```
#PySpark Coding Assessment-1 Master Task Set
#Ingestion, Transformation, Spark SQL, Aggregations, Joins, UDFs, and Storage


from pyspark.sql import SparkSession
import pandas as pd
from io import StringIO

spark=SparkSession.builder.appName("PySpark Assessment").getOrCreate()

print(spark)
```

```
<pyspark.sql.session.SparkSession object at 0x7bce726f9b10>
```

```
#Task 1- Data Ingestion & Exploration

# Saving sample datasets
customers_data = """CustomerID,Name,Email,City,SignupDate
101,Ali,ali@gmail.com,Mumbai,2022-05-10
102,Neha,neha@yahoo.com,Delhi,2023-01-15
103,Ravi,ravi@hotmail.com,Bangalore,2021-11-01
104,Sneha,sneha@outlook.com,Hyderabad,2020-07-22
105,Amit,amit@gmail.com,Chennai,2023-03-10"""

orders_data = """OrderID,CustomerID,Product,Category,Quantity,Price,OrderDate
1,101,Laptop,Electronics,2,50000.0,2024-01-10
2,101,Mouse,Electronics,1,1200.0,2024-01-15
3,102,Tablet,Electronics,1,20000.0,2024-02-01
4,103,Bookshelf,Furniture,1,3500.0,2024-02-10
5,104,Mixer,Appliances,1,5000.0,2024-02-15
6,105,Notebook,Stationery,5,500.0,2024-03-01
7,102,Phone,Electronics,1,30000.0,2024-03-02"""

with open("customers.csv", "w") as f:
    f.write(customers_data)

with open("orders.csv", "w") as f:
    f.write(orders_data)

# Loading CSV with schema inference
customers = spark.read.csv("customers.csv", header=True, inferSchema=True)
orders = spark.read.csv("orders.csv", header=True, inferSchema=True)

# Show schema
customers.printSchema()
orders.printSchema()

# Count total customers and orders
print("Customers count:", customers.count())
print("Orders count:", orders.count())

# Distinct cities
distinct_cities=customers.select("City").distinct()
print("Distinct cities:")
distinct_cities.show()
```

```
root
 |-- CustomerID: integer (nullable = true)
 |-- Name: string (nullable = true)
 |-- Email: string (nullable = true)
 |-- City: string (nullable = true)
 |-- SignupDate: date (nullable = true)

root
 |-- OrderID: integer (nullable = true)
 |-- CustomerID: integer (nullable = true)
 |-- Product: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- Price: double (nullable = true)
 |-- OrderDate: date (nullable = true)

Customers count: 5
Orders count: 7
Distinct cities:
+---------+
|     City|
```

```
+---------+
|Bangalore|
|  Chennai|
|   Mumbai|
|    Delhi|
|Hyderabad|
+---------+
```

```
#Task 2- DataFrame Transformations
from pyspark.sql.functions import col, year

# Add a column TotalAmount = Price * Quantity
orders = orders.withColumn("TotalAmount", col("Price") * col("Quantity"))
orders.show()

# Create a new column OrderYear from OrderDate
orders = orders.withColumn("OrderYear", year("OrderDate"))
orders.show()

# Filter orders with TotalAmount > 10,000
orders_filtered = orders.filter(col("TotalAmount") > 10000)
orders_filtered.show()

# Drop the Email column from customers
customers = customers.drop("Email")
customers.show()
```

```
+-------+----------+---------+-----------+--------+-------+----------+-----------+
|OrderID|CustomerID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|
+-------+----------+---------+-----------+--------+-------+----------+-----------+
|      1|       101|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|
|      2|       101|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|
|      3|       102|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|
|      4|       103|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|
|      5|       104|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|
|      6|       105| Notebook| Stationery|       5|  500.0|2024-03-01|     2500.0|
|      7|       102|    Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|
+-------+----------+---------+-----------+--------+-------+----------+-----------+

+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+
|OrderID|CustomerID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+
|      1|       101|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|
|      2|       101|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|     2024|
|      3|       102|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|
|      4|       103|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|     2024|
|      5|       104|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|     2024|
|      6|       105| Notebook| Stationery|       5|  500.0|2024-03-01|     2500.0|     2024|
|      7|       102|    Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|     2024|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+

+-------+----------+-------+-----------+--------+-------+----------+-----------+---------+
|OrderID|CustomerID|Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|
+-------+----------+-------+-----------+--------+-------+----------+-----------+---------+
|      1|       101| Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|
|      3|       102| Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|
|      7|       102|  Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|     2024|
+-------+----------+-------+-----------+--------+-------+----------+-----------+---------+

+----------+-----+---------+----------+
|CustomerID| Name|     City|SignupDate|
+----------+-----+---------+----------+
|       101|  Ali|   Mumbai|2022-05-10|
|       102| Neha|    Delhi|2023-01-15|
|       103| Ravi|Bangalore|2021-11-01|
|       104|Sneha|Hyderabad|2020-07-22|
|       105| Amit|  Chennai|2023-03-10|
+----------+-----+---------+----------+
```

```
#Task 3- Handling Nulls & Conditionals
from pyspark.sql.functions import when, lit, to_date

#Simulate a null in City and fill it with “Unknown”.
customers = customers.withColumn("City", when(col("CustomerID") == 102, None).otherwise(col("City")))
customers = customers.fillna({"City": "Unknown"})
customers.show()

#Label customers as “Loyal” if SignupDate is before 2022, else “New”.
```

```
customers = customers.withColumn("CustomerType", when(to_date("SignupDate") < "2022-01-01", "Loyal").otherwise("New"))
customers.show()

#Create OrderType column: "Low" if < 5,000, "High" if ≥ 5,000.
orders = orders.withColumn("OrderType", when(col("TotalAmount") < 5000, "Low").otherwise("High"))
orders.show()
```

```
+----------+-----+---------+----------+------------+
|CustomerID| Name|     City|SignupDate|CustomerType|
+----------+-----+---------+----------+------------+
|       101|  Ali|   Mumbai|2022-05-10|         New|
|       102| Neha|  Unknown|2023-01-15|         New|
|       103| Ravi|Bangalore|2021-11-01|       Loyal|
|       104|Sneha|Hyderabad|2020-07-22|       Loyal|
|       105| Amit|  Chennai|2023-03-10|         New|
+----------+-----+---------+----------+------------+

+----------+-----+---------+----------+------------+
|CustomerID| Name|     City|SignupDate|CustomerType|
+----------+-----+---------+----------+------------+
|       101|  Ali|   Mumbai|2022-05-10|         New|
|       102| Neha|  Unknown|2023-01-15|         New|
|       103| Ravi|Bangalore|2021-11-01|       Loyal|
|       104|Sneha|Hyderabad|2020-07-22|       Loyal|
|       105| Amit|  Chennai|2023-03-10|         New|
+----------+-----+---------+----------+------------+

+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+
|OrderID|CustomerID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|OrderType|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+
|      1|       101|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|     High|
|      2|       101|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|     2024|      Low|
|      3|       102|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|     High|
|      4|       103|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|     2024|      Low|
|      5|       104|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|     2024|     High|
|      6|       105| Notebook| Stationery|       5|  500.0|2024-03-01|     2500.0|     2024|      Low|
|      7|       102|    Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|     2024|     High|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+
```

```
#Task 4- Joins & Aggregations
from pyspark.sql.functions import sum as _sum, count as _count

# Join
joined = orders.join(customers, on="CustomerID", how="inner")
joined.show()

# Total orders and revenue per city
orders_revenue_city= joined.groupBy("City").agg(
    _count("OrderID").alias("TotalOrders"),
    _sum("TotalAmount").alias("Revenue")
)
orders_revenue_city.show()

# Top 3 customers by total spend
customers_by_total_spend= joined.groupBy("CustomerID", "Name").agg(
    _sum("TotalAmount").alias("TotalSpend")
).orderBy(col("TotalSpend").desc())
customers_by_total_spend.show(3)

# Count products sold per category
product_per_category = orders.groupBy("Category").agg(_count("Product").alias("ProductsSold"))
product_per_category.show()
```

```
+----------+-------+---------+-----------+--------+-------+----------+-----------+---------+---------+-----+---------+----------+-------
|CustomerID|OrderID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|OrderType| Name|     City|SignupDate|Custome
+----------+-------+---------+-----------+--------+-------+----------+-----------+---------+---------+-----+---------+----------+-------
|       101|      1|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|     High|  Ali|   Mumbai|2022-05-10|
|       101|      2|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|     2024|      Low|  Ali|   Mumbai|2022-05-10|
|       102|      3|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|     High| Neha|  Unknown|2023-01-15|
|       103|      4|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|     2024|      Low| Ravi|Bangalore|2021-11-01|
|       104|      5|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|     2024|     High|Sneha|Hyderabad|2020-07-22|
|       105|      6| Notebook| Stationery|       5|  500.0|2024-03-01|     2500.0|     2024|      Low| Amit|  Chennai|2023-03-10|
|       102|      7|    Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|     2024|     High| Neha|  Unknown|2023-01-15|
+----------+-------+---------+-----------+--------+-------+----------+-----------+---------+---------+-----+---------+----------+-------

+---------+-----------+--------+
|     City|TotalOrders| Revenue|
+---------+-----------+--------+
|Bangalore|          1|  3500.0|
|  Chennai|          1|  2500.0|
```

```
|   Mumbai|          2|101200.0|
|  Unknown|          2| 50000.0|
|Hyderabad|          1|  5000.0|
+---------+-----------+--------+

+----------+-----+----------+
|CustomerID| Name|TotalSpend|
+----------+-----+----------+
|       101|  Ali|  101200.0|
|       102| Neha|   50000.0|
|       104|Sneha|    5000.0|
+----------+-----+----------+
only showing top 3 rows

+-----------+------------+
|   Category|ProductsSold|
+-----------+------------+
| Stationery|           1|
|Electronics|           4|
|  Furniture|           1|
| Appliances|           1|
+-----------+------------+
```

```
spark.sql("SELECT * FROM customers").show()
spark.sql("SELECT * FROM orders").show()
```

```
+----------+-----+---------+----------+------------+
|CustomerID| Name|     City|SignupDate|CustomerType|
+----------+-----+---------+----------+------------+
|       101|  Ali|   Mumbai|2022-05-10|         New|
|       102| Neha|  Unknown|2023-01-15|         New|
|       103| Ravi|Bangalore|2021-11-01|       Loyal|
|       104|Sneha|Hyderabad|2020-07-22|       Loyal|
|       105| Amit|  Chennai|2023-03-10|         New|
+----------+-----+---------+----------+------------+

+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+
|OrderID|CustomerID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|OrderType|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+
|      1|       101|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|     High|
|      2|       101|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|     2024|      Low|
|      3|       102|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|     High|
|      4|       103|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|     2024|      Low|
|      5|       104|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|     2024|     High|
|      6|       105| Notebook| Stationery|       5|  500.0|2024-03-01|     2500.0|     2024|      Low|
|      7|       102|    Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|     2024|     High|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+
```

```
#Task 5-Spark SQL Tasks
# Create Database sales and tables
spark.sql("CREATE DATABASE IF NOT EXISTS sales")
spark.sql("USE sales")
customers.write.mode("overwrite").saveAsTable("sales.customers")
orders.write.mode("overwrite").saveAsTable("sales.orders")

# 1. List of Orders by Delhi customers
list_delhi = spark.sql("""
SELECT o.* FROM orders o
JOIN customers c ON o.CustomerID = c.CustomerID
WHERE c.City = 'Delhi'
""")
list_delhi.show()

# 2. Average order value per category
avg_order_value = spark.sql("""
SELECT Category, AVG(Price * Quantity) as AvgOrderValue
FROM orders GROUP BY Category
""")
avg_order_value.show()

# 3. Monthly orders view
spark.sql("""
CREATE OR REPLACE TEMP VIEW monthly_orders AS
SELECT MONTH(OrderDate) as Month, SUM(Price * Quantity) as TotalAmount
FROM orders GROUP BY MONTH(OrderDate)
```

```
""")
spark.sql("SELECT * FROM monthly_orders").show()
```

```
+-------+----------+-------+--------+--------+-----+---------+-----------+---------+---------+
|OrderID|CustomerID|Product|Category|Quantity|Price|OrderDate|TotalAmount|OrderYear|OrderType|
+-------+----------+-------+--------+--------+-----+---------+-----------+---------+---------+
+-------+----------+-------+--------+--------+-----+---------+-----------+---------+---------+

+-----------+-------------+
|   Category|AvgOrderValue|
+-----------+-------------+
| Stationery|       2500.0|
|Electronics|      37800.0|
|  Furniture|       3500.0|
| Appliances|       5000.0|
+-----------+-------------+

+-----+-----------+
|Month|TotalAmount|
+-----+-----------+
|    1|   101200.0|
|    3|    32500.0|
|    2|    28500.0|
+-----+-----------+
```

```
#Task 6-String & Date Functions
from pyspark.sql.functions import regexp_extract, concat_ws, datediff, current_date, month, date_format, concat, lit

# Mask emails using regex (e.g., a***@gmail.com )
first_char = regexp_extract("Email", r"^(.).*@", 1)
domain = regexp_extract("Email", r"(@.*)", 1)
masked = customers.withColumn("MaskedEmail", concat(first_char, lit("***"), domain))
masked.select("Email", "MaskedEmail").show(truncate=False)

# Concatenate Name and City as “Name from City”
customers = customers.withColumn("Summary", concat_ws(" from ", "Name", "City"))
customers.show()

# Use datediff() to calculate customer age in days
customers = customers.withColumn("AgeInDays", datediff(current_date(), to_date("SignupDate")))
customers.show()

# Extract month name from OrderDate
orders = orders.withColumn("MonthName", date_format("OrderDate", "MMMM"))
orders.show()
```

```
+-----------------+----------------+
|Email            |MaskedEmail     |
+-----------------+----------------+
|ali@gmail.com    |a***@gmail.com  |
|neha@yahoo.com   |n***@yahoo.com  |
|ravi@hotmail.com |r***@hotmail.com|
|sneha@outlook.com|s***@outlook.com|
|amit@gmail.com   |a***@gmail.com  |
+-----------------+----------------+

+----------+-----+-----------------+---------+----------+--------------------+---------+
|CustomerID| Name|            Email|     City|SignupDate|             Summary|AgeInDays|
+----------+-----+-----------------+---------+----------+--------------------+---------+
|       101|  Ali|    ali@gmail.com|   Mumbai|2022-05-10|     Ali from Mumbai|     1126|
|       102| Neha|   neha@yahoo.com|    Delhi|2023-01-15|     Neha from Delhi|      876|
|       103| Ravi| ravi@hotmail.com|Bangalore|2021-11-01| Ravi from Bangalore|     1316|
|       104|Sneha|sneha@outlook.com|Hyderabad|2020-07-22|Sneha from Hyderabad|     1783|
|       105| Amit|   amit@gmail.com|  Chennai|2023-03-10|   Amit from Chennai|      822|
+----------+-----+-----------------+---------+----------+--------------------+---------+

+----------+-----+-----------------+---------+----------+--------------------+---------+
|CustomerID| Name|            Email|     City|SignupDate|             Summary|AgeInDays|
+----------+-----+-----------------+---------+----------+--------------------+---------+
|       101|  Ali|    ali@gmail.com|   Mumbai|2022-05-10|     Ali from Mumbai|     1126|
|       102| Neha|   neha@yahoo.com|    Delhi|2023-01-15|     Neha from Delhi|      876|
|       103| Ravi| ravi@hotmail.com|Bangalore|2021-11-01| Ravi from Bangalore|     1316|
|       104|Sneha|sneha@outlook.com|Hyderabad|2020-07-22|Sneha from Hyderabad|     1783|
|       105| Amit|   amit@gmail.com|  Chennai|2023-03-10|   Amit from Chennai|      822|
+----------+-----+-----------------+---------+----------+--------------------+---------+

+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+---------+
|OrderID|CustomerID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|OrderType|MonthName|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+---------+
|      1|       101|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|     High|  January|
```

```
|      2|       101|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|     2024|      Low|  January|
|      3|       102|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|     High| February|
|      4|       103|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|     2024|      Low| February|
|      5|       104|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|     2024|     High| February|
|      6|       105| Notebook| Stationery|       5|  500.0|2024-03-01|     2500.0|     2024|      Low|    March|
|      7|       102|    Phone|Electronics|       1|30000.0|2024-03-02|    30000.0|     2024|     High|    March|
+-------+----------+---------+-----------+--------+-------+----------+-----------+---------+---------+---------+
```

```
#Task 7- UDFs and Complex Logic
from pyspark.sql.functions import udf
from pyspark.sql.types import StringType

# 1. Tagging customers
def tag_customer(spend):
    if spend > 50000:
        return "Gold"
    elif spend >= 10000:
        return "Silver"
    return "Bronze"

tag_udf = udf(tag_customer, StringType())

# Join and aggregate spend
spend_df = joined.groupBy("CustomerID", "Name").agg(_sum("TotalAmount").alias("Spend"))
spend_df = spend_df.withColumn("Tier", tag_udf(col("Spend")))
spend_df.show()

# 2. Shorten product names
def short_name(name):
    return name[:3] + "..." if name else ""

short_udf = udf(short_name, StringType())
orders = orders.withColumn("ShortProduct", short_udf(col("Product")))
orders.select("Product", "ShortProduct").show()
```

```
+----------+-----+--------+------+
|CustomerID| Name|   Spend|  Tier|
+----------+-----+--------+------+
|       105| Amit|  2500.0|Bronze|
|       104|Sneha|  5000.0|Bronze|
|       101|  Ali|101200.0|  Gold|
|       102| Neha| 50000.0|Silver|
|       103| Ravi|  3500.0|Bronze|
+----------+-----+--------+------+

+---------+------------+
|  Product|ShortProduct|
+---------+------------+
|   Laptop|      Lap...|
|    Mouse|      Mou...|
|   Tablet|      Tab...|
|Bookshelf|      Boo...|
|    Mixer|      Mix...|
| Notebook|      Not...|
|    Phone|      Pho...|
+---------+------------+
```

```
#Task 8- Parquet & Views
import time

# Save the joined result as a Parquet file.
joined.write.mode("overwrite").parquet("/content/joined_data.parquet")

# Read it back and verify schema.
parquet_df = spark.read.parquet("/content/joined_data.parquet")
parquet_df.printSchema()

# Create and query a global temp view.
parquet_df.createOrReplaceGlobalTempView("global_joined")
spark.sql("SELECT * FROM global_temp.global_joined LIMIT 5").show()

# Compare performance between CSV read and Parquet read.
start = time.time()
spark.read.csv("orders.csv", header=True, inferSchema=True).count()
print("CSV read time:", time.time() - start)
```

```
start = time.time()
spark.read.parquet("/content/joined_data.parquet").count()
print("Parquet read time:", time.time() - start)
```

```
root
 |-- CustomerID: integer (nullable = true)
 |-- OrderID: integer (nullable = true)
 |-- Product: string (nullable = true)
 |-- Category: string (nullable = true)
 |-- Quantity: integer (nullable = true)
 |-- Price: double (nullable = true)
 |-- OrderDate: date (nullable = true)
 |-- TotalAmount: double (nullable = true)
 |-- OrderYear: integer (nullable = true)
 |-- OrderType: string (nullable = true)
 |-- Name: string (nullable = true)
 |-- City: string (nullable = true)
 |-- SignupDate: date (nullable = true)
 |-- CustomerType: string (nullable = true)

+----------+-------+---------+-----------+--------+-------+----------+-----------+---------+---------+-----+---------+----------+-------
|CustomerID|OrderID|  Product|   Category|Quantity|  Price| OrderDate|TotalAmount|OrderYear|OrderType| Name|     City|SignupDate|Custome
+----------+-------+---------+-----------+--------+-------+----------+-----------+---------+---------+-----+---------+----------+-------
|       101|      1|   Laptop|Electronics|       2|50000.0|2024-01-10|   100000.0|     2024|     High|  Ali|   Mumbai|2022-05-10|
|       101|      2|    Mouse|Electronics|       1| 1200.0|2024-01-15|     1200.0|     2024|      Low|  Ali|   Mumbai|2022-05-10|
|       102|      3|   Tablet|Electronics|       1|20000.0|2024-02-01|    20000.0|     2024|     High| Neha|  Unknown|2023-01-15|
|       103|      4|Bookshelf|  Furniture|       1| 3500.0|2024-02-10|     3500.0|     2024|      Low| Ravi|Bangalore|2021-11-01|
|       104|      5|    Mixer| Appliances|       1| 5000.0|2024-02-15|     5000.0|     2024|     High|Sneha|Hyderabad|2020-07-22|
+----------+-------+---------+-----------+--------+-------+----------+-----------+---------+---------+-----+---------+----------+-------

CSV read time: 0.8827962875366211
Parquet read time: 0.5064787864685059
```