code-for-transeformation-by-rajni

October 19, 2025

[0]:
```
spark
```

[0]:
```
<pyspark.sql.connect.session.SparkSession at 0x7f0f7eb66e30>
```

[0]:
```
storage_account = "roliststoragedataaccount"
application_id = "07df566b-31d4-47a3-ab29-4f6f32eba579"
directory_id = "abde9a70-abf5-43e7-a0bf-128f4bd4f146"

spark.conf.set(f"fs.azure.account.auth.type.{storage_account}.dfs.core.windows.
 ↪net", "OAuth")
spark.conf.set(f"fs.azure.account.oauth.provider.type.{storage_account}.dfs.
 ↪core.windows.net", "org.apache.hadoop.fs.azurebfs.oauth2.
 ↪ClientCredsTokenProvider")
spark.conf.set(f"fs.azure.account.oauth2.client.id.{storage_account}.dfs.core.
 ↪windows.net", application_id)
spark.conf.set(f"fs.azure.account.oauth2.client.secret.{storage_account}.dfs.
 ↪core.windows.net", "rdg8Q~auwb6IC1LLCZjZQnfBa2pszkAyQJNLMbg.")
spark.conf.set(f"fs.azure.account.oauth2.client.endpoint.{storage_account}.dfs.
 ↪core.windows.net", f"https://login.microsoftonline.com/{directory_id}/oauth2/
 ↪token")
```

[0]:
```
customers_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_customers_dataset.csv")

display(customers_df)
```

##Reading the Data

[0]:
```
sellers_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_sellers_dataset.csv")
```

```
display(sellers_df)



geolocation_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_geolocation_dataset.csv")

display(geolocation_df)



payments_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_order_payments_dataset.csv")

display(payments_df)


reviews_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_order_reviews_dataset.csv")

display(reviews_df)


orders_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_orders_dataset.csv")

display(orders_df)

products_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_products_dataset.csv")

display(products_df)
```

```
items_df = spark.read.format("csv") \
    .option("header", "true") \
    .option("inferSchema", "true") \
    .load("abfss://olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.
 ↪net/bronze/olist_order_items_dataset.csv")

display(items_df)
```

[0]:
```
import pymongo
from pymongo import MongoClient
```

###Reading Data from Pymongo

[0]:
```
# importing module
from pymongo import MongoClient

hostname = "v658py.h.filess.io"
database = "olistdatanosql_gulfartnow"
port = "61034"
username = "olistdatanosql_gulfartnow"
password = "8306dc728c0276238b01de3484cb009d44b4d508"

uri = "mongodb://" + username + ":" + password + "@" + hostname + ":" + port +␣
 ↪"/" + database

# Connect with the portnumber and host
client = MongoClient(uri)

# Access database
mydatabase = client[database]
mydatabase
```

[0]:
```
Database(MongoClient(host=['v658py.h.filess.io:61034'], document_class=dict,
tz_aware=False, connect=True), 'olistdatanosql_gulfartnow')
```

[0]:
```
import pandas as pd
collection = mydatabase['product_categories']

mongo_data = pd.DataFrame(list(collection.find()))
```

[0]:
```
mongo_data
```

[0]:
```
                        _id  … product_category_name_english
0  68f093f415d9a5b5a8dd336d  …                  health_beauty
1  68f093f415d9a5b5a8dd336e  …         computers_accessories
2  68f093f415d9a5b5a8dd336f  …                          auto
```

```
3   68f093f415d9a5b5a8dd3370  …                bed_bath_table
4   68f093f415d9a5b5a8dd3371  …              furniture_decor
..                       …  …                           …
66  68f093f415d9a5b5a8dd33af  …                      flowers
67  68f093f415d9a5b5a8dd33b0  …       arts_and_craftmanship
68  68f093f415d9a5b5a8dd33b1  …         diapers_and_hygiene
69  68f093f415d9a5b5a8dd33b2  …   fashion_childrens_clothes
70  68f093f415d9a5b5a8dd33b3  …       security_and_services

[71 rows x 3 columns]
```

[0]:
```
display(products_df)
```

0.0.1 cleaning the data

[0]:
```
from pyspark.sql.functions import col, to_date, datediff, current_date, when
```

[0]:
```
def clean_datafram(df, name):
    print("Cleaning", name)
    return df.dropDuplicates().na.drop('all')

orders_df = clean_datafram(orders_df, "Orders")
display(orders_df)
```

```
Cleaning Orders
```

[0]:
```
# convert date columns
orderes_df = orders_df.withColumn("order_purchase_timestamp",␣
 ↪to_date(col("order_purchase_timestamp")))\
    .withColumn("order_delivered_customer_date",␣
 ↪to_date(col("order_delivered_customer_date")))\
        .withColumn("order_estimated_delivery_date",␣
 ↪to_date(col("order_estimated_delivery_date")))
```

[0]:
```
orderes_df = orderes_df.withColumn(
    "actual_delivery_time",
    datediff(
        col("order_delivered_customer_date"),
        col("order_purchase_timestamp")
    )
)
orderes_df = orderes_df.withColumn(
    "estimated_delivery_time",
    datediff(
        col("order_estimated_delivery_date"),
        col("order_purchase_timestamp")
    )
```

```
)
orderes_df = orderes_df.withColumn(
    "delay_time", col("actual_delivery_time") - col("estimated_delivery_time"),
)
display(orderes_df)
```

0.0.2 Joining

[0]:
```
from pyspark.sql.functions import col

# 1  Join Orders + Customers
orders_customer_df = (
    orders_df.alias("o")
    .join(
        customers_df.alias("c"),
        col("o.customer_id") == col("c.customer_id"),
        "left"
    )
    .select(
        "o.*",  # keep all order columns
        "c.customer_unique_id",
        "c.customer_zip_code_prefix",
        "c.customer_city",
        "c.customer_state"
    )
)

# 2  Join with Payments
orders_payment_df = (
    orders_customer_df.alias("oc")
    .join(
        payments_df.alias("p"),
        col("oc.order_id") == col("p.order_id"),
        "left"
    )
    .select(
        "oc.*",
        "p.payment_sequential",
        "p.payment_type",
        "p.payment_installments",
        "p.payment_value"
    )
)

# 3  Join with Order Items
orders_items_df = (
    orders_payment_df.alias("op")
```

```
    .join(
        items_df.alias("i"),
        col("op.order_id") == col("i.order_id"),
        "left"
    )
    .select(
        "op.*",
        "i.order_item_id",
        "i.product_id",
        "i.seller_id",
        "i.shipping_limit_date",
        "i.price",
        "i.freight_value"
    )
)

# 4  Join with Products
orders_items_products_df = (
    orders_items_df.alias("oi")
    .join(
        products_df.alias("pr"),
        col("oi.product_id") == col("pr.product_id"),
        "left"
    )
    .select(
        "oi.*",
        "pr.product_category_name",
        "pr.product_name_lenght",
        "pr.product_description_lenght",
        "pr.product_photos_qty",
        "pr.product_weight_g",
        "pr.product_length_cm",
        "pr.product_height_cm",
        "pr.product_width_cm"
    )
)

# 5  Join with Sellers
final_df = (
    orders_items_products_df.alias("f")
    .join(
        sellers_df.alias("s"),
        col("f.seller_id") == col("s.seller_id"),
        "left"
    )
    .select(
        "f.*",
```

```
        "s.seller_zip_code_prefix",
        "s.seller_city",
        "s.seller_state"
    )
)

#  Display & Verify
display(final_df.limit(5))
print(f"Final DataFrame has {len(final_df.columns)} columns")
final_df.printSchema()
```

```
Final DataFrame has 33 columns
root
 |-- order_id: string (nullable = true)
 |-- customer_id: string (nullable = true)
 |-- order_status: string (nullable = true)
 |-- order_purchase_timestamp: timestamp (nullable = true)
 |-- order_approved_at: timestamp (nullable = true)
 |-- order_delivered_carrier_date: timestamp (nullable = true)
 |-- order_delivered_customer_date: timestamp (nullable = true)
 |-- order_estimated_delivery_date: timestamp (nullable = true)
 |-- customer_unique_id: string (nullable = true)
 |-- customer_zip_code_prefix: integer (nullable = true)
 |-- customer_city: string (nullable = true)
 |-- customer_state: string (nullable = true)
 |-- payment_sequential: integer (nullable = true)
 |-- payment_type: string (nullable = true)
 |-- payment_installments: integer (nullable = true)
 |-- payment_value: double (nullable = true)
 |-- order_item_id: integer (nullable = true)
 |-- product_id: string (nullable = true)
 |-- seller_id: string (nullable = true)
 |-- shipping_limit_date: timestamp (nullable = true)
 |-- price: double (nullable = true)
 |-- freight_value: double (nullable = true)
 |-- product_category_name: string (nullable = true)
 |-- product_name_lenght: integer (nullable = true)
 |-- product_description_lenght: integer (nullable = true)
 |-- product_photos_qty: integer (nullable = true)
 |-- product_weight_g: integer (nullable = true)
 |-- product_length_cm: integer (nullable = true)
 |-- product_height_cm: integer (nullable = true)
 |-- product_width_cm: integer (nullable = true)
 |-- seller_zip_code_prefix: integer (nullable = true)
 |-- seller_city: string (nullable = true)
 |-- seller_state: string (nullable = true)
```

[0]:
```
display(final_df)
```

[0]:
```
mongo_data.drop('_id', axis = 1, inplace = True)
mongo_spark_df = spark.createDataFrame(mongo_data)
display(mongo_spark_df)
```

[0]:
```
final_df = final_df.join(mongo_spark_df, "product_category_name", "left")
```

[0]:
```
display(final_df)
```

[0]:
```
final_df.write.mode("overwrite").parquet("abfss://
  ↪olistdatabyrajni@roliststoragedataaccount.dfs.core.windows.net/silver")
```