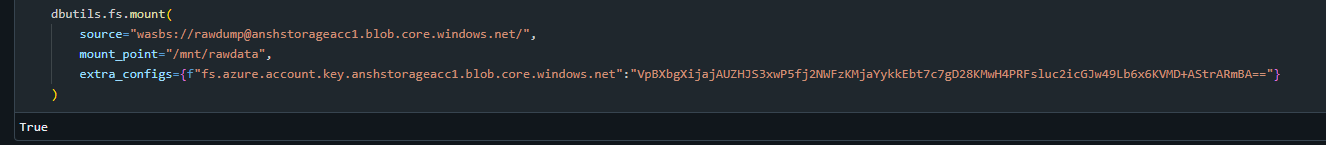
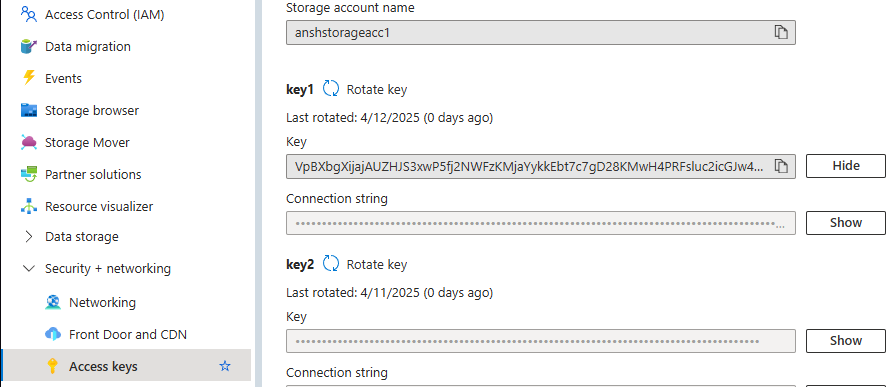
   AI-generated content may be incorrect.
2. Now that we know our databricks workspace is ready we can start performing ETL tasks.

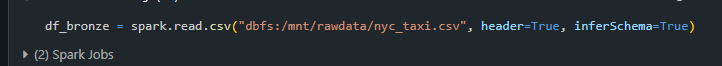
**Exercise 2 – Data Ingestion**

**TASK 1: Load a dataset into your Databricks using Spark**

1. Our data resides in a ‘rawdump’ named container in our storage account.   
   A screenshot of a computer

   AI-generated content may be incorrect.
2. Creating a spark session  
   A screen shot of a computer

   AI-generated content may be incorrect.
3. Now we have to mount our data point to our workspace.  
     
   You can get your access key by going to storage account > Security + networking > Access Keys  
   
4. Reading data into a dataframe  
   A screenshot of a computer

   AI-generated content may be incorrect.  
   

**TASK 2: Inspect the data**

1. Displaying the data  
   A screenshot of a computer

   AI-generated content may be incorrect.
2. Dataframe schema  
   A screen shot of a computer program

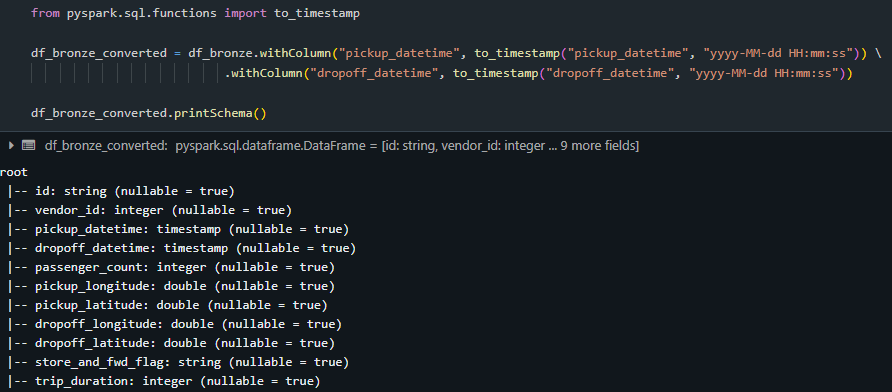
   AI-generated content may be incorrect.
3. Number of rows  
   A screenshot of a computer

   AI-generated content may be incorrect.

**TASK 3: Perform data cleaning**

1. Dropping null values from the df  
   A screen shot of a computer screen

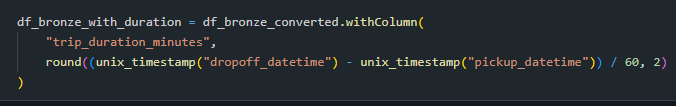
   AI-generated content may be incorrect.  
   Now there should not be any null values  
   A computer screen shot of a computer code

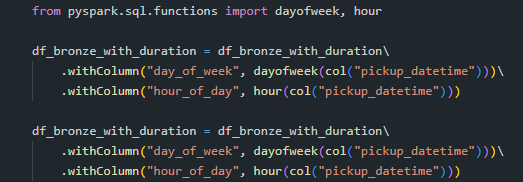
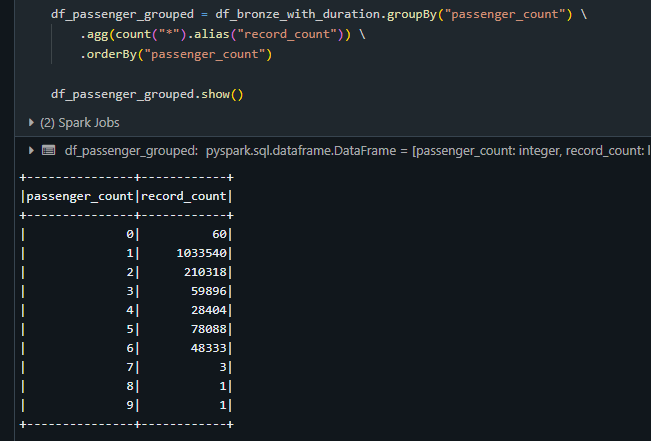
   AI-generated content may be incorrect.
2. As we saw earlier, our pickup and dropoff columns are of double data type. We need to convert them into standard datetime format  
   
3. Checking for any rows where dropoff time might be before pickup time for data quality  
   A screen shot of a computer code

   AI-generated content may be incorrect.

**EXERCISE 3: Data Transformation**

**TASK 1 and 2: Apply transformation to data (filtering, grouping, etc)**

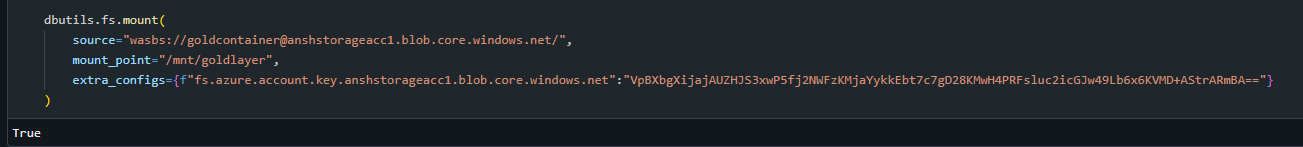
1. Creating a new column trip\_duration as difference of pickup and dropoff time in minutes  
     
   A screen shot of a computer

   AI-generated content may be incorrect.
2. Creating new columns day\_od\_week and Hour\_of\_day  
   
3. Finding number of trips per number of passengers  
   
4. Grouping data by day of week to get number of trips made for each day of the week  
   A screen shot of a computer program

   AI-generated content may be incorrect.

**EXERCISE 4 – Data Storage and Retrieval**

**TASK 1: save the transformed data into Azure blob**

1. Mounting gold layer storage container  
   
2. Writing the dataframe in form of Delta Tables in Blob container  
   
3. Checking data in gold container  
   A screenshot of a computer

   AI-generated content may be incorrect.

**TASK 2: Read the saved data back into df**

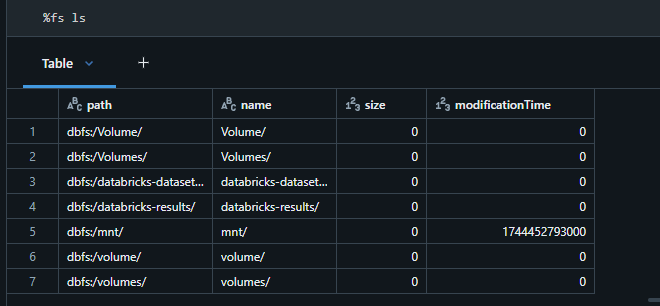
1. Reading the delta table back as a new df  
   A screenshot of a computer program

   AI-generated content may be incorrect.

**TASK 3: Explore Databricks Storage Options**

**Azure Databricks Storage Options**

1. **DBFS (Databricks File System)**
   * Managed storage layer built into Databricks.
   * Mounts your cloud storage as /dbfs/.
   * Easy for quick data loading and temp storage.



1. **Mounting Azure Storage (Blob or ADLS)**
   * Mount external storage (Blob or ADLS Gen2) to /mnt/your-mount-name.
   * Allows persistent storage with access to raw, bronze, silver, gold layers.
   * Uses dbutils.fs.mount() with access keys or service principals.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Direct Access with ABFS or WASBS URLs**
   * No mount required.
   * Example: "abfss://container@storage.dfs.core.windows.net/"
   * Best for secure, scalable access with Unity Catalog.
2. **External Tables in Data Lake (Delta)**
   * Store Delta tables in ADLS or Blob and register them in Hive or Unity Catalog.
   * Enables scalable data lake architecture (bronze/silver/gold).

**EXERCISE 5 – Advanced Spark Topics and Optimization**

**TASK 1: Implement caching and Broadcasting**

1. Caching a dataframe in memory allows for faster reuse
2. To cache a dataframe run df.cache()  
   A black background with white text

   AI-generated content may be incorrect.
3. If we are joining a large dataframe with a smaller one. We can broadcast the smaller dataframe across all worker nodes to avoid shuffling.  
   A screen shot of a computer code

   AI-generated content may be incorrect.

**TASK 2: Use spark’s advanced concepts**

**1. Use DataFrames and Spark SQL (not RDDs)**

* DataFrames are optimized by **Catalyst** and **Tungsten**, making them much faster.
* Avoid low-level RDDs unless absolutely necessary.

**2. Cache / Persist Smartly**

* Use .cache() or .persist() **only** when you reuse a DataFrame **multiple times**.
* Don’t over-cache — memory is limited!

**3. Broadcast Small Lookup Tables**

* When joining with small datasets (< 100 MB), use:

from pyspark.sql.functions import broadcast  
df\_large.join(broadcast(df\_small), "key")

**4. Filter Early, Select Only Needed Columns**

* Apply .filter() and .select() early to reduce data volume.

df.select("col1", "col2").filter("col1 > 100")

**5. Partition Smartly**

* Use .repartition() to distribute load for wide transformations or large joins.
* Use .coalesce(1) **only** for final small outputs (e.g., exports).

df = df.repartition(10, "some\_column")

**6. Use Delta Lake Format**

* Delta tables are faster for reads, support ACID, schema evolution, and time travel.
* Use OPTIMIZE and ZORDER to improve read performance.

OPTIMIZE my\_table ZORDER BY (col\_name)

**7. Avoid Exploding Joins & Shuffles**

* Shuffles are expensive (joins, groupBy, distinct).
* Repartition wisely to avoid skew.

**8. Monitor with Spark UI**

* Always check stages and tasks in **Spark UI** (or the **Databricks Job Run view**).
* Look out for:
  + Long tasks
  + Skewed stages
  + Too many small files