databricks Spark Test

(https://databricks.com)

```
spark
```

SparkSession - hive

SparkContext

Spark UI

Version
v3.3.2
Master
local[8]
AppName
Databricks Shell

READ THE 'Employees' DATASET IN DATABRICKS COMMUNITY

```
emp_data=spark.read.format("csv")\
    .option("header", "true")\
    .option("inferschema", "false")\
    .option("mode", "FAILFAST")\
    .load("/FileStore/tables/employees.csv")

emp_data.show(5)
```

```
+-----------+----------+---------+--------+------------+---------+--------+------+--------------+----------+-------------+
|EMPLOYEE_ID|FIRST_NAME|LAST_NAME|   EMAIL|PHONE_NUMBER|HIRE_DATE|  JOB_ID|SALARY|COMMISSION_PCT|MANAGER_ID|DEPARTMENT_ID|
+-----------+----------+---------+--------+------------+---------+--------+------+--------------+----------+-------------+
|        198|    Donald| OConnell|DOCONNEL|650.507.9833|21-JUN-07|SH_CLERK|  2600|           - |       124|           50|
|        199|   Douglas|    Grant|  DGRANT|650.507.9844|13-JAN-08|SH_CLERK|  2600|           - |       124|           50|
|        200|  Jennifer|   Whalen| JWHALEN|515.123.4444|17-SEP-03| AD_ASST|  4400|           - |       101|           10|
|        201|   Michael|Hartstein|MHARTSTE|515.123.5555|17-FEB-04|  MK_MAN| 13000|           - |       100|           20|
|        202|       Pat|      Fay|    PFAY|603.123.6666|17-AUG-05|  MK_REP|  6000|           - |       201|           20|
+-----------+----------+---------+--------+------------+---------+--------+------+--------------+----------+-------------+
only showing top 5 rows
```

```
from pyspark.sql.functions import col,avg,round
from pyspark.sql.window import Window
```

What is GROUP BY? Perform in Spark

In Spark, groupBy is an operation that groups the rows of a DataFrame based on one or more columns. It is used in conjunction with aggregation functions to perform computations on each group independently.

```
# Groupby
emp_data = emp_data.withColumn("SALARY", col("SALARY").cast("int"))
emp_data = emp_data.withColumn("DEPARTMENT_ID", col("DEPARTMENT_ID").cast("int"))

# Average salary by department
avg_salary_by_department = emp_data.groupBy("DEPARTMENT_ID").agg(round(avg("SALARY"), 0).alias("Average_Salary"))
avg_salary_by_department.show()
```

```
+-------------+--------------+
|DEPARTMENT_ID|Average_Salary|
+-------------+--------------+
|           20|        9500.0|
|           40|        6500.0|
|          100|        8601.0|
|           10|        4400.0|
|           50|        3722.0|
|           70|       10000.0|
|           90|       19333.0|
|           60|        5760.0|
|          110|       10154.0|
|           30|        4150.0|
+-------------+--------------+
```

What is Count? Perform in Spark

In Spark, Count is an aggregation function that calculates the number of rows in a DataFrame. It is used in conjunction with the groupBy operation to determine the count of rows in each group.

```
# Count of employees whose salary is greater than the average department salary
dept_section = Window().partitionBy("DEPARTMENT_ID")
avg_salary_by_department = emp_data.withColumn("Average_Salary", avg("SALARY").over(dept_section))
avg_salary_by_department.filter(col("SALARY") > col("Average_Salary")).groupBy("DEPARTMENT_ID").count().show()
```

```
+-------------+-----+
|DEPARTMENT_ID|count|
+-------------+-----+
|           20|    1|
|           30|    1|
|           50|    5|
|           60|    2|
|           90|    1|
|          100|    2|
|          110|    1|
+-------------+-----+
```

How many types of modes do we have in Spark?

Spark provides three error handling modes: failfast, dropmalformed, and permissive.

1. **Failfast Mode**:

- In failfast mode, Spark immediately fails the operation if it encounters any malformed records during the reading process. It raises an exception, and the entire job is terminated.
- This mode is suitable when you want to ensure strict adherence to the specified schema and data quality is critical.Commonly used during the early stages of development or testing to quickly identify issues with the data.

2. **Dropmalformed Mode**:

- In dropmalformed mode, Spark ignores and drops any rows that do not conform to the specified schema. It reads as much valid data as possible and discards the malformed records.
- This mode can be applied in scenarios where some degree of data loss is acceptable.

3. **Permissive Mode**:

- In permissive mode, Spark attempts to parse and read all records, including malformed ones. It creates a new column called _corrupt_record to store the raw content of the problematic records.
- This mode is suitable when you want to capture all data, including the malformed records, for further analysis or processing.
- Allows you to inspect and handle the problematic data separately.

What is Clustering in Spark?

Clustering in Spark involves distributing a sizable dataset across multiple nodes in a cluster, allowing parallelized processing for enhanced performance. Spark adopts a master/worker architecture, with a driver program coordinating tasks and workers executing them independently. This distributed computing model enables efficient data processing, leveraging the collective computing power of the cluster's nodes. Each node processes a subset of the data in parallel, promoting scalability. Fault tolerance is achieved through lineage tracking, ensuring resilience in case of node failures. Overall, clustering in Spark optimizes large-scale data processing by harnessing the capabilities of a distributed computing environment.

What is a Table in Spark?

In Spark, "tables" commonly refers to the representation of structured data using Spark SQL's DataFrames and Datasets. A table in Spark is essentially a distributed collection of data organized into named columns, much like a table in a relational database. DataFrames provide a high-level abstraction, offering a familiar tabular structure for data manipulation and analysis. Spark SQL enhances the utility of tables by allowing SQL-like queries on these distributed datasets. This abstraction simplifies complex data operations, making it accessible to users familiar with relational databases. The parallelized processing capabilities of Spark enable efficient and scalable operations on these tables across a cluster of nodes. Tables in Spark serve as a powerful tool for handling large-scale datasets, offering a flexible and intuitive interface for various analytical tasks and data exploration.
