Shwetank Singh
GritSetGrow - GSGLearn.com

SQL AND PYSPARK

Task	SQL	PySpark DataFrame API
Select Columns	SELECT column1, column2 FROM table;	df.select("column1", "column2")
Filter Rows	SELECT * FROM table WHERE condition;	df.filter("condition")
Aggregate Functions	SELECT AVG(column) FROM table;	df.select(F.avg("column"))
Group By	SELECT column, COUNT(*) FROM table GROUP BY column;	df.groupBy("column").count()
Order By	SELECT * FROM table ORDER BY column ASC;	df.orderBy("column", ascending=True)
Join	SELECT * FROM table1 JOIN table2 ON table1.id = table2.id;	df1.join(df2, df1.id == df2.id)
Union	SELECT * FROM table1 UNION SELECT * FROM table2;	df1.union(df2)
Limit	SELECT * FROM table LIMIT 100;	df.limit(100)
Distinct Values	SELECT DISTINCT column FROM table;	df.select("column").distinct()
Adding a New Column	SELECT *, (column1 + column2) AS new_column FROM table;	df.withColumn("new_column", F.col("column1") + F.col("column2"))
Column Alias	SELECT column AS alias_name FROM table;	df.select(F.col("column").alias("alias_name"))
Filtering on Multiple Conditions	SELECT * FROM table WHERE condition1 AND condition2;	df.filter((F.col("condition1")) & (F.col("condition2")))
Subquery	SELECT * FROM (SELECT * FROM table WHERE condition) AS subquery;	df.filter("condition").alias("subquery")
Between	SELECT * FROM table WHERE column BETWEEN val1 AND val2;	df.filter(F.col("column").between("val1", "val2"))
Like	SELECT * FROM table WHERE column LIKE pattern;	df.filter(F.col("column").like("pattern"))
Case When	SELECT CASE WHEN condition THEN result1 ELSE result2 END FROM table;	df.select(F.when(F.col("condition"), "result1").otherwise("result2"))

Shwetank Singh
GritSetGrow - GSGLearn.com

SQL AND PYSPARK

Task	SQL	PySpark DataFrame API
Case When	SELECT CASE WHEN condition THEN result1 ELSE result2 END FROM table;	df.select(F.when(F.col("condition"), "result1").otherwise("result2"))
Cast Data Type	SELECT CAST(column AS datatype) FROM table;	df.select(F.col("column").cast("datatype"))
Count Distinct	SELECT COUNT(DISTINCT column) FROM table;	df.select(F.countDistinct("column"))
Substring	SELECT SUBSTRING(column, start, length) FROM table;	df.select(F.substring("column", start, length))
Concatenate Columns	SELECT CONCAT(column1, column2) AS new_column FROM table;	df.withColumn("new_column", F.concat(F.col("column1"), F.col("column2")))
Average Over Partition	SELECT AVG(column) OVER (PARTITION BY column2) FROM table;	df.withColumn("avg", F.avg("column").over(Window.partitionBy("column2")))
Sum Over Partition	SELECT SUM(column) OVER (PARTITION BY column2) FROM table;	df.withColumn("sum", F.sum("column").over(Window.partitionBy("column2")))
Lead Function	SELECT LEAD(column, 1) OVER (ORDER BY column2) FROM table;	df.withColumn("lead", F.lead("column", 1).over(Window.orderBy("column2")))
Lag Function	SELECT LAG(column, 1) OVER (ORDER BY column2) FROM table;	df.withColumn("lag", F.lag("column", 1).over(Window.orderBy("column2")))
Row Count	SELECT COUNT(*) FROM table;	df.count()
Drop Column	ALTER TABLE table DROP COLUMN column; *(Not directly in SELECT)*	df.drop("column")
Rename Column	ALTER TABLE table RENAME COLUMN column1 TO column2; *(Not directly in SELECT)*	df.withColumnRenamed("column1", "column2")
Change Column Type	ALTER TABLE table ALTER COLUMN column TYPE new_type; *(Not directly in SELECT)*	df.withColumn("column", df["column"].cast("new_type"))

Shwetank Singh
GritSetGrow - GSGLearn.com

SQL AND PYSPARK

Task	SQL	PySpark DataFrame API
Creating a Table from Select	CREATE TABLE new_table AS SELECT * FROM table;	(df.write.format("parquet").saveAsTable("new_table"))
Inserting Selected Data into Table	INSERT INTO table2 SELECT * FROM table1;	(df1.write.insertInto("table2"))
Creating a Table with Specific Columns	CREATE TABLE new_table AS SELECT column1, column2 FROM table;	(df.select("column1", "column2").write.format("parquet").saveAsTable("new_table"))
Aggregate with Alias	SELECT column, COUNT(*) AS count FROM table GROUP BY column;	df.groupBy("column").agg(F.count("*").alias("count"))
Nested Subquery	SELECT * FROM (SELECT * FROM table WHERE condition) sub WHERE sub.condition2;	df.filter("condition").alias("sub").filter("sub.condition2")
Multiple Joins	SELECT * FROM table1 JOIN table2 ON table1.id = table2.id JOIN table3 ON table1.id = table3.id;	df1.join(df2, "id").join(df3, "id")
Cross Join	SELECT * FROM table1 CROSS JOIN table2;	df1.crossJoin(df2)
Group By Having Count Greater Than	SELECT column, COUNT(*) FROM table GROUP BY column HAVING COUNT(*) > 1;	df.groupBy("column").count().filter(F.col("count") > 1)
Alias for Table in Join	SELECT t1.* FROM table1 t1 JOIN table2 t2 ON t1.id = t2.id;	df1.alias("t1").join(df2.alias("t2"), F.col("t1.id") == F.col("t2.id"))
Selecting from Multiple Tables	SELECT t1.column, t2.column FROM table1 t1, table2 t2 WHERE t1.id = t2.id;	df1.join(df2, df1.id == df2.id).select(df1.column, df2.column)
Case When with Multiple Conditions	SELECT CASE WHEN condition THEN 'value1' WHEN condition2 THEN 'value2' ELSE 'value3' END FROM table;	df.select(F.when(F.col("condition"), "value1").when(F.col("condition2"), "value2").otherwise("value3"))

Shwetank Singh
GritSetGrow - GSGLearn.com

SQL AND PYSPARK

Task	SQL	PySpark DataFrame API
Extracting Date Parts	SELECT EXTRACT(YEAR FROM date_column) FROM table;	df.select(F.year(F.col("date_column")))
Inequality Filtering	SELECT * FROM table WHERE column != 'value';	df.filter(df.column != 'value')
In List	SELECT * FROM table WHERE column IN ('value1', 'value2');	df.filter(df.column.isin('value1', 'value2'))
Not In List	SELECT * FROM table WHERE column NOT IN ('value1', 'value2');	df.filter(~df.column.isin('value1', 'value2'))
Null Values	SELECT * FROM table WHERE column IS NULL;	df.filter(df.column.isNull())
Not Null Values	SELECT * FROM table WHERE column IS NOT NULL;	df.filter(df.column.isNotNull())
String Upper Case	SELECT UPPER(column) FROM table;	df.select(F.upper(df.column))
String Lower Case	SELECT LOWER(column) FROM table;	df.select(F.lower(df.column))
String Length	SELECT LENGTH(column) FROM table;	df.select(F.length(df.column))
Trim String	SELECT TRIM(column) FROM table;	df.select(F.trim(df.column))
Left Trim String	SELECT LTRIM(column) FROM table;	df.select(F.ltrim(df.column))
Right Trim String	SELECT RTRIM(column) FROM table;	df.select(F.rtrim(df.column))
String Replace	SELECT REPLACE(column, 'find', 'replace') FROM table;	df.select(F.regexp_replace(df.column, 'find', 'replace'))
Substring Index	SELECT SUBSTRING_INDEX(column, 'delim', count) FROM table;	df.select(F.expr("split(column, 'delim')[count-1]")) *(Assuming 1-based index)*
Date Difference	SELECT DATEDIFF('date1', 'date2') FROM table;	df.select(F.datediff(F.col('date1'), F.col('date2')))
Add Months to Date	SELECT ADD_MONTHS(date_column, num_months) FROM table;	df.select(F.add_months(df.date_column, num_months))

Shwetank Singh
GritSetGrow - GSGLearn.com

SQL AND PYSPARK

Task	SQL	PySpark DataFrame API
First Value in Group	SELECT FIRST_VALUE(column) OVER (PARTITION BY column2) FROM table;	df.withColumn("first_val", F.first("column").over(Window.partitionBy("column2")))
Last Value in Group	SELECT LAST_VALUE(column) OVER (PARTITION BY column2) FROM table;	df.withColumn("last_val", F.last("column").over(Window.partitionBy("column2")))
Row Number Over Partition	SELECT ROW_NUMBER() OVER (PARTITION BY column ORDER BY column) FROM table;	df.withColumn("row_num", F.row_number().over(Window.partitionBy("column").orderBy("column")))
Rank Over Partition	SELECT RANK() OVER (PARTITION BY column ORDER BY column) FROM table;	df.withColumn("rank", F.rank().over(Window.partitionBy("column").orderBy("column")))
Dense Rank Over Partition	SELECT DENSE_RANK() OVER (PARTITION BY column ORDER BY column) FROM table;	df.withColumn("dense_rank", F.dense_rank().over(Window.partitionBy("column").orderBy("column")))
Count Rows	SELECT COUNT(*) FROM table;	df.count()
Mathematical Operations	SELECT column1 + column2 FROM table;	df.select(F.col("column1") + F.col("column2"))
String Concatenation	SELECT column1 || column2 AS new_column FROM table;	df.withColumn("new_column", F.concat_ws("||", F.col("column1"), F.col("column2")))
Find Minimum Value	SELECT MIN(column) FROM table;	df.select(F.min("column"))
Find Maximum Value	SELECT MAX(column) FROM table;	df.select(F.max("column"))
Removing Duplicates	SELECT DISTINCT * FROM table;	df.distinct()

Shwetank Singh
GritSetGrow - GSGLearn.com

SQL AND PYSPARK

Task	SQL	PySpark DataFrame API
Left Join	SELECT * FROM table1 LEFT JOIN table2 ON table1.id = table2.id;	df1.join(df2, df1.id == df2.id, "left")
Right Join	SELECT * FROM table1 RIGHT JOIN table2 ON table1.id = table2.id;	df1.join(df2, df1.id == df2.id, "right")
Full Outer Join	SELECT * FROM table1 FULL OUTER JOIN table2 ON table1.id = table2.id;	df1.join(df2, df1.id == df2.id, "outer")
Group By with Having	SELECT column, COUNT(*) FROM table GROUP BY column HAVING COUNT(*) > 10;	df.groupBy("column").count().filter(F.col("count") > 10)
Round Decimal Values	SELECT ROUND(column, 2) FROM table;	df.select(F.round("column", 2))
Get Current Date	SELECT CURRENT_DATE();	df.select(F.current_date())
Date Addition	SELECT DATE_ADD(date_column, 10) FROM table;	df.select(F.date_add(F.col("date_column"), 10))
Date Subtraction	SELECT DATE_SUB(date_column, 10) FROM table;	df.select(F.date_sub(F.col("date_column"), 10))
Extract Year from Date	SELECT YEAR(date_column) FROM table;	df.select(F.year(F.col("date_column")))
Extract Month from Date	SELECT MONTH(date_column) FROM table;	df.select(F.month(F.col("date_column")))
Extract Day from Date	SELECT DAY(date_column) FROM table;	df.select(F.dayofmonth(F.col("date_column")))
Sorting Descending	SELECT * FROM table ORDER BY column DESC;	df.orderBy(F.col("column").desc())
Group By Multiple Columns	SELECT col1, col2, COUNT(*) FROM table GROUP BY col1, col2;	df.groupBy("col1", "col2").count()
Conditional Column Update	UPDATE table SET column1 = CASE WHEN condition THEN 'value1' ELSE 'value2' END;	df.withColumn("column1", F.when(F.col("condition"), "value1").otherwise("value2"))

THANK you