Data Engineering 101 - SQL vs PySpark 80 + comparisons

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SELECT COLUMNS

SQL

```
SELECT column1, column2
FROM table;
```

PYSPARK

```
df.select("column1", "column2")
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FILTER ROWS

SQL

```
SELECT * FROM table
WHERE condition;
```

PYSPARK

```
df.filter("condition")
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

AGGREGATE FUNCTIONS

SQL

```
SELECT AVG(column)
FROM table;
```

3

PYSPARK

```
df.select(F.avg("column"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

GROUP BY

SQL

```
SELECT column, COUNT(*)
FROM table
GROUP BY column;
```

PYSPARK

```
df.groupBy("column").count()
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ORDER BY

SQL

```
SELECT *
FROM table
ORDER BY column ASC;
```

PYSPARK

```
df.orderBy("column",
ascending=True)
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

JOIN

SQL

```
SELECT * FROM table1
JOIN table2
ON table1.id = table2.id;
```

PYSPARK

```
df1.join(df2, df1.id == df2.id)
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

UNION

SQL

```
SELECT * FROM table1
UNION
SELECT * FROM table2;
```

PYSPARK

```
df1.union(df2)
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

LIMIT

SQL

```
SELECT *
FROM table
LIMIT 100;
```

PYSPARK

```
df.limit(100)
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DISTINCT VALUES

SQL

```
SELECT DISTINCT column
FROM table;
```

PYSPARK

```
df.select("column").distinct()
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ADDING A NEW COLUMN

SQL

```
SELECT *, (column1 + column2)
AS new_column
FROM table;
```

PYSPARK

```
df.withColumn("new_column",
F.col("column1") +
F.col("column2"))
```

COLUMN ALIAS

SQL

```
SELECT column AS alias_name
FROM table;
```

PYSPARK

```
df.select(F.col("column").alias("
alias_name"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FILTERING ON MULTIPLE CONDITIONS

SQL

```
SELECT * FROM table
WHERE
condition1 AND condition2;
```

PYSPARK

```
df.filter((F.col("condition1")) &
(F.col("condition2")))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

SUBQUERY

SQL

```
SELECT * FROM
(SELECT * FROM table
WHERE condition) AS subquery;
```

PYSPARK

```
df.filter("condition").alias("subquery")
```

BETWEEN

SQL

```
SELECT * FROM table
WHERE column
BETWEEN val1 AND val2;
```

PYSPARK

```
df.filter(F.col("column") \
.between("val1", "val2"))
```

LIKE

SQL

```
SELECT * FROM table
WHERE column LIKE pattern;
```

PYSPARK

```
df.filter(F.col("column") \
.like("pattern"))
```

CASE WHEN

SQL

```
SELECT CASE
WHEN condition THEN result1
ELSE result2 END
FROM table;
```

PYSPARK

```
df.select(F.when(F.col("conditio
n"), "result1") \
.otherwise("result2"))
```

CAST DATA TYPE

SQL

```
SELECT
CAST(column AS datatype)
FROM table;
```

PYSPARK

```
df.select(F.col("column") \
.cast("datatype"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

COUNT DISTINCT

SQL

```
SELECT
COUNT(DISTINCT column)
FROM table;
```

PYSPARK

```
df.select(F.countDistinct("colu
mn"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SUBSTRING

SQL

```
SELECT SUBSTRING(column,
start, length)
FROM table;
```

PYSPARK

```
df.select(F.substring("column",
start, length))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CONCATENATE COLUMNS

SQL

```
SELECT
CONCAT(column1, column2) AS
new_column
FROM table;
```

PYSPARK

```
df.withColumn("new_column",
F.concat(F.col("column1"),
F.col("column2")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

AVERAGE OVER PARTITION

SQL

```
SELECT AVG(column)
OVER (PARTITION BY column2)
FROM table;
```

PYSPARK

```
df.withColumn("avg", F.avg("column") \
.over(Window.partitionBy("column2")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SUM OVER PARTITION

SQL

```
SELECT SUM(column)
OVER (PARTITION BY column2)
FROM table;
```

PYSPARK

```
df.withColumn("sum", F.sum("column") \
.over(Window.partitionBy("column2")))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

LEAD FUNCTION

SQL

```
SELECT LEAD(column, 1)
OVER (ORDER BY column2)
FROM table;
```

PYSPARK

```
df.withColumn("lead",
F.lead("column", 1) \
.over(Window.orderBy("column2")))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

LAG FUNCTION

SQL

```
SELECT LAG(column, 1)
OVER (ORDER BY column2)
FROM table;
```

PYSPARK

```
df.withColumn("lag", F.lag("column", 1) \
.over(Window.orderBy("column2")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ROW COUNT

SQL

```
SELECT COUNT(*)
FROM table;
```

PYSPARK

```
df.count()
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DROP COLUMN

SQL

```
ALTER TABLE table
DROP COLUMN column;
```

PYSPARK

```
df.drop("column")
```

Shwetank Singh
GritSetGrow - GSGLearn.com

RENAME COLUMN

SQL

```
ALTER TABLE table RENAME
COLUMN column1 TO column2;
```

PYSPARK

```
df.withColumnRenamed("colu
mn1", "column2")
```

CHANGE COLUMN TYPE

SQL

```
ALTER TABLE table
ALTER COLUMN column TYPE
new_type;
```

PYSPARK

```
df.withColumn("column",
df["column"] \
.cast("new_type"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CREATING A TABLE FROM SELECT

SQL

```
CREATE TABLE new_table
AS SELECT * FROM table;
```

PYSPARK

```
(df.write.format("parquet") \
.saveAsTable("new_table"))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

INSERTING SELECTED DATA INTO TABLE

SQL

```
INSERT INTO table2
SELECT * FROM table1;
```

PYSPARK

```
(df1.write.insertInto("table2"))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

CREATING A TABLE WITH SPECIFIC COLUMNS

SQL

```
CREATE TABLE new_table
AS
SELECT column1, column2
FROM table;
```

PYSPARK

```
(df.select("column1", "column2") \
.write.format("parquet") \
.saveAsTable("new_table"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

AGGREGATE WITH ALIAS

SQL

```
SELECT column,
COUNT(*) AS count
FROM table
GROUP BY column;
```

PYSPARK

```
df.groupBy("column") \
.agg(F.count("*") \
.alias("count"))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

NESTED SUBQUERY

SQL

```
SELECT * FROM
(SELECT *
FROM table
WHERE condition) sub
WHERE sub.condition2;
```

PYSPARK

```
df.filter("condition") \
.alias("sub") \
.filter("sub.condition2")
```

Shwetank Singh
GritSetGrow - GSGLearn.com

MULTIPLE JOINS

SQL

```
SELECT * FROM table1
JOIN table2
ON table1.id = table2.id
JOIN table3
ON table1.id = table3.id;
```

PYSPARK

```
df1.join(df2, "id").join(df3, "id")
```

Shwetank Singh
GritSetGrow - GSGLearn.com

CROSS JOIN

SQL

```
SELECT *
FROM table1
CROSS JOIN table2;
```

PYSPARK

```
df1.crossJoin(df2)
```

Shwetank Singh
GritSetGrow - GSGLearn.com

GROUP BY HAVING COUNT GREATER THAN

SQL

```
SELECT column,
COUNT(*)
FROM table
GROUP BY column
HAVING COUNT(*) > 1;
```

PYSPARK

```
df.groupBy("column") \
.count() \
.filter(F.col("count") > 1)
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ALIAS FOR TABLE IN JOIN

SQL

```
SELECT t1.*
FROM table1 t1
JOIN table2 t2
ON t1.id = t2.id;
```

PYSPARK

```
df1.alias("t1") \
.join(df2.alias("t2"), F.col("t1.id")
== F.col("t2.id"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SELECTING FROM MULTIPLE TABLES

SQL

```
SELECT t1.column, t2.column
FROM table1 t1, table2 t2
WHERE t1.id = t2.id;
```

PYSPARK

```
df1.join(df2, df1.id == df2.id) \
.select(df1.column, df2.column)
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CASE WHEN WITH MULTIPLE CONDITIONS

SQL

```
SELECT CASE WHEN
condition THEN 'value1'
WHEN condition2 THEN 'value2' ELSE
'value3'
END
FROM table;
```

PYSPARK

```
df.select(F.when(F.col("condition"),
"value1").when(F.col("condition2"),
"value2").otherwise("value3"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

EXTRACTING DATE PARTS

SQL

```
SELECT EXTRACT(YEAR FROM date_column)
FROM table;
```

PYSPARK

```
df.select(F.year(F.col("date_column")))
```

INEQUALITY FILTERING

SQL

```
SELECT *
FROM table
WHERE column != 'value';
```

PYSPARK

```
df.filter(df.column != 'value')
```

Shwetank Singh
GritSetGrow - GSGLearn.com

IN LIST

SQL

```
SELECT *
FROM table
WHERE column IN ('value1',
'value2');
```

PYSPARK

```
df.filter(df.column.isin('value1',
'value2'))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

NOT IN LIST

SQL

```
SELECT *
FROM table
WHERE column NOT IN ('value1',
'value2');
```

PYSPARK

```
df.filter(~df.column.isin('value1',
'value2'))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

NULL VALUES

SQL

```
SELECT * FROM
table
WHERE column IS NULL;
```

PYSPARK

```
df.filter(df.column.isNull())
```

Shwetank Singh
GritSetGrow - GSGLearn.com

NOT NULL VALUES

SQL

```
SELECT *
FROM table
WHERE column IS NOT NULL;
```

PYSPARK

```
df.filter(df.column.isNotNull())
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

STRING UPPER CASE

SQL

```
SELECT UPPER(column)
FROM table;
```

PYSPARK

```
df.select(F.upper(df.column))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

STRING LOWER CASE

SQL

```
SELECT LOWER(column)
FROM table;
```

PYSPARK

```
df.select(F.lower(df.column))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

STRING LENGTH

SQL

```
SELECT LENGTH(column)
FROM table;
```

48

PYSPARK

```
df.select(F.length(df.column))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

TRIM STRING

SQL

```
SELECT TRIM(column)
FROM table;
```

PYSPARK

```
df.select(F.trim(df.column))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

LEFT TRIM STRING

SQL

```
SELECT LTRIM(column)
FROM table;
```

PYSPARK

```
df.select(F.ltrim(df.column))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

RIGHT TRIM STRING

SQL

```
SELECT RTRIM(column)
FROM table;
```

PYSPARK

```
df.select(F.rtrim(df.column))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

STRING REPLACE

SQL

```
SELECT REPLACE(column, 'find', 'replace')
FROM table;
```

PYSPARK

```
df.select(F.regexp_replace(df.column, 'find', 'replace'))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

SUBSTRING INDEX

SQL

```
SELECT
SUBSTRING_INDEX(column,
'delim', count)
FROM table;
```

PYSPARK

```
df.select(F.expr("split(column,
'delim')[count-1]"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DATE DIFFERENCE

SQL

```
SELECT DATEDIFF('date1', 'date2')
FROM table;
```

PYSPARK

```
df.select(F.datediff(F.col('date1'
), F.col('date2')))
```

ADD MONTHS TO DATE

SQL

```
SELECT
ADD_MONTHS(date_column,
num_months)
FROM table;
```

PYSPARK

```
df.select(F.add_months
(df.date_column, num_months))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FIRST VALUE IN GROUP

SQL

```
SELECT FIRST_VALUE(column)
OVER (PARTITION BY column2)
FROM table;
```

PYSPARK

```
df.withColumn("first_val",
F.first("column") \
.over(Window.partitionBy("column2")))
```

LAST VALUE IN GROUP

SQL

```
SELECT LAST_VALUE(column)
OVER (PARTITION BY column2)
FROM table;
```

PYSPARK

```
df.withColumn("last_val",
F.last("column") \
.over(Window.partitionBy("column2")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

ROW NUMBER OVER PARTITION

SQL

```
SELECT ROW_NUMBER()
OVER (PARTITION BY column
ORDER BY column)
FROM table;
```

PYSPARK

```
df.withColumn("row_num",
F.row_number() \
.over(Window.partitionBy("column") \
.orderBy("column")))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

RANK OVER PARTITION

SQL

```
SELECT RANK()
OVER (PARTITION BY column
ORDER BY column)
FROM table;
```

PYSPARK

```
df.withColumn("rank",
F.rank().over(Window.partitionBy
("column").orderBy("column")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DENSE RANK OVER PARTITION

SQL

```
SELECT DENSE_RANK()
OVER (PARTITION BY column
ORDER BY column)
FROM table;
```

PYSPARK

```
df.withColumn("dense_rank",
F.dense_rank().over(Window.partitio
nBy("column").orderBy("column")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

COUNT ROWS

SQL

```
SELECT COUNT(*)
FROM table;
```

PYSPARK

```
df.count()
```

Shwetank Singh
GritSetGrow - GSGLearn.com

MATHEMATICAL OPERATIONS

SQL

```
SELECT column1 + column2
FROM table;
```

PYSPARK

```
df.select(F.col("column1") +
F.col("column2"))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

STRING CONCATENATION

SQL

```
SELECT column1 | column2
AS new_column
FROM table;
```

PYSPARK

```
df.withColumn("new_column",
F.concat_ws("|",
F.col("column1"),
F.col("column2")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FIND MINIMUM VALUE

SQL

```
SELECT MIN(column)
FROM table;
```

PYSPARK

```
df.select(F.min("column"))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

FIND MAXIMUM VALUE

SQL

```
SELECT MAX(column)
FROM table;
```

PYSPARK

```
df.select(F.max("column"))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

REMOVING DUPLICATES

SQL

```
SELECT DISTINCT *
FROM table;
```

PYSPARK

```
df.distinct()
```

Shwetank Singh
GritSetGrow - GSGLearn.com

LEFT JOIN

SQL

```
SELECT * FROM table1
LEFT JOIN table2
ON table1.id = table2.id;
```

PYSPARK

```
df1.join(df2, df1.id == df2.id,
"left")
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

RIGHT JOIN

SQL

```
SELECT * FROM table1
RIGHT JOIN table2
ON table1.id = table2.id;
```

PYSPARK

```
df1.join(df2, df1.id == df2.id,
"right")
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

FULL OUTER JOIN

SQL

```
SELECT * FROM table1
FULL OUTER
JOIN table2
ON table1.id = table2.id;
```

PYSPARK

```
df1.join(df2, df1.id == df2.id,
"outer")
```

Shwetank Singh
GritSetGrow - GSGLearn.com

GROUP BY WITH HAVING

SQL

```
SELECT column, COUNT(*)
FROM table
GROUP BY column
HAVING COUNT(*) > 10;
```

PYSPARK

```
df.groupBy("column") \
.count() \
.filter(F.col("count") > 10)
```

Shwetank Singh
GritSetGrow - GSGLearn.com

ROUND DECIMAL VALUES

SQL

```
SELECT ROUND(column, 2)
FROM table;
```

PYSPARK

```
df.select(F.round("column", 2))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

GET CURRENT DATE

SQL

```
SELECT CURRENT_DATE();
```

PYSPARK

```
df.select(F.current_date())
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

DATE ADDITION

SQL

```
SELECT
DATE_ADD(date_column, 10)
FROM table;
```

PYSPARK

```
df.select(F.date_add(F.col("dat
e_column"), 10))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

DATE SUBTRACTION

SQL

```
SELECT
DATE_SUB(date_column, 10)
FROM table;
```

PYSPARK

```
df.select(F.date_sub(F.col("dat
e_column"), 10))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

EXTRACT YEAR FROM DATE

SQL

```
SELECT YEAR(date_column)
FROM table;
```

PYSPARK

```
df.select(F.year(F.col("date_column")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

EXTRACT MONTH FROM DATE

SQL

```
SELECT MONTH(date_column)
FROM table;
```

PYSPARK

```
df.select(F.month(F.col("date_
column")))
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

EXTRACT DAY FROM DATE

SQL

```
SELECT DAY(date_column)
FROM table;
```

PYSPARK

```
df.select(F.dayofmonth(F.col("d
ate_column")))
```

Shwetank Singh
GritSetGrow - GSGLearn.com

SORTING DESCENDING

SQL

```
SELECT *
FROM table
ORDER BY column DESC;
```

PYSPARK

```
df.orderBy(F.col("column").desc())
```

GROUP BY MULTIPLE COLUMNS

SQL

```
SELECT col1, col2, COUNT(*)
FROM table
GROUP BY col1, col2;
```

PYSPARK

```
df.groupBy("col1", "col2") \
.count()
```

Feel free to follow me for more content

Shwetank Singh
GritSetGrow - GSGLearn.com

CONDITIONAL COLUMN UPDATE

SQL

```
UPDATE table
SET column1 = CASE
WHEN condition
THEN 'value1' ELSE 'value2' END;
```

PYSPARK

```
df.withColumn("column1",
F.when(F.col("condition"),
"value1").otherwise("value2"))
```

THANK YOU