Bytewise Fellowship

Task 11: Data Cleaning and Transformation for Demand Forecasting

Name	Muhammad Ahsan Saleem
Date	13th August, 2024
Submitted to:	Muhammad Bilal

Task 11: Data Cleaning and Transformation for Demand Forecasting

Task Statement: As a Data Engineer at Voltmart, an electronics e-commerce company, your task is to clean and preprocess the order data from last year. The cleaned data will be used by the Machine Learning team to build a demand forecasting model. The analyst provided a CSV file called "data.csv," and you must ensure that the data adheres to the required format, adding new columns where necessary, and finally save the result as a Parquet file.

Requirements:

- Read the provided CSV file into a PySpark DataFrame.
- Add a time_of_day column based on the hour of the order_date.
- Filter out rows where the hour in order_date is between 0 and 5 inclusive.
- Convert the order_date column from a timestamp to a date.
- Remove rows where the product column has the value "TV".
- Ensure all values in the category column are lowercase.
- Extract the state from the purchase_address and store it in a new state column.
- Save the final DataFrame as a Parquet file.

Steps and Code:

In this section, I will outline the steps taken to clean and transform the orders data. Each step is accompanied by the corresponding PySpark code.

Step 1: Reading the Data

- We start by reading the CSV file into a PySpark DataFrame using the spark.read.csv() function.

Code:

```
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder \
    .appName("BWF-PySparkTask") \
    .getOrCreate()

# Path to your CSV file
file_path = 'data.csv'
```

```
# Read CSV into DataFrame
orders_data = spark.read.csv(file_path, header=True, inferSchema=True)

# Display the DataFrame
orders_data.show()
```

Output:

```
+-------------------+--------+--------------------+------------+-----------+--------------------+----------------+-----------+----------+--------+--------+
|         order_date|order_id|             product|  product_id|   category|    purchase_address|quantity_ordered|  price_each|cost_price|turnover|  margin|
+-------------------+--------+--------------------+------------+-----------+--------------------+----------------+-----------+----------+--------+--------+
|2023-01-22 21:25:00|  141234|              iPhone|5638008983335|  Vêtements| """944 Walnut St|          Boston| MA 02215"""|         1|   700.0|   231.0|
|2023-01-28 14:15:00|  141235|Lightning Chargin...|5563319511488|Alimentation|  """185 Maple St|        Portland| OR 97035"""|         1|   14.95|   7.475|
|2023-01-17 13:33:00|  141236|    Wired Headphones|2113973395220|  Vêtements|  """538 Adams St|   San Francisco| CA 94016"""|         2|   11.99|   5.995|
|2023-01-05 20:33:00|  141237|    27in FHD Monitor|3069156759167|     Sports|   """738 10th St|     Los Angeles| CA 90001"""|         1|  149.99| 97.4935|
|2023-01-25 11:59:00|  141238|    Wired Headphones|9692680938163|Électronique|   """387 10th St|          Austin| TX 73301"""|         1|   11.99|   5.995|
|2023-01-29 20:22:00|  141239|AAA Batteries (4-...|2953868554188|Alimentation| """775 Willow St|   San Francisco| CA 94016"""|         1|    2.99|   1.495|
|2023-01-26 12:16:00|  141240|27in 4K Gaming Mo...|5173670800988|  Vêtements|   """979 Park St|     Los Angeles| CA 90001"""|         1|  389.99|128.6967|
|2023-01-05 12:04:00|  141241|USB-C Charging Cable|8051736777568|  Vêtements|   """181 6th St|   San Francisco| CA 94016"""|         1|   11.95|   5.975|
|2023-01-01 10:30:00|  141242|Bose SoundSport H...|1508418177978|Électronique| """867 Willow St|     Los Angeles| CA 90001"""|         1|   99.99|  49.995|
|2023-01-22 21:20:00|  141243|Apple Airpods Hea...|1386344211590|Électronique|"""657 Johnson St|   San Francisco| CA 94016"""|         1|   150.0|    97.5|
|2023-01-07 11:29:00|  141244|Apple Airpods Hea...|4332898830865|  Vêtements| """492 Walnut St|   San Francisco| CA 94016"""|         1|   150.0|    97.5|
|2023-01-31 10:12:00|  141245|  Macbook Pro Laptop|1169379570345|  Vêtements|    """322 6th St|   San Francisco| CA 94016"""|         1|  1700.0|   561.0|
|2023-01-09 18:57:00|  141246|AAA Batteries (4-...|4436184749366|  Vêtements|    """618 7th St|     Los Angeles| CA 90001"""|         3|    2.99|   1.495|
|2023-01-25 19:19:00|  141247|    27in FHD Monitor|7313825995563|  Vêtements| """512 Wilson St|   San Francisco| CA 94016"""|         1|  149.99| 97.4935|
|2023-01-03 21:54:00|  141248|       Flatscreen TV|4062756463060|Électronique| """363 Spruce St|          Austin| TX 73301"""|         1|   300.0|    99.0|
|2023-01-05 17:20:00|  141249|    27in FHD Monitor|9643428300795|Alimentation|  """440 Cedar St|        Portland| OR 97035"""|         1|  149.99| 97.4935|
|2023-01-10 11:20:00|  141250|     Vareebadd Phone|6721780072847|Alimentation| """471 Center St|     Los Angeles| CA 90001"""|         1|   400.0|   132.0|
|2023-01-24 08:13:00|  141251|Apple Airpods Hea...|2700099961823|Alimentation| """414 Walnut St|          Boston| MA 02215"""|         1|   150.0|    97.5|
|2023-01-30 09:28:00|  141252|USB-C Charging Cable|3692435232121|     Sports|    """220 9th St|     Los Angeles| CA 90001"""|         1|   11.95|   5.975|
|2023-01-17 00:09:00|  141253|AA Batteries (4-p...|6741495725758|Alimentation|   """385 11th St|         Atlanta| GA 30301"""|         1|    3.84|    1.92|
+-------------------+--------+--------------------+------------+-----------+--------------------+----------------+-----------+----------+--------+--------+
only showing top 20 rows
```

Step 2: Adding time_of_day Column

- A new column, time_of_day, is added based on the hour of the order_date. The new column categorizes the time into "morning", "afternoon", "evening", and "night".

Code:

```
from pyspark.sql.functions import when, hour, col

# Add time_of_day column based on the hour
orders_data = orders_data.withColumn(
    "time_of_day",
    when((hour(col("order_date")) >= 5) & (hour(col("order_date")) < 12), "morning")
    .when((hour(col("order_date")) >= 12) & (hour(col("order_date")) < 18),
"afternoon")
    .when((hour(col("order_date")) >= 18) & (hour(col("order_date")) < 24),
"evening")
    .otherwise("night")
)
```

Step 3: Filtering Out Rows Based on Time

- Rows where the hour in order_date is between 0 and 5 inclusive are filtered out to focus on meaningful order times.

Code:

```
# Filter out rows where the hour is between 0 and 5 inclusive
orders_data = orders_data.filter((hour(col("order_date")) < 0) |
(hour(col("order_date")) > 5))
```

Step 4: Converting order_date to Date

- The order_date column is converted from a timestamp to a date format to simplify the data.

Code:

```
from pyspark.sql.functions import to_date

# Convert order_date column from timestamp to date
orders_data = orders_data.withColumn("order_date", to_date(col("order_date")))
```

Step 5: Removing Specific Product Rows

- Any rows where the product column has the value "TV" are removed from the DataFrame.

Code:

```
# Remove rows where the product column has the value "TV"
orders_data = orders_data.filter(col("product") != "TV")
```

Step 6: Standardizing category Column

- The category column values are converted to lowercase to maintain consistency.

Code:

```
from pyspark.sql.functions import lower

# Ensure all values in the purchase_state column are lowercase
orders_data = orders_data.withColumn("category", lower(col("category")))
```

Step 7: Extracting the state from purchase_address

- The state information is extracted from the purchase_address column and stored in a new state column.

Code:

```
from pyspark.sql.functions import substring

# State is the 7th and 8th last characters
orders_data = orders_data.withColumn("state", substring("purchase_address", -8, 2))
```

Step 8: Extracting the state from purchase_address

- The final cleaned and transformed DataFrame is saved as a Parquet file for further processing by the Machine Learning team.

Code:

```
# Path to save the Parquet file
parquet_path = '/orders_data.parquet'

# Save DataFrame as Parquet
orders_data.write.mode('overwrite').parquet(parquet_path)
```

Conclusion:

In this task, we successfully cleaned and transformed the order data by following a systematic approach. We handled time-based filtering, standardized data formats, and extracted relevant information from existing columns to meet the specified requirements. The final output, a Parquet file, is well-structured and optimized for further analysis or use in machine learning models. This process ensured that the data is in a clean and consistent state, ready for subsequent stages of data processing and modeling.