Data Exploration

Spark Session Initialization

In [ ]:

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Data Exploration") \
    .config('spark.jars.packages', 'mysql:mysql-connector-java:8.0.32') \
    .getOrCreate()
```

Data Loading

In [ ]:

```
mysql_host = "sql8.freesqldatabase.com"
mysql_port = "3306"
mysql_database = "sql8696474"
mysql_username = "sql8696474"
mysql_password = "2gVPjJi7xV"
mysql_table = "jendouba_sales"

jdbc_url = f"jdbc:mysql://{mysql_host}:{mysql_port}/{mysql_database}"

mysql_properties = {
    "user": mysql_username,
    "password": mysql_password,
    "driver": "com.mysql.cj.jdbc.Driver"
}

df = spark.read.jdbc(url=jdbc_url, table=mysql_table, properties=mysql_properties)
```

Displaying the first few rows of the DataFrame

In [ ]:

```
df.show(n=3, truncate=True)
```

```
+---------+------+------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|          pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
+---------+------+------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 3 rows
```

Displaying the schema of the DataFrame

In [ ]:

```
df.printSchema()
```

```
root
 |-- client_id: integer (nullable = true)
 |-- pos_id: integer (nullable = true)
 |-- pos_name: string (nullable = true)
 |-- article: string (nullable = true)
 |-- quantity: double (nullable = true)
 |-- price: double (nullable = true)
 |-- total: double (nullable = true)
 |-- sale_type: string (nullable = true)
 |-- payment_mode: string (nullable = true)
 |-- sale_time: timestamp (nullable = true)
```

Calculating Descriptive Statistics

In [ ]:

```
df.describe("total").show()
```

```
+-------+-----------------+
|summary|            total|
+-------+-----------------+
|  count|             9672|
|   mean|67.66650124069496|
| stddev|51.39601950189552|
|    min|              1.5|
|    max|            240.0|
+-------+-----------------+
```

Calculation of Correlation

In [ ]:

```
from pyspark.sql.functions import corr

df.select(corr("quantity", "total")).show()
```

```
+---------------------+
|corr(quantity, total)|
+---------------------+
|   0.7167740253074978|
+---------------------+
```

Counting the number of rows and distinct values

Counting the rows

In [ ]:
```
df.count()
```

Out[ ]:
```
9876
```

Distinct values

In [ ]:
```
df.select("article").distinct().count()
```

Out[ ]:
```
23
```

In [ ]:
```
df.select("article").distinct().show(23)
```

```
+----------------+
|         article|
+----------------+
|      Cream Puff|
|          Muffin|
|       Pecan Pie|
|        Napoleon|
|           Scone|
|            null|
|      Cherry Pie|
|      Cheesecake|
|  Apple Turnover|
|        Baguette|
|         Palmier|
|       Croissant|
|       Bear Claw|
|Chocolate Eclair|
|         Cupcake|
|   Key Lime Tart|
| Red Velvet Cake|
|       Lemon Bar|
|         Strudel|
|Blueberry Muffin|
|   Cinnamon Roll|
|      Fruit Tart|
|   Danish Pastry|
+----------------+
```

Distinct rows

In [ ]:
```
Distinct_Df = df.distinct()
```

In [ ]:
```
df.count()
```

Out[ ]:
```
9876
```

In [ ]:
```
Distinct_Df.count()
```

Out[ ]:
```
9410
```

Search and Removal of Duplicates and Missing Values

Search and Removal of Duplicates

Searching for Duplicate Values

In [ ]:
```
from pyspark.sql.functions import col

df.groupBy("article").count().where(col("count") > 1).show(5)
```

```
+----------+-----+
|   article|count|
+----------+-----+
|Cream Puff|  429|
|    Muffin|  427|
| Pecan Pie|  411|
|  Napoleon|  418|
|     Scone|  467|
+----------+-----+
only showing top 5 rows
```

In [ ]:

```
from pyspark.sql.functions import desc,  col

duplicate_values = df.groupBy("pos_id","pos_name","article","sale_time").count().where(col("count") > 1).orderBy(desc("count")).show(5)
```

```
+------+-------------------+-------+-------------------+-----+
|pos_id|           pos_name|article|          sale_time|count|
+------+-------------------+-------+-------------------+-----+
|     4|   Jendouba_Tabarka|   null|2024-04-03 22:59:54|   54|
|     3|Jendouba_Ain_Drahem|   null|2024-04-03 22:59:54|   54|
|     2|  Jendouba_Bousalem|   null|2024-04-03 22:59:54|   48|
|     1|    Jendouba_Center|   null|2024-04-03 22:59:54|   48|
|     3|Jendouba_Ain_Drahem|Palmier|2024-04-03 22:53:30|    7|
+------+-------------------+-------+-------------------+-----+
only showing top 5 rows
```

Removing Duplicate Values

In [ ]:

```
new_df = df.dropDuplicates(["pos_id","pos_name","article","sale_time"])
```

In [ ]:

```
df.count()
```

Out[ ]:

```
9876
```

In [ ]:

```
new_df.count()
```

Out[ ]:

```
8894
```

Searching for Duplicate Rows

In [ ]:

```
from pyspark.sql.functions import col

df.groupBy(df.columns).count().where(col("count") > 1).show(5)
```

```
+---------+------+-------------------+----------+--------+-----+-----+---------+------------+-------------------+-----+
|client_id|pos_id|           pos_name|   article|quantity|price|total|sale_type|payment_mode|          sale_time|count|
+---------+------+-------------------+----------+--------+-----+-----+---------+------------+-------------------+-----+
|    29402|     3|Jendouba_Ain_Drahem|Cherry Pie|     5.0|  9.5| 47.5|livraison|      online|2023-06-26 16:39:41|    2|
|     null|     2|  Jendouba_Bousalem|      null|    null| null| null|livraison|      online|2024-04-03 22:59:54|   22|
|    72249|     4|   Jendouba_Tabarka| Croissant|    10.0|  1.5| 15.0|livraison|      online|2023-03-10 03:01:41|    2|
|     2502|     3|Jendouba_Ain_Drahem| Croissant|    14.0|  1.5| 21.0|   direct|        cash|2023-10-25 17:55:03|    2|
|    71085|     4|   Jendouba_Tabarka|Cherry Pie|    20.0|  9.5|190.0|   direct|        cash|2023-04-26 17:12:23|    2|
+---------+------+-------------------+----------+--------+-----+-----+---------+------------+-------------------+-----+
only showing top 5 rows
```

In [ ]:

```
from pyspark.sql.functions import col

df.groupBy(df.columns).count().where(col("count") > 1).count()
```

Out[ ]:

```
285
```

Removing Duplicate Rows

In [ ]:

```
new_df = df.dropDuplicates()
```

In [ ]:

```
df.count()
```

Out[ ]:

```
9876
```

In [ ]:

```
new_df.count()
```

Out[ ]:

```
9410
```

Searching for Missing Values

In [ ]:

```
from pyspark.sql.functions import col

df.filter(col("article").isNull()).show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     2|  Jendouba_Bousalem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

In [ ]:

```
from pyspark.sql.functions import col

df.filter(col("article").isNull()).count()
```

Out[ ]:

```
204
```

Deleting Rows Containing Missing Values

Deleting Rows Containing Null Values

Method 1 **dropna**

```
In [ ]: cleaned_df = df.dropna(how="any")
```

```
In [ ]: cleaned_df.filter(col("article").isNull()).show()
```

```
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
|client_id|pos_id|pos_name|article|quantity|price|total|sale_type|payment_mode|sale_time|
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
```

Method 2 **na.drop**

```
In [ ]: cleaned_df = df.na.drop(how="any")
```

```
In [ ]: cleaned_df.filter(col("article").isNull()).show()
```

```
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
|client_id|pos_id|pos_name|article|quantity|price|total|sale_type|payment_mode|sale_time|
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
```

Deleting Rows Containing Null Values in Specific Cloumns "Article" and "Quantity"

```
In [ ]: cleaned_df = df.na.drop(subset=["article", "quantity"])
```

```
In [ ]: cleaned_df.filter(col("article").isNull()).show()
```

```
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
|client_id|pos_id|pos_name|article|quantity|price|total|sale_type|payment_mode|sale_time|
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
```

Replacing Missing (Null) Values with a Specific Value

Method 1: **na.fill**

Test 1 For Numerical Data

```
In [ ]: filled_df = df.na.fill(0)
```

```
In [ ]: filled_df.filter(col("article").isNull()).show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|        0|     3|Jendouba_Ain_Drahem|   null|     0.0|  0.0|  0.0|livraison|      online|2024-04-03 22:59:54|
|        0|     3|Jendouba_Ain_Drahem|   null|     0.0|  0.0|  0.0|livraison|      online|2024-04-03 22:59:54|
|        0|     3|Jendouba_Ain_Drahem|   null|     0.0|  0.0|  0.0|livraison|      online|2024-04-03 22:59:54|
|        0|     3|Jendouba_Ain_Drahem|   null|     0.0|  0.0|  0.0|livraison|      online|2024-04-03 22:59:54|
|        0|     2|  Jendouba_Bousalem|   null|     0.0|  0.0|  0.0|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Test 2 For Numerical And Categorical Data

```
In [ ]: filled_df = df.na.fill("0")
```

```
In [ ]: filled_df.filter(col("article").isNull()).show(5)
```

```
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
|client_id|pos_id|pos_name|article|quantity|price|total|sale_type|payment_mode|sale_time|
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
+---------+------+--------+-------+--------+-----+-----+---------+------------+---------+
```

Method 2: **fillna**

```
In [ ]: filled_df = df.fillna(0)
```

Column Selection

```
In [ ]: df.select("Total").show(n=10, truncate=True)
```

```
+-----+
|Total|
+-----+
| 45.6|
|136.0|
| 28.0|
| 90.0|
| 49.0|
| 15.0|
|190.0|
| 64.0|
|100.0|
| 72.0|
+-----+
only showing top 10 rows
```

Column Renaming

Method 1: Using the "withColumnRenamed()" method

In [ ]:
```
renamed_column_df = df.withColumnRenamed("article", "product")
renamed_column_df.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         product|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Method 2: Using the "alias()" method

In [ ]:
```
aliased_column_df = df.select("pos_id", "pos_name", df["article"].alias("product"), "quantity", "price", "total", "sale_type", "payment_mode
aliased_column_df.show(5)
```

```
+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|pos_id|           pos_name|         product|quantity|price|total|sale_type|payment_mode|          sale_time|
+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Creating New Columns

Example 1: Creating a New column by multiplying two existing columns

In [ ]:
```
from pyspark.sql.functions import col

new_df1 = df.withColumn("total_price", col("quantity") * col("price"))
new_df1.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|       total_price|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|45.599999999999994|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|             136.0|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|              28.0|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|              90.0|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|              49.0|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+------------------+
only showing top 5 rows
```

Example 2: Creating a New column by concatenating two existing columns

In [ ]:
```
from pyspark.sql.functions import concat, lit

new_df2 = df.withColumn("Concatenated_Column", concat(df['quantity'], lit(" * "), df['price']))
new_df2.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Concatenated_Column|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|         12.0 * 3.8|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|         20.0 * 6.8|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|         14.0 * 2.0|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|         15.0 * 6.0|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|         14.0 * 3.5|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
only showing top 5 rows
```

Example 3: Creating a New Column Using Conditions "When"

In [ ]:

```python
from pyspark.sql.functions import when

new_df3 = df.withColumn("is_high_quantity", when(df['quantity'] > 10, 1).otherwise(0))
new_df3.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|is_high_quantity|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|               1|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|               1|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|               1|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|               1|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|               1|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
only showing top 5 rows
```

Dropping Column

In [ ]:

```python
df_without_total = df.drop("total")
df_without_total.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+---------+------------+-------------------+
only showing top 5 rows
```

Value Replacement

In [ ]:

```python
from pyspark.sql.functions import regexp_replace

df_replace = df.withColumn("article", regexp_replace("article", " ", "_"))
df_replace.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry_Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear_Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon_Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Value Transformation

Transforming values to lowercase

In [ ]:

```python
from pyspark.sql.functions import lower

df_lower = df.withColumn("article_lower", lower(df['article']))
df_lower.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|   article_lower|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|blueberry muffin|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|       bear claw|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|        baguette|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|       lemon bar|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|          muffin|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
only showing top 5 rows
```

Transforming Values to Uppercase

```
In [ ]: from pyspark.sql.functions import upper

        df_upper = df.withColumn("pos_name_upper", upper(df['pos_name']))
        df_upper.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|     pos_name_upper|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|JENDOUBA_AIN_DRAHEM|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|JENDOUBA_AIN_DRAHEM|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|JENDOUBA_AIN_DRAHEM|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|JENDOUBA_AIN_DRAHEM|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|JENDOUBA_AIN_DRAHEM|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
only showing top 5 rows
```

Capitalizing the First Letter of Each Word (Example: Firstname Lastname)

```
In [ ]: from pyspark.sql.functions import initcap

        df_initcap = df.withColumn("article_initcap", initcap(df['article']))
        df_initcap.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time| article_initcap|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|Blueberry Muffin|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|       Bear Claw|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|        Baguette|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|       Lemon Bar|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|          Muffin|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------------+
only showing top 5 rows
```

Removing Leading and Trailing Spaces

```
In [ ]: from pyspark.sql.functions import trim

        df_trimmed = df.withColumn("pos_name_trimmed", trim(df['pos_name']))
        df_trimmed.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|   pos_name_trimmed|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|Jendouba_Ain_Drahem|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|Jendouba_Ain_Drahem|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|Jendouba_Ain_Drahem|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|Jendouba_Ain_Drahem|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|Jendouba_Ain_Drahem|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-------------------+
only showing top 5 rows
```

Extracting Substrings from a Column

```
In [ ]: from pyspark.sql.functions import substring

        df_substring = df.withColumn("article_substring", substring(df['sale_time'], 1, 4))
        df_substring.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|article_substring|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|             2024|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|             2024|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|             2024|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|             2024|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|             2023|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------------+
only showing top 5 rows
```

Extracting Date and Time Components

Adding, Subtracting Days

```
In [ ]: from pyspark.sql.functions import date_add, date_sub

        date_df = df.withColumn('DateWithDay', date_add('sale_time', 1))
        date_df.show(3)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|DateWithDay|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42| 2024-02-20|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42| 2024-02-20|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42| 2024-02-20|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------+
only showing top 3 rows
```

Extracting the Month, Year, Quarter

```
In [ ]: from pyspark.sql.functions import month, year, quarter

        month_df = df.withColumn('Month', month('sale_time'))
        month_df.show(3)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Month|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|    2|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|    2|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|    2|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----+
only showing top 3 rows
```

Extracting the Day of the Month

```
In [ ]: from pyspark.sql.functions import dayofmonth

        day_df = df.withColumn('Day_Number', dayofmonth('sale_time'))
        day_df.show(2)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Day_Number|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|        19|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|        19|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----------+
only showing top 2 rows
```

Extracting the Day of the Week Name

```
In [ ]: from pyspark.sql.functions import date_format

        day_name_df = df.withColumn('Day_Name', date_format('sale_time', 'EEEE'))
        day_name_df.show(3)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+--------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Day_Name|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+--------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|  Monday|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|  Monday|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|  Monday|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+--------+
only showing top 3 rows
```

Extracting the Hour, Minute, second

```
In [ ]: from pyspark.sql.functions import hour, minute, second

        hour_df = df.withColumn('Hour', hour('sale_time'))
        hour_df.show(3)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Hour|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|  13|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|  13|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|  13|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+----+
only showing top 3 rows
```

Data Filtering and Sorting

Data Filtering

```
In [ ]: from pyspark.sql.functions import col
```

Data Filtering Using the filter() Function

```
In [ ]: filter_1 = df.filter(col('quantity') > 5)
        filter_1.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Data Filtering Using the where() Function

In [ ]:

```python
filter_2 = df.where(col('payment_mode') == "online")
filter_2.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|  Apple Turnover|     3.0|  5.0| 15.0|livraison|      online|2023-03-08 18:29:41|
|     2365|     2|  Jendouba_Bousalem|         Strudel|     8.0|  8.0| 64.0|livraison|      online|2023-02-19 12:06:11|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the Logical Operator "&" (AND)

In [ ]:

```python
filter_3 = df.filter((col('sale_type') == 'direct') & (col('payment_mode') == 'card'))
filter_3.show(5)
```

```
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|       article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|      Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|        Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
|     2365|     2|  Jendouba_Bousalem|Apple Turnover|    20.0|  5.0|100.0|   direct|        card|2023-02-19 12:06:11|
|    91492|     4|   Jendouba_Tabarka|    Cheesecake|    12.0| 10.5|126.0|   direct|        card|2024-01-13 20:01:18|
|    91492|     4|   Jendouba_Tabarka|      Baguette|     4.0|  2.0|  8.0|   direct|        card|2024-01-13 20:01:18|
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the Logical Operator "|" (OR)

In [ ]:

```python
filter_4 = df.filter((col('sale_type') == 'direct') | (col('payment_mode') == 'card'))
filter_4.show(5)
```

```
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|        article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|       Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|         Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
|    39564|     2|  Jendouba_Bousalem|     Cherry Pie|    20.0|  9.5|190.0|   direct|        cash|2023-11-17 02:10:48|
|     2365|     2|  Jendouba_Bousalem| Apple Turnover|    20.0|  5.0|100.0|   direct|        card|2023-02-19 12:06:11|
|     2365|     2|  Jendouba_Bousalem|Red Velvet Cake|     6.0| 12.0| 72.0|   direct|        cash|2023-02-19 12:06:11|
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the isin() Method

In [ ]:

```python
Article_List = ['Croissant', 'Chocolate Eclair', 'Fruit Tart', 'Cinnamon Roll', 'Danish Pastry']

filter_5 = df.filter(col('article').isin(Article_List))
filter_5.show(5)
```

```
+---------+------+-----------------+-------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|         pos_name|      article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-----------------+-------------+--------+-----+-----+---------+------------+-------------------+
|    91492|     4| Jendouba_Tabarka|Danish Pastry|    11.0|  6.5| 71.5|   direct|        card|2024-01-13 20:01:18|
|    19413|     1|  Jendouba_Center|    Croissant|    13.0|  1.5| 19.5|livraison|      online|2024-02-13 19:35:23|
|    13685|     1|  Jendouba_Center|Danish Pastry|     7.0|  6.5| 45.5|livraison|      online|2023-03-20 17:35:06|
|    13685|     1|  Jendouba_Center|Cinnamon Roll|    14.0|  4.0| 56.0|livraison|      online|2023-03-20 17:35:06|
|    96107|     2|Jendouba_Bousalem|Cinnamon Roll|    14.0|  4.0| 56.0|   direct|        card|2024-02-29 23:34:19|
+---------+------+-----------------+-------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the Logical Operator "~" (NOT)

In [ ]:

```python
Article_List = ['Croissant', 'Chocolate Eclair', 'Fruit Tart', 'Cinnamon Roll', 'Danish Pastry']

filter_6 = df.filter(~col('article').isin(Article_List))
filter_6.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the like() Method

In [ ]:

```
filter_7 = df.filter(col('article').like("Apple %"))
filter_7.show(5)
```

```
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|       article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+
|    75376|     3|Jendouba_Ain_Drahem|Apple Turnover|     3.0|  5.0| 15.0|livraison|      online|2023-03-08 18:29:41|
|     2365|     2|  Jendouba_Bousalem|Apple Turnover|    20.0|  5.0|100.0|   direct|        card|2023-02-19 12:06:11|
|    96063|     4|   Jendouba_Tabarka|Apple Turnover|     7.0|  5.0| 35.0|livraison|      online|2023-10-20 07:35:48|
|    75660|     4|   Jendouba_Tabarka|Apple Turnover|    20.0|  5.0|100.0|livraison|      online|2023-05-18 01:27:22|
|    87457|     2|  Jendouba_Bousalem|Apple Turnover|     7.0|  5.0| 35.0|livraison|      online|2023-10-29 11:36:11|
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the endswith() Method

In [ ]:

```
filter_8 = df.filter(col('article').endswith("Pie"))
filter_8.show(5)
```

```
+---------+------+-----------------+----------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|         pos_name|   article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-----------------+----------+--------+-----+-----+---------+------------+-------------------+
|    39564|     2|Jendouba_Bousalem|Cherry Pie|    20.0|  9.5|190.0|   direct|        cash|2023-11-17 02:10:48|
|    19413|     1|  Jendouba_Center| Pecan Pie|     6.0| 11.0| 66.0|   direct|        cash|2024-02-13 19:35:23|
|    10105|     2|Jendouba_Bousalem|Cherry Pie|    17.0|  9.5|161.5|   direct|        card|2023-05-09 22:35:40|
|    96107|     2|Jendouba_Bousalem| Pecan Pie|     5.0| 11.0| 55.0|   direct|        card|2024-02-29 23:34:19|
|    87457|     2|Jendouba_Bousalem| Pecan Pie|    12.0| 11.0|132.0|   direct|        cash|2023-10-29 11:36:11|
+---------+------+-----------------+----------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the between() Method

In [ ]:

```
filter_9 = df.filter(col("quantity").between(10.0, 20.0))
filter_9.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the isNotNull() Method

In [ ]:

```
filter_10 = df.filter(col("article").isNotNull())
filter_10.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Filtering with the isNull() Method

In [ ]:

```
filter_11 = df.filter(col("article").isNull())
filter_11.show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     2|  Jendouba_Bousalem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Sorting Data (Ascending and Descending) with the "sort" or "orderBy" Functions

Sorting Data in Ascending Order

Method 1: Using the **sort()** Function

In [ ]:
```
sorted_df = df.sort('total')
sorted_df.show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     2|  Jendouba_Bousalem|   null|    null| null| null|   direct|        cash|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Method 2: Using the **orderBy()** Function

In [ ]:
```
sorted_df = df.orderBy('total')
sorted_df.show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     2|  Jendouba_Bousalem|   null|    null| null| null|   direct|        cash|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

In [ ]:
```
sorted_df = df.orderBy('total', ascending=True)
sorted_df.show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     2|  Jendouba_Bousalem|   null|    null| null| null|   direct|        cash|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

In [ ]:
```
from pyspark.sql.functions import asc

sorted_df = df.orderBy(asc('total'))
sorted_df.show(5)
```

```
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     2|  Jendouba_Bousalem|   null|    null| null| null|   direct|        cash|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
|     null|     3|Jendouba_Ain_Drahem|   null|    null| null| null|livraison|      online|2024-04-03 22:59:54|
+---------+------+-------------------+-------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Sorting Data in Descending Order

In [ ]:
```
sorted_df = df.orderBy('total', ascending=False)
sorted_df.show(5)
```

```
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|        article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
|    46483|     1|    Jendouba_Center|Red Velvet Cake|    20.0| 12.0|240.0|livraison|      online|2023-07-30 15:11:18|
|    26491|     4|   Jendouba_Tabarka|Red Velvet Cake|    20.0| 12.0|240.0|   direct|        card|2023-08-11 08:49:50|
|    62551|     3|Jendouba_Ain_Drahem|Red Velvet Cake|    20.0| 12.0|240.0|livraison|      online|2023-10-22 16:14:39|
|    55939|     3|Jendouba_Ain_Drahem|Red Velvet Cake|    20.0| 12.0|240.0|   direct|        card|2023-06-19 09:16:36|
|      492|     1|    Jendouba_Center|Red Velvet Cake|    20.0| 12.0|240.0|   direct|        card|2024-01-31 17:53:07|
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

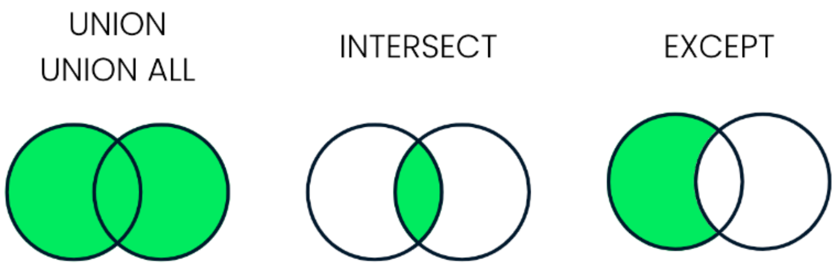
In [ ]:
```
from pyspark.sql.functions import desc

sorted_df = df.orderBy(desc('total'))
sorted_df.show(5)
```

```
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|        article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
|    46483|     1|    Jendouba_Center|Red Velvet Cake|    20.0| 12.0|240.0|livraison|      online|2023-07-30 15:11:18|
|    26491|     4|   Jendouba_Tabarka|Red Velvet Cake|    20.0| 12.0|240.0|   direct|        card|2023-08-11 08:49:50|
|    62551|     3|Jendouba_Ain_Drahem|Red Velvet Cake|    20.0| 12.0|240.0|livraison|      online|2023-10-22 16:14:39|
|    55939|     3|Jendouba_Ain_Drahem|Red Velvet Cake|    20.0| 12.0|240.0|   direct|        card|2023-06-19 09:16:36|
|      492|     1|    Jendouba_Center|Red Velvet Cake|    20.0| 12.0|240.0|   direct|        card|2024-01-31 17:53:07|
+---------+------+-------------------+---------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Merge two or more DataFrames

Difference between union intersect and except :

- **UNION** operator returns all the unique rows from both the left and the right query.
- **UNION ALL** included the duplicates as well.
- **INTERSECT** operator retrieves the common unique rows from both the left and the right query.
- **EXCEPT** operator returns unique rows from the left query that aren't in the right query's results.



Let us understand these differences with examples. We will use the following 2 tables for the examples :

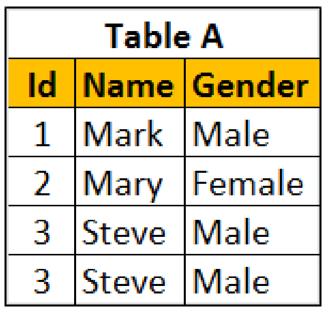


Table A		
Id	Name	Gender
1	Mark	Male
2	Mary	Female
3	Steve	Male
3	Steve	Male

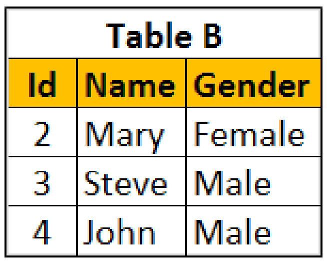
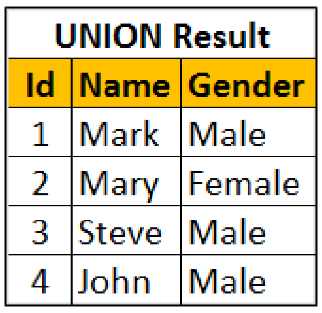


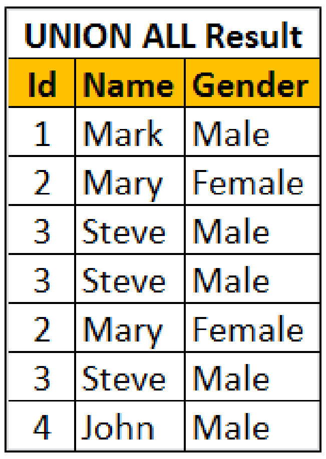
Table B		
Id	Name	Gender
2	Mary	Female
3	Steve	Male
4	John	Male

UNION :



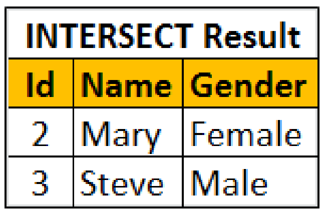
UNION Result		
Id	Name	Gender
1	Mark	Male
2	Mary	Female
3	Steve	Male
4	John	Male

UNION ALL :



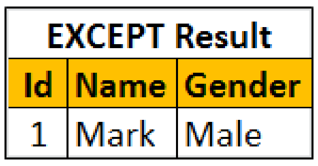
UNION ALL Result		
Id	Name	Gender
1	Mark	Male
2	Mary	Female
3	Steve	Male
3	Steve	Male
2	Mary	Female
3	Steve	Male
4	John	Male

INTERSECT :



INTERSECT Result		
Id	Name	Gender
2	Mary	Female
3	Steve	Male

EXCEPT :



EXCEPT Result		
Id	Name	Gender
1	Mark	Male

In [ ]:

```
mysql_host = "sql8.freesqldatabase.com"
mysql_port = "3306"
mysql_database = "sql8696474"
mysql_username = "sql8696474"
mysql_password = "2gVPjJi7xV"
mysql_table = "merge_sales"

jdbc_url = f"jdbc:mysql://{mysql_host}:{mysql_port}/{mysql_database}"

mysql_properties = {
    "user": mysql_username,
    "password": mysql_password,
    "driver": "com.mysql.cj.jdbc.Driver"
}

merge_df = spark.read.jdbc(url=jdbc_url, table=mysql_table, properties=mysql_properties)
```

In [ ]:

```
df.count()
```

Out[ ]:

```
9876
```

In [ ]:

```
merge_df.count()
```

Out[ ]:

```
528
```

In [ ]:

```
merge_df.distinct().count()
```

Out[ ]:

```
523
```

In [ ]:

```
merge_df.select("pos_id","pos_name").distinct().show()
```

```
+------+-------------------+
|pos_id|           pos_name|
+------+-------------------+
|     3|Jendouba_Ain_Drahem|
|     2|  Jendouba_Bousalem|
|     4|   Jendouba_Tabarka|
|     1|        Beja_Center|
|    10|              Tunis|
|     3|         Beja_Nefza|
|     8|                Kef|
|     4|       Beja_Testour|
|     1|    Jendouba_Center|
|     9|             Bizert|
|     2|        Beja_Amdoun|
+------+-------------------+
```

UNION of Two DataFrames

In [ ]:

```
merged_df_1 = df.union(merge_df)

merged_df_1.count()
```

Out[ ]:

```
10404
```

In [ ]:

```
9876 + 528
```

Out[ ]:

```
10404
```

UNION ALL of Two DataFrames link

In [ ]:

```
merged_df_2 = df.unionAll(merge_df)

merged_df_2.count()
```

Out[ ]:

```
10404
```

INTERSECT of Two DataFrames

In [ ]:

```
merged_df_3 = df.intersect(merge_df)

merged_df_3.count()
```

Out[ ]:

```
72
```

EXCEPT of Two DataFrames

In [ ]:

```
merged_df_4 = df.exceptAll(merge_df)

merged_df_4.count()
```

Out[ ]:

```
9804
```

In [ ]:

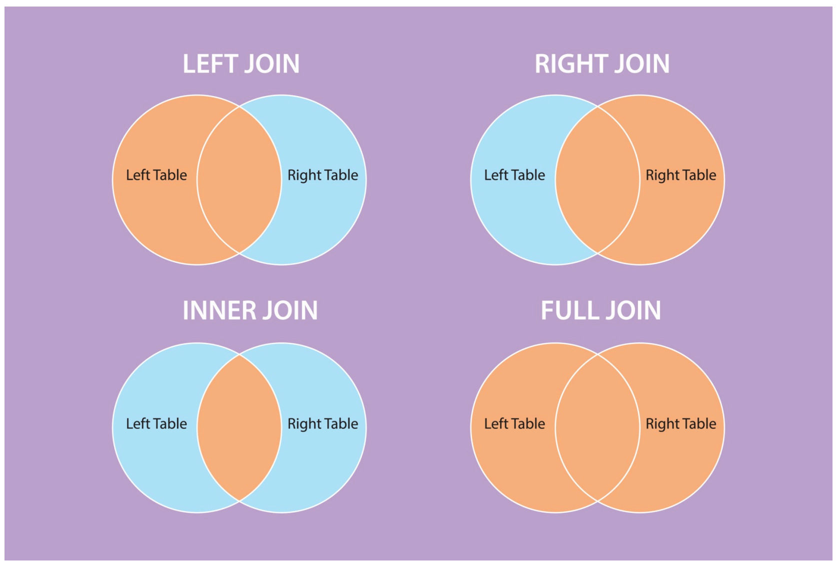
```
9876 - 72
```

Out[ ]:

```
9804
```

Joining Two DataFrames

Difference between INNER, LEFT, RIGHT and FULL JOIN :



- **INNER JOIN**: Returns only the rows with matching values in both the left and right tables based on the specified condition.
- **LEFT JOIN**: Returns all rows from the left table, and the matched rows from the right table. If no match is found, NULL values are returned for the right table columns.
- **RIGHT JOIN**: Returns all rows from the right table, and the matched rows from the left table. If no match is found, NULL values are returned for the left table columns.
- **FULL JOIN**: Returns all rows from both the left and right tables, combining the results where possible. If no match is found in either table, NULL values are returned for the columns of the table with no match.

Let us understand these differences with examples :

INNER JOIN :

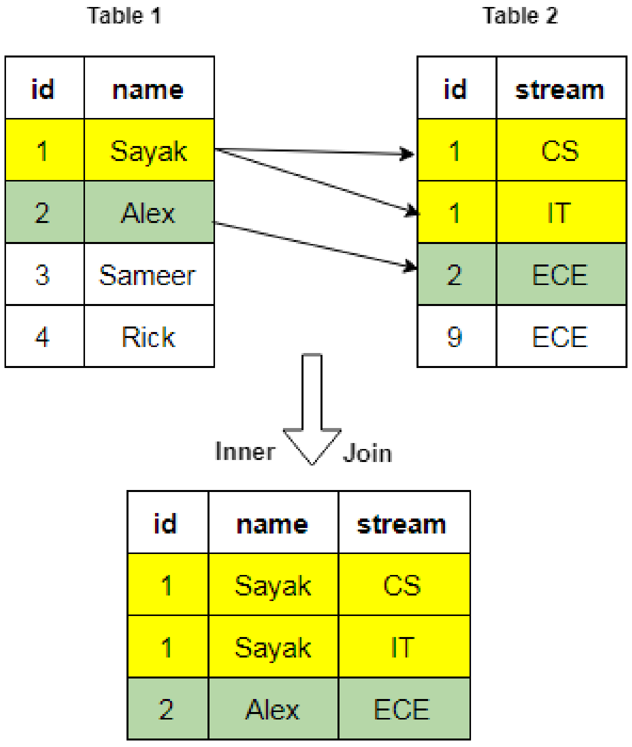


Table 1

id	name
1	Sayak
2	Alex
3	Sameer
4	Rick

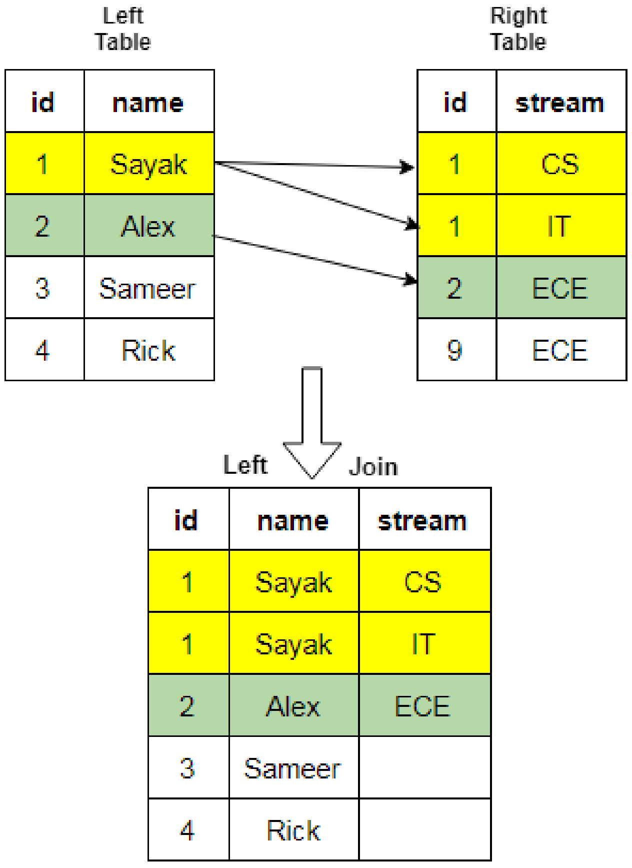
Table 2

id	stream
1	CS
1	IT
2	ECE
9	ECE

Inner Join

id	name	stream
1	Sayak	CS
1	Sayak	IT
2	Alex	ECE

LEFT JOIN :



Left Table

id	name
1	Sayak
2	Alex
3	Sameer
4	Rick

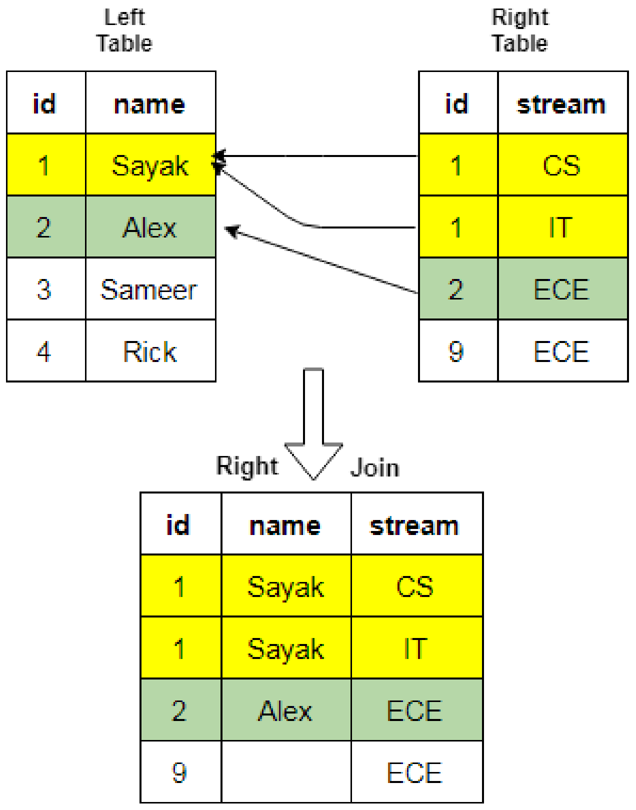
Right Table

id	stream
1	CS
1	IT
2	ECE
9	ECE

Left Join

id	name	stream
1	Sayak	CS
1	Sayak	IT
2	Alex	ECE
3	Sameer	
4	Rick	

RIGHT JOIN :

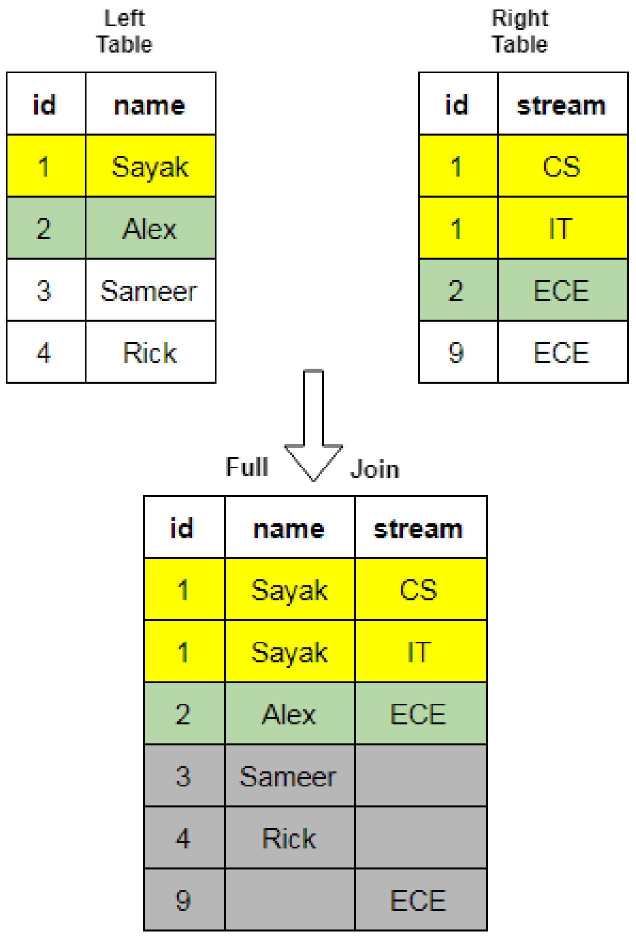


id	name
1	Sayak
2	Alex
3	Sameer
4	Rick

id	stream
1	CS
1	IT
2	ECE
9	ECE

id	name	stream
1	Sayak	CS
1	Sayak	IT
2	Alex	ECE
9		ECE

FULL JOIN :



id	name
1	Sayak
2	Alex
3	Sameer
4	Rick

id	stream
1	CS
1	IT
2	ECE
9	ECE

id	name	stream
1	Sayak	CS
1	Sayak	IT
2	Alex	ECE
3	Sameer	
4	Rick	
9		ECE

In [ ]:

```
mysql_host = "sql8.freesqldatabase.com"
mysql_port = "3306"
mysql_database = "sql8696474"
mysql_username = "sql8696474"
mysql_password = "2gVPjJi7xV"
mysql_table = "join_sales"

jdbc_url = f"jdbc:mysql://{mysql_host}:{mysql_port}/{mysql_database}"

mysql_properties = { "user": mysql_username, "password": mysql_password, "driver": "com.mysql.cj.jdbc.Driver"}

join_df = spark.read.jdbc(url=jdbc_url, table=mysql_table, properties=mysql_properties)

join_df = join_df.filter(col("article") != "Muffin")
```

In [ ]:

```
join_df.show(10, False)
```

```
+----------------+----------------------------------------------------------------------------------------------+
|article         |description                                                                                   |
+----------------+----------------------------------------------------------------------------------------------+
|Croissant       |A buttery, flaky pastry originating from France.                                              |
|Chocolate Eclair|A delicious pastry filled with chocolate cream and topped with chocolate icing.               |
|Fruit Tart      |A tart filled with assorted fresh fruits on top of a custard or cream filling.                |
|Cinnamon Roll   |A sweet roll served commonly in Northern Europe and North America.                            |
|Danish Pastry   |A multilayered, laminated sweet pastry in the viennoiserie tradition.                         |
|Palmier         |A pastry in the shape of a palm leaf or butterfly wings, made from puff pastry and sugar.     |
|Cream Puff      |A filled French pastry ball with a typically sweet and moist filling.                         |
|Apple Turnover  |A pastry made by placing apple filling on a piece of dough, then folding the dough over.      |
|Bear Claw       |A sweet, yeast-raised pastry, often shaped like a bear's paw and topped with almonds.         |
|Napoleon        |A pastry made of layers of puff pastry alternating with a sweet filling, usually pastry cream.|
+----------------+----------------------------------------------------------------------------------------------+
only showing top 10 rows
```

clean the DataFrame df for missing values

In [ ]:
```
new_df = df.dropna(how="any")
```

In [ ]:
```
new_df.count()
```

Out[ ]:
```
9506
```

INNER JOIN

Joining on a Common Column

Method 1:

In [ ]:
```
merged_inner = new_df.join(join_df, on='article', how='inner')

merged_inner.show(5)
```

```
+--------------+---------+------+-------------------+--------+-----+-----+---------+------------+-------------------+--------------------+
|       article|client_id|pos_id|           pos_name|quantity|price|total|sale_type|payment_mode|          sale_time|         description|
+--------------+---------+------+-------------------+--------+-----+-----+---------+------------+-------------------+--------------------+
|Apple Turnover|    75376|     3|Jendouba_Ain_Drahem|     3.0|  5.0| 15.0|livraison|      online|2023-03-08 18:29:41|A pastry made by ...|
|Apple Turnover|     2365|     2|  Jendouba_Bousalem|    20.0|  5.0|100.0|   direct|        card|2023-02-19 12:06:11|A pastry made by ...|
|Apple Turnover|    96063|     4|   Jendouba_Tabarka|     7.0|  5.0| 35.0|livraison|      online|2023-10-20 07:35:48|A pastry made by ...|
|Apple Turnover|    75660|     4|   Jendouba_Tabarka|    20.0|  5.0|100.0|livraison|      online|2023-05-18 01:27:22|A pastry made by ...|
|Apple Turnover|    87457|     2|  Jendouba_Bousalem|     7.0|  5.0| 35.0|livraison|      online|2023-10-29 11:36:11|A pastry made by ...|
+--------------+---------+------+-------------------+--------+-----+-----+---------+------------+-------------------+--------------------+
only showing top 5 rows
```

Method 2:

In [ ]:
```
merged_inner = new_df.join(join_df, df.article == join_df.article, how='inner')

merged_inner.count()
```

Out[ ]:
```
9088
```

In [ ]:
```
9506 - 9088
```

Out[ ]:
```
418
```

Joining on Two Common Columns

In [ ]:
```
join_condition = (df.article == join_df.article) & (df.pos_name == join_df.pos_name)

merged_inner = df.join(df_descriptions, join_condition, how='inner')
```

LEFT JOIN

In [ ]:
```
merged_left = new_df.join(join_df, new_df.article == join_df.article, how='left')

merged_left.count()
```

Out[ ]:
```
9506
```

RIGHT JOIN

In [ ]:
```
merged_right = new_df.join(join_df, new_df.article == join_df.article, how='right')

merged_right.count()
```

Out[ ]:
```
9088
```

In [ ]:
```
9506 - 9088
```

Out[ ]:
```
418
```

FULL JOIN

In [ ]:
```
merged_full = new_df.join(join_df, new_df.article == join_df.article, how='full')

merged_full.count()
```

Out[ ]:
```
9506
```

Aggregation, Grouping, and Window Functions

Aggregation Functions

- **AVG()**: Calculates the average of the set of values.
- **COUNT()**: Returns the count of rows.
- **SUM()**: Calculates the arithmetic sum of the set of numeric values.
- **MAX()**: From a group of values, returns the maximum value.
- **MIN()**: From a group of values, returns the minimum value.

In [ ]:

```
from pyspark.sql.functions import sum, avg,max, min, count

new_df.agg(sum('total').alias('Total_Sum'),
       avg('total').alias('Total_Mean'),
       max('total').alias('Total_Max'),
       min('total').alias('Total_Min'),
       count('total').alias('Total_Count')).show()
```

```
+-----------------+-----------------+---------+---------+-----------+
|        Total_Sum|       Total_Mean|Total_Max|Total_Min|Total_Count|
+-----------------+-----------------+---------+---------+-----------+
|642509.0000000019|67.58983799705469|    240.0|      1.5|       9506|
+-----------------+-----------------+---------+---------+-----------+
```

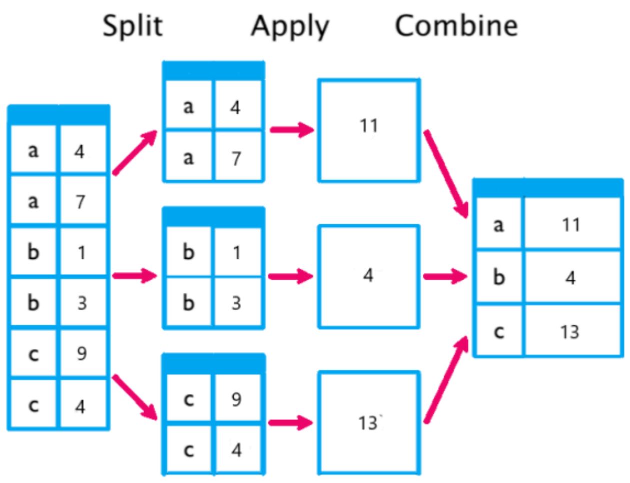
Grouping Functions

The Group By statement is used to group together any rows of a column with the same value stored in them, based on a function specified in the statement. Generally, these functions are one of the aggregate functions such as MAX() and SUM().

The Group By statement uses the split-apply-combine strategy:

- **Split**: The different groups are split with their values.
- **Apply**: The aggregate function is applied to the values of these groups.
- **Combine**: The values are combined in a single row.

The Group By function is typically used when you want to apply multiple aggregation operations on different columns at the same time, or when you want to rename the aggregated columns.



Example 1:

In [ ]:

```
new_df.groupBy('article').sum('total').show(5)
```

```
+----------+-----------------+
|   article|       sum(total)|
+----------+-----------------+
|Cream Puff|          40077.0|
|    Muffin|          15540.0|
| Pecan Pie|          46948.0|
|  Napoleon|33527.599999999984|
|     Scone|13378.399999999992|
+----------+-----------------+
only showing top 5 rows
```

Example 2:

In [ ]:

```
new_df.groupBy('article') \
  .agg(
        sum('total').alias('Total_Sum'),
        count('total').alias('Total_Count')) \
  .show(5)
```

```
+----------+-----------------+-----------+
|   article|        Total_Sum|Total_Count|
+----------+-----------------+-----------+
|Cream Puff|          40077.0|        422|
|    Muffin|          15540.0|        418|
| Pecan Pie|          46948.0|        405|
|  Napoleon|33527.599999999984|        407|
|     Scone|13378.399999999992|        456|
+----------+-----------------+-----------+
only showing top 5 rows
```

Example 3:

In [ ]:

```
from pyspark.sql.functions import sum, count

new_df.groupBy('article','pos_name') \
  .agg(
        sum('total').alias('Total_Sum'),
        count('total').alias('Total_Count')) \
  .show(5)
```

```
+----------------+-----------------+------------------+-----------+
|         article|         pos_name|         Total_Sum|Total_Count|
+----------------+-----------------+------------------+-----------+
|         Palmier|Jendouba_Bousalem|            3612.0|        108|
|   Key Lime Tart|  Jendouba_Center|10791.199999999999|        123|
|         Cupcake| Jendouba_Tabarka|            5301.0|        105|
|   Key Lime Tart|Jendouba_Bousalem| 7609.599999999996|        102|
|Chocolate Eclair|Jendouba_Bousalem|            8430.0|        109|
+----------------+-----------------+------------------+-----------+
only showing top 5 rows
```

Pivot Tables

The column whose distinct values become new columns.

In [ ]:

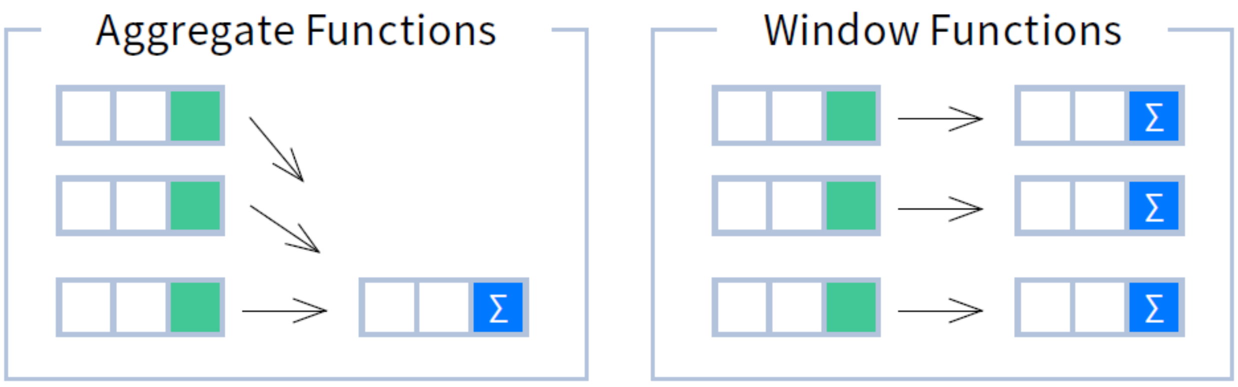
```
new_df.groupBy('article') \
  .pivot('sale_type') \
  .agg(sum('total').alias('Total_Sum')) \
  .show(5)
```

```
+----------+------------------+------------------+
|   article|            direct|         livraison|
+----------+------------------+------------------+
|Cream Puff|           20385.0|           19692.0|
|    Muffin|            8368.5|            7171.5|
|  Napoleon|18312.200000000004|15215.399999999992|
| Pecan Pie|           21538.0|           25410.0|
|     Scone| 6775.999999999996|            6602.4|
+----------+------------------+------------------+
only showing top 5 rows
```

Window Functions

Aggregate Functions vs Window Functions

Unlike aggregate functions, window functions do not collapse rows.



In [ ]:

```
from pyspark.sql.window import Window
from pyspark.sql.functions import sum, count
```

GroupBy Example

In [ ]:

```
new_df.groupBy('article') \
  .agg(
        sum('total').alias('Total_Sum'),
        count('total').alias('Total_Count')) \
  .orderBy("Total_Count") \
  .show(5)
```

```
+----------------+------------------+-----------+
|         article|         Total_Sum|Total_Count|
+----------------+------------------+-----------+
|Chocolate Eclair|           29970.0|        399|
|       Pecan Pie|           46948.0|        405|
|        Napoleon|33527.599999999984|        407|
|         Cupcake|           20052.0|        412|
|          Muffin|           15540.0|        418|
+----------------+------------------+-----------+
only showing top 5 rows
```

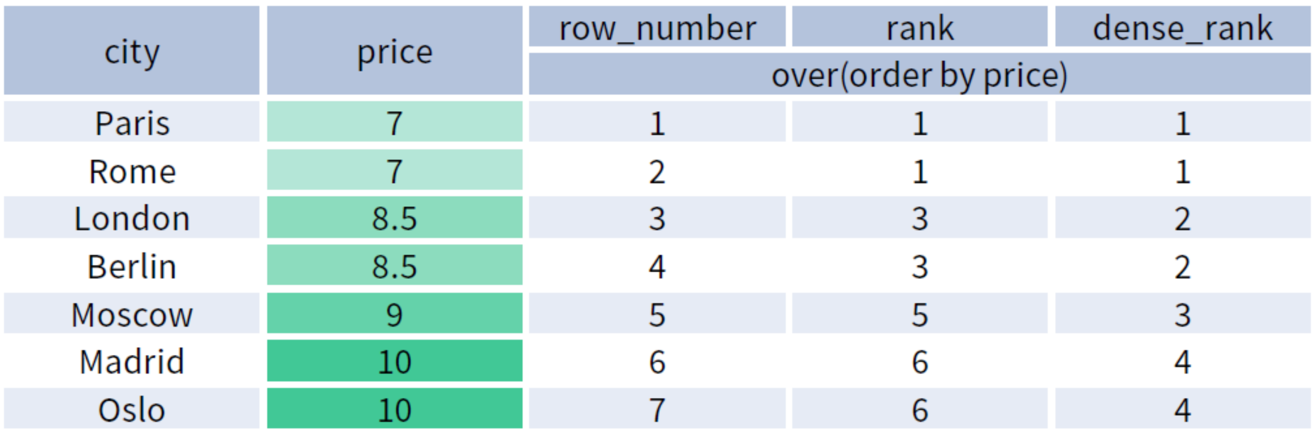
Window.partitionBy Example

```
In [ ]: window = Window.partitionBy('article')

        windowed_df = new_df.withColumn('Total_Sum', sum('total').over(window)) \
                        .withColumn('Total_Count', count('total').over(window)) \
                        .orderBy("Total_Count") \
                        .show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+---------+-----------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Total_Sum|Total_Count|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+---------+-----------+
|    91945|     3|Jendouba_Ain_Drahem|Chocolate Eclair|     6.0|  7.5| 45.0|livraison|      online|2023-07-25 07:05:59|  29970.0|        399|
|    51955|     3|Jendouba_Ain_Drahem|Chocolate Eclair|    15.0|  7.5|112.5|   direct|        cash|2023-08-12 22:55:16|  29970.0|        399|
|    57811|     1|    Jendouba_Center|Chocolate Eclair|     9.0|  7.5| 67.5|   direct|        cash|2023-07-05 18:16:34|  29970.0|        399|
|    43892|     4|   Jendouba_Tabarka|Chocolate Eclair|     1.0|  7.5|  7.5|livraison|      online|2024-01-05 06:53:42|  29970.0|        399|
|    94275|     2|  Jendouba_Bousalem|Chocolate Eclair|    16.0|  7.5|120.0|livraison|      online|2023-11-18 22:06:52|  29970.0|        399|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+---------+-----------+
only showing top 5 rows
```

Window + Ranking Functions

- **row_number()** - unique number for each row within partition, with different numbers for tied values
- **rank()** - ranking within partition, with gaps and same ranking for tied values
- **dense_rank()** - ranking within partition, with no gaps and same ranking for tied values



city	price	row_number	rank	dense_rank
		over(order by price)		
Paris	7	1	1	1
Rome	7	2	1	1
London	8.5	3	3	2
Berlin	8.5	4	3	2
Moscow	9	5	5	3
Madrid	10	6	6	4
Oslo	10	7	6	4

Example 1

```
In [ ]: from pyspark.sql.window import Window
        from pyspark.sql.functions import row_number, desc
```

```
In [ ]: window = Window.partitionBy('article').orderBy('total')
        ranked_df = new_df.withColumn('rank', row_number().over(window))
        ranked_df.show(5)
```

```
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+----+
|client_id|pos_id|           pos_name|       article|quantity|price|total|sale_type|payment_mode|          sale_time|rank|
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+----+
|    42619|     2|  Jendouba_Bousalem|Apple Turnover|     1.0|  5.0|  5.0|   direct|        cash|2023-03-03 23:44:35|   1|
|     3649|     3|Jendouba_Ain_Drahem|Apple Turnover|     1.0|  5.0|  5.0|livraison|      online|2023-04-30 08:45:21|   2|
|    11902|     4|   Jendouba_Tabarka|Apple Turnover|     1.0|  5.0|  5.0|   direct|        cash|2023-04-30 10:18:21|   3|
|    44116|     1|    Jendouba_Center|Apple Turnover|     1.0|  5.0|  5.0|livraison|      online|2023-12-31 08:06:08|   4|
|    64700|     3|Jendouba_Ain_Drahem|Apple Turnover|     1.0|  5.0|  5.0|   direct|        card|2023-08-25 01:11:10|   5|
+---------+------+-------------------+--------------+--------+-----+-----+---------+------------+-------------------+----+
only showing top 5 rows
```

Example 2

```
In [ ]: window = Window.partitionBy('Article').orderBy(desc('Total'))
        ranked_df = new_df.withColumn('rank', row_number().over(window))
        ranked_df.show(5)
```

```
+---------+------+-----------------+--------------+--------+-----+-----+---------+------------+-------------------+----+
|client_id|pos_id|         pos_name|       article|quantity|price|total|sale_type|payment_mode|          sale_time|rank|
+---------+------+-----------------+--------------+--------+-----+-----+---------+------------+-------------------+----+
|     2365|     2|Jendouba_Bousalem|Apple Turnover|    20.0|  5.0|100.0|   direct|        card|2023-02-19 12:06:11|   1|
|    75660|     4| Jendouba_Tabarka|Apple Turnover|    20.0|  5.0|100.0|livraison|      online|2023-05-18 01:27:22|   2|
|    34432|     1|  Jendouba_Center|Apple Turnover|    20.0|  5.0|100.0|livraison|      online|2023-09-05 01:10:37|   3|
|    76567|     2|Jendouba_Bousalem|Apple Turnover|    20.0|  5.0|100.0|   direct|        card|2023-05-16 23:32:05|   4|
|    26798|     2|Jendouba_Bousalem|Apple Turnover|    20.0|  5.0|100.0|   direct|        cash|2023-06-28 15:23:30|   5|
+---------+------+-----------------+--------------+--------+-----+-----+---------+------------+-------------------+----+
only showing top 5 rows
```

Window + Distribution Functions

- **cume_dist()** the cumulative distribution of a value within a group of values, i.e., the number of rows with values less than or equal to the current row's value divided by the total number of rows; a value in (0, 1] interval

- **percent_rank()** the percentile ranking number of a row—a value in [0, 1] interval: (rank-1) / (total number of rows - 1)

`cume_dist() OVER(ORDER BY sold)`

city	sold	cume_dist
Paris	100	0.2
Berlin	150	0.4
Rome	200	0.8
Moscow	200	0.8
London	300	1

80% of values are less than or equal to this one

`percent_rank() OVER(ORDER BY sold)`

city	sold	percent_rank
Paris	100	0
Berlin	150	0.25
Rome	200	0.5
Moscow	200	0.5
London	300	1

without this row 50% of values are less than this row's value

In [ ]:

```python
from pyspark.sql.window import Window
from pyspark.sql.functions import col, cume_dist

windowSpec = Window.orderBy(col("total"))

cume_df = new_df.withColumn("cumulative_distribution", cume_dist().over(windowSpec))
cume_df.show(5)
```

```
+---------+------+-------------------+---------+--------+-----+-----+---------+------------+-------------------+----------------------+
|client_id|pos_id|           pos_name|  article|quantity|price|total|sale_type|payment_mode|          sale_time|cumulative_distribution|
+---------+------+-------------------+---------+--------+-----+-----+---------+------------+-------------------+----------------------+
|    64726|     3|Jendouba_Ain_Drahem|Croissant|     1.0|  1.5|  1.5|livraison|      online|2023-01-26 01:39:01|  0.002103934357248054|
|    82657|     4|   Jendouba_Tabarka|Croissant|     1.0|  1.5|  1.5|   direct|        card|2023-02-09 21:36:48|  0.002103934357248054|
|    28496|     4|   Jendouba_Tabarka|Croissant|     1.0|  1.5|  1.5|livraison|      online|2023-08-30 16:56:55|  0.002103934357248054|
|    13206|     2|  Jendouba_Bousalem|Croissant|     1.0|  1.5|  1.5|livraison|      online|2024-01-16 03:17:05|  0.002103934357248054|
|    32696|     3|Jendouba_Ain_Drahem|Croissant|     1.0|  1.5|  1.5|livraison|      online|2024-03-12 19:29:30|  0.002103934357248054|
+---------+------+-------------------+---------+--------+-----+-----+---------+------------+-------------------+----------------------+
only showing top 5 rows
```

Window + Analytic Functions

- **lead(expr, offset, default)** the percentile ranking number of a row—a value in [0, 1] interval: (rank-1) / (total number of rows - 1)

- **lag(expr, offset, default)** the cumulative distribution of a value within a group of values, i.e., the number of rows with values less than or equal to the current row's value divided by the total number of rows; a value in (0, 1] interval

`lag(sold) OVER(ORDER BY month)`

order by month

month	sold	
1	500	NULL
2	300	500
3	400	300
4	100	400
5	500	100

`lead(sold) OVER(ORDER BY month)`

order by month

month	sold	
1	500	300
2	300	400
3	400	100
4	100	500
5	500	NULL

`lag(sold, 2, 0) OVER(ORDER BY month)`

order by month

month	sold	
1	500	0
2	300	0
3	400	500
4	100	300
5	500	400

offset=2

`lead(sold, 2, 0) OVER(ORDER BY month)`

order by month

month	sold	
1	500	400
2	300	100
3	400	500
4	100	0
5	500	0

offset=2

- **first_value(expr)** the value for the first row within the window frame

- **last_value(expr)** the value for the last row within the window frame

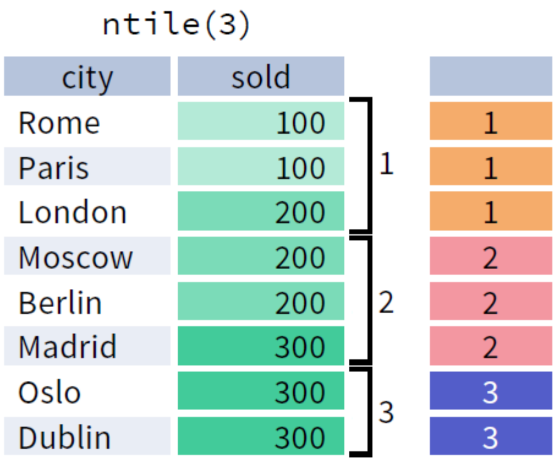
```
first_value(sold) OVER
(PARTITION BY city ORDER BY month)
```

city	month	sold	first_value
Paris	1	500	500
Paris	2	300	500
Paris	3	400	500
Rome	2	200	200
Rome	3	300	200
Rome	4	500	200

```
last_value(sold) OVER
(PARTITION BY city ORDER BY month
RANGE BETWEEN UNBOUNDED PRECEDING
AND UNBOUNDED FOLLOWING)
```

city	month	sold	last_value
Paris	1	500	400
Paris	2	300	400
Paris	3	400	400
Rome	2	200	500
Rome	3	300	500
Rome	4	500	500

- **ntile(n)** divide rows within a partition as equally as possible into n groups, and assign each row its group number.



ntile(3)

city	sold		
Rome	100	1	1
Paris	100		1
London	200		1
Moscow	200	2	2
Berlin	200		2
Madrid	300		2
Oslo	300	3	3
Dublin	300		3

In [ ]:
```
from pyspark.sql.window import Window
from pyspark.sql.functions import col, lag

windowSpec = Window.orderBy(col("sale_time"))

lag_df = new_df.withColumn("Previous Total", lag("Total", 1).over(windowSpec))
lag_df.show(5)
```

```
+---------+------+---------------+-------------+--------+-----+-----+---------+------------+-------------------+--------------+
|client_id|pos_id|       pos_name|      article|quantity|price|total|sale_type|payment_mode|          sale_time|Previous Total|
+---------+------+---------------+-------------+--------+-----+-----+---------+------------+-------------------+--------------+
|    46899|     1|Jendouba_Center|      Palmier|     9.0|  3.0| 27.0|   direct|        card|2023-01-01 00:48:33|          null|
|    46899|     1|Jendouba_Center|   Cream Puff|     2.0|  9.0| 18.0|livraison|      online|2023-01-01 00:48:33|          27.0|
|    46899|     1|Jendouba_Center|   Cream Puff|     3.0|  9.0| 27.0|   direct|        cash|2023-01-01 00:48:33|          18.0|
|    70879|     1|Jendouba_Center|Danish Pastry|    12.0|  6.5| 78.0|   direct|        card|2023-01-01 06:22:22|          27.0|
|    70879|     1|Jendouba_Center|     Baguette|     6.0|  2.0| 12.0|   direct|        cash|2023-01-01 06:22:22|          78.0|
+---------+------+---------------+-------------+--------+-----+-----+---------+------------+-------------------+--------------+
only showing top 5 rows
```

User-Defined Functions (UDF)

In [ ]:
```
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType, DoubleType
```

Example 1

In [ ]:
```
def kg_to_pounds(quantity):
    return quantity * 2.20462

kg_to_pounds_udf = udf(kg_to_pounds, DoubleType())

df_with_pounds = new_df.withColumn('Quantity_in_Pounds', kg_to_pounds_udf('Quantity'))
df_with_pounds.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Quantity_in_Pounds|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|26.455439999999996|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|           44.0924|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|30.864679999999996|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|           33.0693|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|30.864679999999996|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+------------------+
only showing top 5 rows
```

Example 2

In [ ]:
```
def categorize_quantity(quantity):
    if quantity < 5:
        return 'Low'
    elif quantity >= 5 and quantity < 10:
        return 'Medium'
    else:
        return 'High'

categorize_udf = udf(categorize_quantity, StringType())

categorized_df = new_df.withColumn('Quantity_Category', categorize_udf(df['Quantity']))
categorized_df.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|Quantity_Category|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|             High|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|             High|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|             High|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|             High|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|             High|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+-----------------+
only showing top 5 rows
```

Spark SQL

In [ ]:

```
new_df.createOrReplaceTempView("sales")
```

Example 1: **Show** Data

In [ ]:

```
query1 = """
SELECT *
FROM sales
"""
result1 = spark.sql(query1)
result1.show(5)
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Lemon Bar|    15.0|  6.0| 90.0|livraison|      online|2024-02-19 13:31:42|
|    75376|     3|Jendouba_Ain_Drahem|          Muffin|    14.0|  3.5| 49.0|   direct|        card|2023-03-08 18:29:41|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 5 rows
```

Example 2: **SUM** Function

In [ ]:

```
query2 = """
SELECT SUM(total) AS total_sales
FROM sales
"""
result2 = spark.sql(query2)
result2.show()
```

```
+-----------------+
|      total_sales|
+-----------------+
|642509.0000000019|
+-----------------+
```

Example 3: **Where** Condition

In [ ]:

```
query3 = """
SELECT article, quantity
FROM sales
WHERE Quantity > 5
"""
result3 = spark.sql(query3)
result3.show(5)
```

```
+----------------+--------+
|         article|quantity|
+----------------+--------+
|Blueberry Muffin|    12.0|
|       Bear Claw|    20.0|
|        Baguette|    14.0|
|       Lemon Bar|    15.0|
|          Muffin|    14.0|
+----------------+--------+
only showing top 5 rows
```

Example 4: **Group By**

In [ ]:

```
query4 = """
SELECT article, AVG(total) AS average_total
FROM sales
GROUP BY article
"""
result4 = spark.sql(query4)
result4.show(5)
```

```
+----------+------------------+
|   article|     average_total|
+----------+------------------+
|Cream Puff|  94.9691943127962|
|    Muffin|37.177033492822964|
| Pecan Pie|115.92098765432098|
|  Napoleon| 82.37739557739553|
|     Scone|29.338596491228053|
+----------+------------------+
only showing top 5 rows
```

Python Data Visualization

In [ ]:
```python
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:
```python
new_df.show(3)
new_df.count()
```

```
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|client_id|pos_id|           pos_name|         article|quantity|price|total|sale_type|payment_mode|          sale_time|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
|    15526|     3|Jendouba_Ain_Drahem|Blueberry Muffin|    12.0|  3.8| 45.6|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|       Bear Claw|    20.0|  6.8|136.0|livraison|      online|2024-02-19 13:31:42|
|    15526|     3|Jendouba_Ain_Drahem|        Baguette|    14.0|  2.0| 28.0|   direct|        card|2024-02-19 13:31:42|
+---------+------+-------------------+----------------+--------+-----+-----+---------+------------+-------------------+
only showing top 3 rows
```

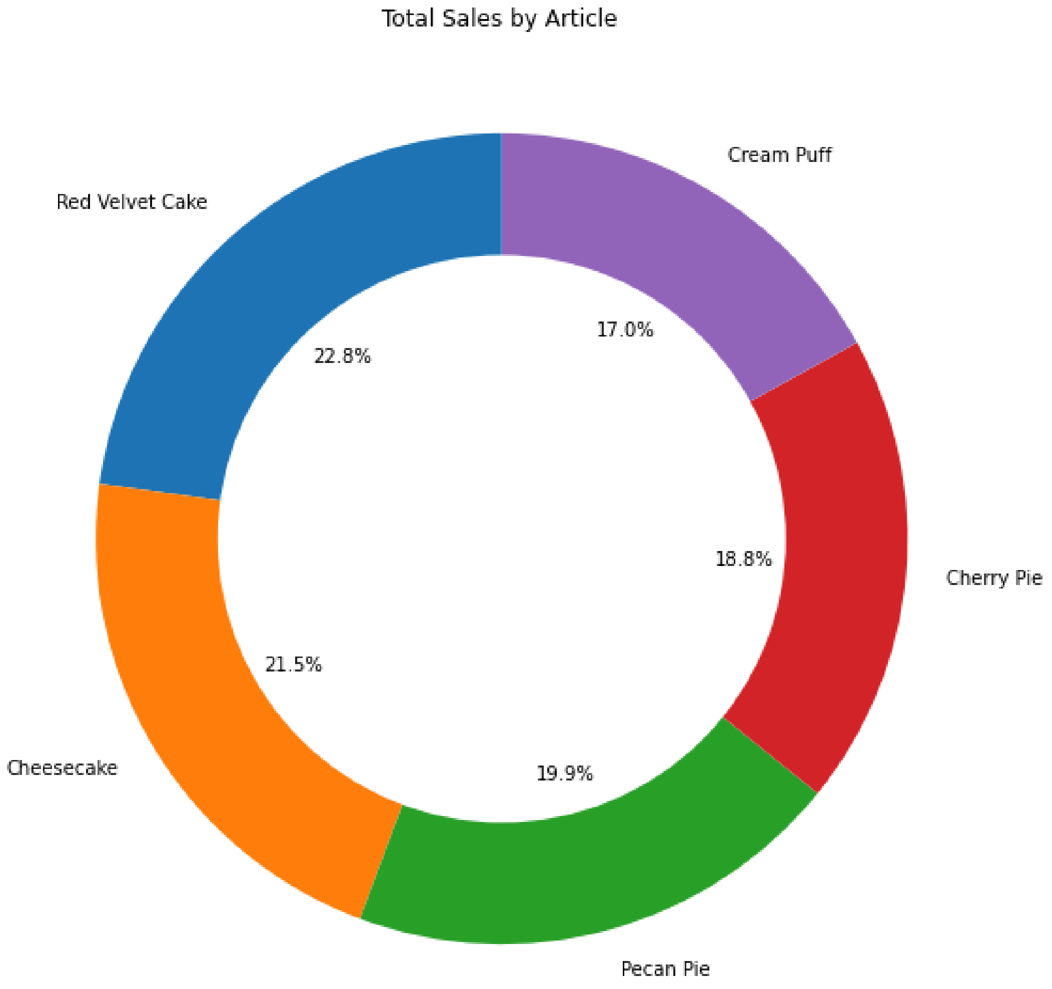
Out[ ]:
```
9506
```

Example 1: Donut (Pie) Chart

In [ ]:
```python
from pyspark.sql.functions import sum

article_totals = new_df.groupBy('article').agg(sum('total').alias('Total')).orderBy('Total', ascending=False)

article_totals_pd = article_totals.toPandas()
```

In [ ]:
```python
article_totals_pd = article_totals_pd[:5]
```

In [ ]:
```python
plt.figure(figsize=(8, 8))
plt.pie(article_totals_pd['Total'], labels=article_totals_pd['article'], autopct='%1.1f%%', startangle=90)

centre_circle = plt.Circle((0,0),0.70,fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.title('Total Sales by Article')
plt.tight_layout()
plt.show()
```



Example 2: Bar Chart

In [ ]:

```
from pyspark.sql.functions import sum

article_quantity = new_df.groupBy('article').agg(sum('quantity').alias('Total')).orderBy('Total', ascending=False)
article_quantity_pd = article_totals.toPandas()
```

In [ ]:

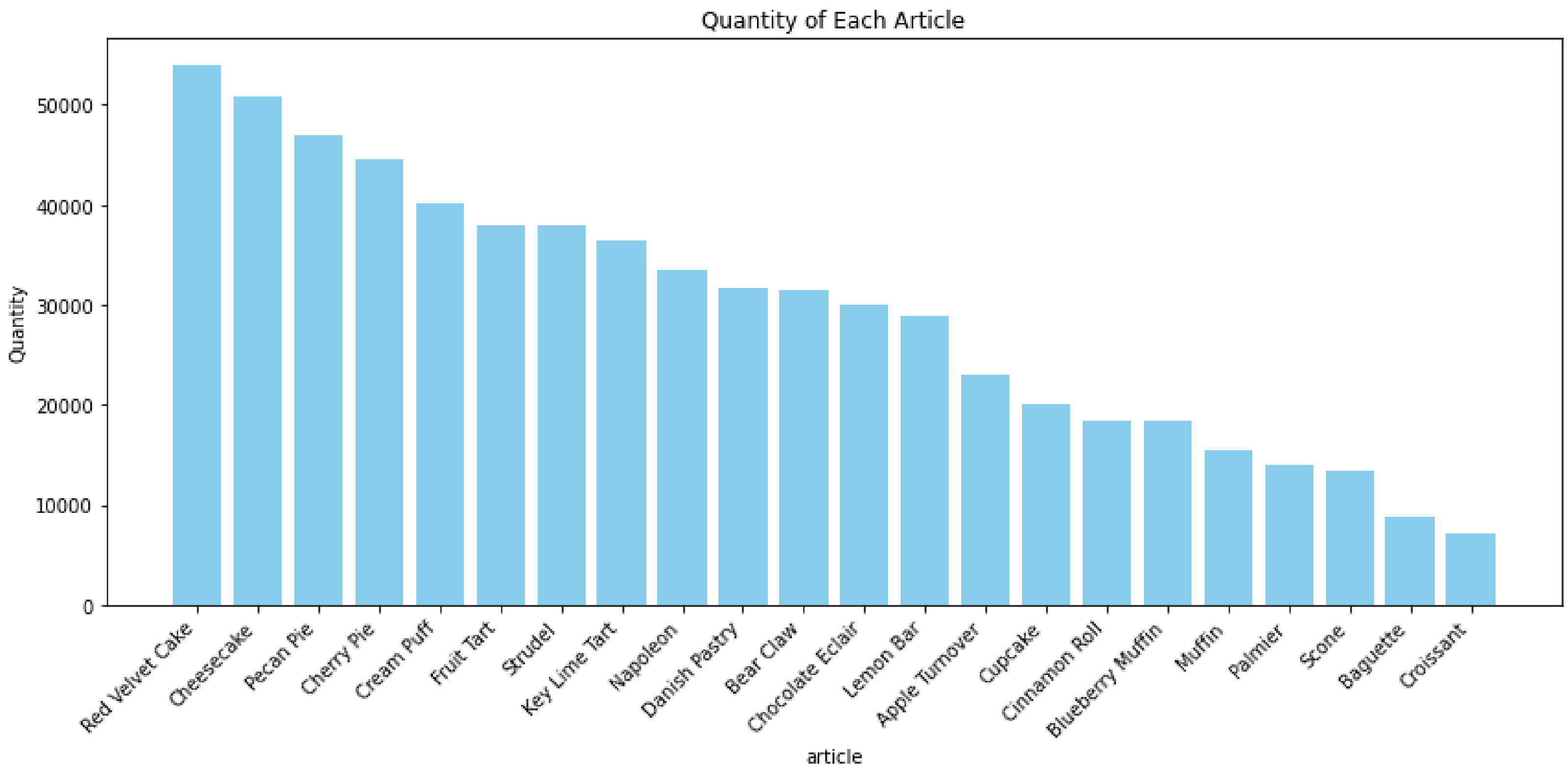
```
article_quantity_pd.head(5)
```

Out[ ]:

	article	Total
0	Red Velvet Cake	53940.0
1	Cheesecake	50778.0
2	Pecan Pie	46948.0
3	Cherry Pie	44517.0
4	Cream Puff	40077.0

In [ ]:

```
plt.figure(figsize=(12, 6))
plt.bar(article_quantity_pd['article'], article_quantity_pd['Total'], color='skyblue')
plt.title('Quantity of Each Article')
plt.xlabel('article')
plt.ylabel('Quantity')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



Example 3: Line Chart

In [ ]:

```
df_Croissant = new_df.where(df['article'] == "Croissant") \
                 .select("article", "quantity", "sale_time") \
                 .orderBy("sale_time", ascending=True) \
                 .dropDuplicates(["sale_time"])

df_Croissant_pd = df_Croissant.toPandas()
df_Croissant_pd.set_index('sale_time', inplace=True)
```

In [ ]:

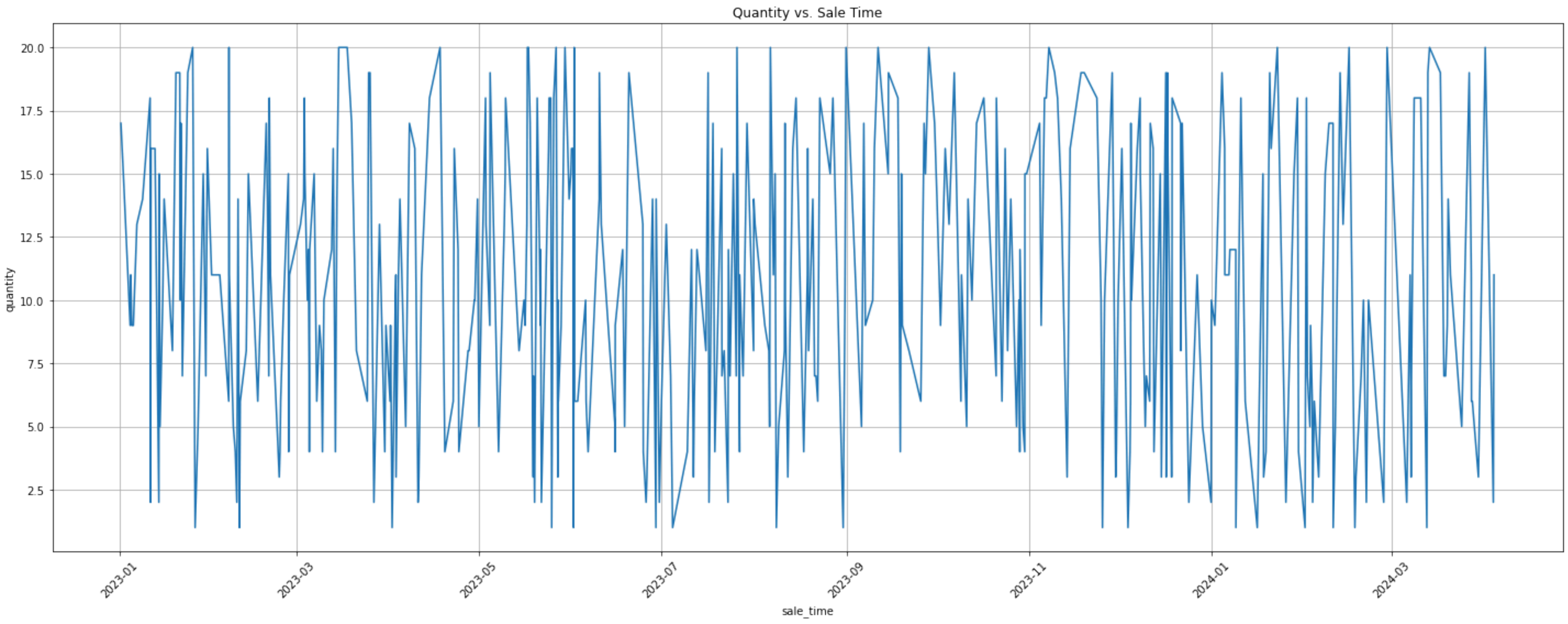
```
df_Croissant_pd.head(5)
```

Out[ ]:

	article	quantity
sale_time		
2023-01-01 06:22:22	Croissant	17.0
2023-01-02 01:59:10	Croissant	15.0
2023-01-04 08:39:15	Croissant	9.0
2023-01-04 13:20:38	Croissant	11.0
2023-01-04 20:48:10	Croissant	9.0

In [ ]:

```
plt.figure(figsize=(20, 8))
plt.plot(df_Croissant_pd.index, df_Croissant_pd['quantity'])
plt.title('Quantity vs. Sale Time')
plt.xlabel('sale_time')
plt.ylabel('quantity')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()
plt.show()
```


Quantity vs. Sale Time
quantity
sale_time
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
2023-01
2023-03
2023-05
2023-07
2023-09
2023-11
2024-01
2024-03