[Databricks] {CheatSheet}

1. Data Reading and Writing

- **Reading Data from DBFS**: `spark.read.format("csv").load("/FileStore/tables/data.csv")`
- **Writing Data to DBFS**: `df.write.format("parquet").save("/FileStore/tables/output")`
- **Mounting S3 Buckets**: `dbutils.fs.mount("s3a://bucket-name", "/mnt/bucket-name")`
- **Reading Data from Mounted S3 Bucket**: `spark.read.parquet("/mnt/bucket-name/data")`

2. Data Transformation and Processing

- **Creating Temp Views for SQL Queries**: `df.createOrReplaceTempView("tempView")`
- **Running SQL Commands**: `%sql SELECT * FROM tempView WHERE column > value`
- **Converting DataFrame to Pandas**: `pandas_df = df.toPandas()`
- **Creating DataFrame from Pandas**: `spark_df = spark.createDataFrame(pandas_df)`

3. Visualization and Display Functions

- **Displaying DataFrame**: `display(df)`
- **Plotting Graphs**: `display(df)` and use the plot options in the output cell.
- **Visualizing Data with SQL**: `%sql SELECT column1, column2 FROM tempView`
- **Custom Plotting with Matplotlib**: `%python import matplotlib.pyplot as plt; plt.plot(x, y)`

4. Databricks Utilities (dbutils)

- **Listing Files in DBFS**: `dbutils.fs.ls("/FileStore/tables/")`
- **Copying Files in DBFS**: `dbutils.fs.cp("/FileStore/tables/data.csv", "/FileStore/tables/data_copy.csv")`
- **Removing Files from DBFS**: `dbutils.fs.rm("/FileStore/tables/data.csv")`
- **Running Shell Commands**: `%sh ls /dbfs/FileStore/tables/`

5. Spark SQL and DataFrames

By: Waleed Mousa

- **Caching a DataFrame**: `df.cache()`
- **Uncaching a DataFrame**: `df.unpersist()`
- **Explaining Query Plan**: `df.explain()`
- **Aggregating Data**: `df.groupBy("column").count()`

6. Optimization Techniques

- **Broadcast Join Hint**: `df1.join(broadcast(df2), Seq("id"))`
- **Repartitioning Data**: `df.repartition(100)`
- **Caching Tables**: `%sql CACHE TABLE tableName`
- **Z-Ordering for Optimized File Layout**: `df.write.format("parquet").option("zorder", "column").save("/mnt/data/z_ordered_data")`

7. Machine Learning with MLlib

- **Using MLlib for Modeling**: `from pyspark.ml.classification import LogisticRegression; val lr = LogisticRegression()`
- **Model Training**: `val model = lr.fit(trainDF)`
- **Model Prediction**: `val predictions = model.transform(testDF)`
- **Model Evaluation**: `from pyspark.ml.evaluation import MulticlassClassificationEvaluator; val evaluator = MulticlassClassificationEvaluator()`

8. Deep Learning with Databricks

- **Using TensorFlow or PyTorch**: `%pip install tensorflow; %pip install torch`
- **Distributed Training with Horovod**: `import horovod.spark`
- **Loading Data for Deep Learning**: `data = spark.read.format("image").load("/mnt/data/images")`

9. Libraries and Dependencies

- **Installing Python Libraries**: `%pip install numpy pandas`
- **Attaching Libraries to Clusters**: Using Databricks UI to attach libraries to clusters.
- **Using Maven Libraries**: `%scala dbutils.library.install("com.databricks:spark-xml_2.12:0.9.0")`
- **Uninstalling Libraries**: `%scala dbutils.library.uninstall("com.databricks:spark-xml_2.12:0.9.0")`

By: Waleed Mousa

10. Job Scheduling and Automation

- **Creating a Job in Databricks UI**: Using ‘Jobs’ tab to create and schedule notebooks or JARs.
- **Parameterizing Notebooks for Jobs**: `dbutils.widgets.text("name", "")` to create input widgets.
- **Running Jobs via Databricks REST API**: Using `POST /jobs/run-now` endpoint.
- **Monitoring Job Runs**: Using ‘Jobs’ tab to monitor runs and view logs.

11. Data Exploration and Analysis

- **Using Koalas for Pandas-like Syntax**: `%python import databricks.koalas as ks; kdf = ks.DataFrame(df)`
- **Histograms and Boxplots**: `%python display(df.describe())`
- **SQL Queries for Analysis**: `%sql SELECT COUNT(*) FROM tempView GROUP BY column`
- **Correlation Analysis**: `%python df.stat.corr("col1", "col2")`

12. Accessing External Data Sources

- **Connecting to JDBC Data Sources**: `val jdbcDF = spark.read.format("jdbc").option("url", jdbcUrl).option("dbtable", "tableName").load()`
- **Writing to External Databases**: `jdbcDF.write.format("jdbc").option("url", jdbcUrl).option("dbtable", "tableName").save()`

13. Delta Lake Integration

- **Creating a Delta Table**: `df.write.format("delta").save("/delta/tablePath")`
- **Reading from Delta Table**: `val deltaDF = spark.read.format("delta").load("/delta/tablePath")`
- **Time Travel Query**: `df.as("alias").where("versionAsOf = 2")`

14. Working with Structured Streaming

- **Defining a Streaming DataFrame**: `val streamDF = spark.readStream.format("source").load("path")`
- **Writing Stream Output to DBFS**: `streamDF.writeStream.format("delta").start("/delta/streamOutput")`
- **Triggering Streaming Jobs**: `streamDF.writeStream.trigger(Trigger.ProcessingTime("1 minute")).start()`

By: Waleed Mousa

15. Databricks CLI: Basic Operations

- **Installing Databricks CLI**: Run `pip install databricks-cli` in your terminal.
- **Configuring Databricks CLI**: Execute `databricks configure --token`, then enter your Databricks host URL and personal access token.
- **Listing Databricks Workspaces**: `databricks workspace ls`
- **Exporting a Notebook**: `databricks workspace export /Users/name/notebook -o notebook.py`
- **Importing a Notebook to Workspace**: `databricks workspace import -l PYTHON notebook.py /Users/name/notebook`

16. Databricks CLI: Managing Clusters

- **Listing Clusters**: `databricks clusters list`
- **Creating a Cluster**: `databricks clusters create --json '{"name":"clusterName", "spark_version":"7.3.x-scala2.12", "node_type_id":"Standard_D3_v2", "num_workers":2}'`
- **Starting a Cluster**: `databricks clusters start --cluster-id 1234`
- **Terminating a Cluster**: `databricks clusters delete --cluster-id 1234`
- **Getting Cluster Information**: `databricks clusters get --cluster-id 1234`

17. Databricks CLI: Jobs Management

- **Creating a Job**: `databricks jobs create --json 'job_json_content'`
- **Running a Job**: `databricks jobs run-now --job-id 1234`
- **Listing Jobs**: `databricks jobs list`
- **Deleting a Job**: `databricks jobs delete --job-id 1234`
- **Getting Job Status**: `databricks jobs get --job-id 1234`

18. Databricks CLI: Managing DBFS

- **Listing DBFS Files**: `databricks fs ls dbfs:/FileStore/tables/`
- **Copying Files to DBFS**: `databricks fs cp local_file.py dbfs:/FileStore/tables/local_file.py`
- **Copying Files from DBFS to Local**: `databricks fs cp dbfs:/FileStore/tables/data.csv local_data.csv`
- **Removing Files from DBFS**: `databricks fs rm dbfs:/FileStore/tables/data.csv`
- **Making Directories in DBFS**: `databricks fs mkdirs dbfs:/FileStore/new_folder`

By: Waleed Mousa

19. Databricks CLI: Libraries Management

- **Installing a Library on a Cluster**: `databricks libraries install --cluster-id 1234 --maven-coordinates "org.jsoup:jsoup:1.11.3"`
- **Listing Libraries on a Cluster**: `databricks libraries list --cluster-id 1234`
- **Uninstalling a Library from a Cluster**: `databricks libraries uninstall --cluster-id 1234 --maven-coordinates "org.jsoup:jsoup:1.11.3"`
- **Checking Library Status on a Cluster**: `databricks libraries cluster-status --cluster-id 1234`

20. Databricks CLI: Advanced Utilities

- **Running a Spark Submit Job**: `databricks jobs run-now --job-id 1234 --jar-params "param1 param2"`
- **Exporting All Notebooks from a Directory**: `databricks workspace export_dir /Users/name /local_directory`
- **Importing All Notebooks to a Directory**: `databricks workspace import_dir /local_directory /Users/name`
- **Running Databricks SQL Queries**: `databricks sql query -q "SELECT * FROM table LIMIT 10" --cluster-id 1234`

21. Databricks CLI: Token Management

- **Creating a Personal Access Token**: `databricks tokens create --comment "token for automation"`
- **Listing Access Tokens**: `databricks tokens list`
- **Revoking an Access Token**: `databricks tokens revoke --token-id abcd1234`

22. Databricks CLI: Environment Information

- **Listing Available Spark Versions**: `databricks clusters spark-versions`
- **Listing Node Types**: `databricks clusters list-node-types`
- **Listing Available Zones**: `databricks clusters list-zones`

23. Databricks CLI: Workspace Management

- **Listing Folders in Workspace**: `databricks workspace ls /Users/name/folder`
- **Deleting a Notebook from Workspace**: `databricks workspace rm /Users/name/notebook`

By: Waleed Mousa

- **Moving a Notebook in Workspace**: `databricks workspace mv /Users/name/notebook /Users/name/new_notebook`

24. Performance Tuning and Best Practices

- **Data Skewness Handling**: Use techniques like salting to mitigate data skew.
- **Broadcast Hints in Joins**: Use `broadcast(df)` to optimize join operations.
- **Persisting Intermediate DataFrames**: Use `df.persist()` or `df.cache()` for reuse.
- **Optimizing File Sizes and Formats**: Choose efficient file formats like Parquet and optimize file sizes for Spark operations.

25. Advanced Analytics

- **MLflow for Experiment Tracking**: Use MLflow to track experiments, log parameters, and results.
- **Hyperparameter Tuning**: Use MLflow or hyperopt for hyperparameter tuning in machine learning.
- **Advanced UDFs**: Write Scala or Python UDFs for complex transformations.

26. Handling Large Scale Data

- **Partitioning Strategies**: Optimize data partitioning based on the workload.
- **Z-Ordering in Delta Lakes**: Use Z-Ordering to optimize data layout for frequently filtered columns.
- **Optimizing Data Shuffles**: Minimize shuffles and repartition data efficiently.

27. Advanced Data Processing

- **GraphFrames for Graph Analysis**: Leverage GraphFrames for complex graph computations.
- **Geospatial Analysis**: Use libraries like Magellan for geospatial data processing.
- **Handling Complex Nested Structures**: Efficiently process nested JSON or XML data structures.

28. Notebook Workflows

By: Waleed Mousa

- **Running Notebooks from Another Notebook**: `dbutils.notebook.run("notebookPath", timeoutSeconds, parameters)`
- **Parameterizing Notebooks**: Use widgets to create parameterized notebooks.

29. Scheduling and Automation

- **Setting Up Jobs and Schedules**: Configure jobs in Databricks to run notebooks or JARs on a schedule.
- **Dependency Management in Jobs**: Ensure proper management of dependencies in complex workflows.

30. Data Exploration and Visualization

- **Built-in Visualization Tools**: Use Databricks' built-in charts and graphs for quick visualization.
- **Interactive Data Exploration with %sql**: Leverage `%sql` magic command for interactive SQL queries.
- **Third-party Visualization Libraries**: Integrate with libraries like Matplotlib or ggplot for advanced visualizations.

31. Monitoring and Logging

- **Monitoring Cluster Metrics**: Use Ganglia or other tools for monitoring cluster performance.
- **Application Logs Analysis**: Analyze Spark application logs for debugging and optimization.
- **Auditing User Actions**: Leverage auditing capabilities to monitor user activities and data access.

32. Using Databricks for ETL

- **ETL Pipelines**: Build robust ETL pipelines leveraging Spark's capabilities.
- **Incremental Data Loading**: Use Delta Lake for efficient incremental data loading.
- **Data Quality Checks**: Implement data quality checks and validations in ETL workflows.