PySpark E-Commerce Tutorial 🚀

This notebook demonstrates how to use **PySpark** to process and analyze e-commerce data in a clean, production-oriented way.

🚢Objectives

- Initialize a Spark Session correctly
- Load raw CSV data into Spark DataFrames
- Explore schemas and validate data quality
- Apply transformations and aggregations using PySpark APIs
- Write clean, readable, and well-documented PySpark code

This notebook is suitable for: - Beginners learning PySpark fundamentals - Junior → Mid Data Engineers - Anyone preparing for Data Engineering interviews

---

Initialize Spark Session

```python
from pyspark.sql import SparkSession

# Create Spark Session
# appName: Logical name for the Spark application
# master: local[*] means use all available CPU cores
spark = (
    SparkSession.builder
    .appName("e-commerce-analysis")
    .master("local[*]")
    .getOrCreate()
)

print("Spark Version:", spark.version)
```

Load Customers Data

```python
# Read customers data from CSV file
# header=True      -> First row contains column names
# inferSchema=True -> Spark automatically infers column data types

df_customers = (
    spark.read
    .option("header", True)
    .option("inferSchema", True)
    .csv("customers.csv")
)

# Inspect schema to understand data structure
df_customers.printSchema()

# Preview a sample record
df_customers.show(1, truncate=False)
```

Load Orders Data

```
# Read orders dataset

df_orders = (
    spark.read
    .option("header", True)
    .option("inferSchema", True)
    .csv("orders.csv")
)

# Check schema and sample rows
df_orders.printSchema()
df_orders.show(5, truncate=False)
```

Basic Data Exploration 🏫

```
# Total number of customers
customers_count = df_customers.count()

# Total number of orders
orders_count = df_orders.count()

print(f"Total Customers: {customers_count}")
print(f"Total Orders: {orders_count}")
```

Orders by Status

Understanding order lifecycle is a common analytical use case in e-commerce systems.

```
from pyspark.sql.functions import col

# Count number of orders per order_status
orders_by_status = (
    df_orders
    .groupBy("order_status")
    .count()
    .orderBy(col("count").desc())
)

orders_by_status.show()
```

Join Customers with Orders

Joining datasets is a core operation in data engineering pipelines.

```
# Join orders with customers on customer_id

df_orders_customers = (
    df_orders
    .join(df_customers, on="customer_id", how="inner")
)

# Validate join result
df_orders_customers.show(5, truncate=False)
```

Business Insight Example

Number of Orders per Customer

```
# Aggregate orders per customer
orders_per_customer = (
    df_orders_customers
    .groupBy("customer_id")
    .count()
    .withColumnRenamed("count", "total_orders")
    .orderBy(col("total_orders").desc())
)

orders_per_customer.show(10)
```

Performance Notes

- Spark uses **lazy evaluation**, so transformations are only executed when an action is triggered
- Use `.select()` to limit columns early and reduce data shuffling
- Partitioning & caching can significantly improve performance in large datasets

---

Conclusion

In this notebook, we covered essential PySpark concepts used in real-world Data Engineering: - Reading structured data - Schema exploration - Aggregations and joins - Writing clean and maintainable Spark code

🛣️Next Steps

- Add partitioning and caching strategies
- Store processed data in Parquet format
- Load final datasets into a Data Warehouse (PostgreSQL / BigQuery)

📌*Learning PySpark is a journey — consistency and practice make the difference.*