Module 1 - Data Ingestion And Exploration

April 19, 2025

[1]:
```python
from pyspark.sql import SparkSession
spark = SparkSession.builder \
.appName('OlistData') \
.getOrCreate()
```

```
25/04/12 15:54:45 WARN SparkSession: Using an existing Spark session; only
runtime SQL configurations will take effect.
```

[2]:
```python
spark
```

[2]:
```
<pyspark.sql.session.SparkSession at 0x7f7610b34cd0>
```

[3]:
```
!hadoop fs -ls /data/olist/
```

```
Found 9 items
-rw-r--r--   2 klpd1299 hadoop    9033957 2025-04-12 15:51
/data/olist/olist_customers_dataset.csv
-rw-r--r--   2 klpd1299 hadoop   61273883 2025-04-12 15:51
/data/olist/olist_geolocation_dataset.csv
-rw-r--r--   2 klpd1299 hadoop   15438671 2025-04-12 15:51
/data/olist/olist_order_items_dataset.csv
-rw-r--r--   2 klpd1299 hadoop    5777138 2025-04-12 15:51
/data/olist/olist_order_payments_dataset.csv
-rw-r--r--   2 klpd1299 hadoop   14451670 2025-04-12 15:51
/data/olist/olist_order_reviews_dataset.csv
-rw-r--r--   2 klpd1299 hadoop   17654914 2025-04-12 15:51
/data/olist/olist_orders_dataset.csv
-rw-r--r--   2 klpd1299 hadoop    2379446 2025-04-12 15:51
/data/olist/olist_products_dataset.csv
-rw-r--r--   2 klpd1299 hadoop     174703 2025-04-12 15:51
/data/olist/olist_sellers_dataset.csv
-rw-r--r--   2 klpd1299 hadoop       2613 2025-04-12 15:51
/data/olist/product_category_name_translation.csv
```

[4]:
```python
hdfs_path = '/data/olist/'
```

[6]:
```python
customer_df = spark.read.csv(hdfs_path + 'olist_customers_dataset.
 ↪csv',header=True,inferSchema=True)
```

[7]:
```
customer_df.show(5)
```

```
[Stage 2:>                                                          (0 + 1) / 1]

+--------------------+--------------------+------------------------+------------
--------+--------------+
|         customer_id|  customer_unique_id|customer_zip_code_prefix|
customer_city|customer_state|
+--------------------+--------------------+------------------------+------------
--------+--------------+
|06b8999e2fba1a1fb…|861eff4711a542e4b…|                   14409|
franca|            SP|
|18955e83d337fd6b2…|290c77bc529b7ac93…|                    9790|sao bernardo
do c…|            SP|
|4e7b3e00288586ebd…|060e732b5b29e8181…|                    1151|
sao paulo|            SP|
|b2b6027bc5c5109e5…|259dac757896d24d7…|                    8775|     mogi
das cruzes|            SP|
|4f2d8ab171c80ec83…|345ecd01c38d18a90…|                   13056|
campinas|            SP|
+--------------------+--------------------+------------------------+------------
--------+--------------+
only showing top 5 rows
```

[8]:
```
category_translation_df = spark.read.csv(hdfs_path +␣
 ↪'product_category_name_translation.csv',header=True,inferSchema=True)
orders_df = spark.read.csv(hdfs_path + 'olist_orders_dataset.
 ↪csv',header=True,inferSchema=True)
order_item_df = spark.read.csv(hdfs_path + 'olist_order_items_dataset.
 ↪csv',header=True,inferSchema=True)
payments_df = spark.read.csv(hdfs_path + 'olist_order_payments_dataset.
 ↪csv',header=True,inferSchema=True)
reviews_df = spark.read.csv(hdfs_path + 'olist_order_reviews_dataset.
 ↪csv',header=True,inferSchema=True)
products_df = spark.read.csv(hdfs_path + 'olist_products_dataset.
 ↪csv',header=True,inferSchema=True)
sellers_df = spark.read.csv(hdfs_path + 'olist_sellers_dataset.
 ↪csv',header=True,inferSchema=True)
geolocation_df = spark.read.csv(hdfs_path + 'olist_geolocation_dataset.
 ↪csv',header=True,inferSchema=True)
```

[10]:
```
customer_df.printSchema()
```

```
root
```

```
 |-- customer_id: string (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)
```

[11]:
```
orders_df.printSchema()
```

```
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)
```

[12]:
```
# Data leakage or drop
print(f'Customers : {customer_df.count()} rows')
print(f'Orders : {orders_df.count()} rows')
```

```
Customers : 99441 rows
Orders : 99441 rows
```

[15]:
```
# NULL or say duplicate values

from pyspark.sql.functions import col

customer_df.select([col(c).isNull().alias(c) for c in customer_df.columns]).
  ↪show()
```

```
+-----------+------------------+------------------------+-------------+---------
-----+
|customer_id|customer_unique_id|customer_zip_code_prefix|customer_city|customer_
state|
+-----------+------------------+------------------------+-------------+---------
-----+
|      false|             false|                   false|        false|
false|
|      false|             false|                   false|        false|
false|
|      false|             false|                   false|        false|
false|
|      false|             false|                   false|        false|
false|
|      false|             false|                   false|        false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
|      false|                 false|                       false|            false|
false|
+-----------+------------------+------------------------+-------------+---------
-----+
only showing top 20 rows
```

[16]:
```
# NULL or say duplicate values

from pyspark.sql.functions import col,when,count

customer_df.select([count(when(col(c).isNull(),1)).alias(c) for c in␣
 ↪customer_df.columns]).show()
```

```
+-----------+------------------+------------------------+-------------+---------
-----+
|customer_id|customer_unique_id|customer_zip_code_prefix|customer_city|customer_
state|
+-----------+------------------+------------------------+-------------+---------
```

```
-----+
|          0|                 0|                      0|            0|
0|
+-----------+------------------+-----------------------+------------+---------
-----+
```

[17]:
```
# Duplicate values
customer_df.groupBy('customer_id').count().filter('count>1').show()
```

```
[Stage 29:=============================>                          (1 + 1) / 2]

+-----------+-----+
|customer_id|count|
+-----------+-----+
+-----------+-----+
```

[19]:
```
## customer distribute by state
customer_df.groupBy('customer_state').count().orderBy('count',ascending=True).
 ↪show()
```

```
+--------------+-----+
|customer_state|count|
+--------------+-----+
|            RR|   46|
|            AP|   68|
|            AC|   81|
|            AM|  148|
|            RO|  253|
|            TO|  280|
|            SE|  350|
|            AL|  413|
|            RN|  485|
|            PI|  495|
|            PB|  536|
|            MS|  715|
|            MA|  747|
|            MT|  907|
|            PA|  975|
|            CE| 1336|
|            PE| 1652|
|            GO| 2020|
|            ES| 2033|
|            DF| 2140|
+--------------+-----+
only showing top 20 rows
```

[21]:
```
orders_df.groupBy('order_status').count().orderBy('count',ascending=True).show()
```

```
+------------+-----+
|order_status|count|
+------------+-----+
|    approved|    2|
|     created|    5|
|  processing|  301|
|    invoiced|  314|
| unavailable|  609|
|    canceled|  625|
|     shipped| 1107|
|   delivered|96478|
+------------+-----+
```

[22]:
```
payments_df.groupBy('payment_type').count().orderBy('count',ascending=True).
  ↪show()
```

```
+------------+-----+
|payment_type|count|
+------------+-----+
| not_defined|    3|
|  debit_card| 1529|
|     voucher| 5775|
|      boleto|19784|
| credit_card|76795|
+------------+-----+
```

[23]:
```
from pyspark.sql.functions import sum

top_products = order_item_df.groupBy("product_id").agg(sum('price').
  ↪alias('total_sales'))
top_products.orderBy('total_sales',ascending=False).show(20)
```

```
[Stage 46:=============================>                           (1 + 1) / 2]

+--------------------+------------------+
|          product_id|       total_sales|
+--------------------+------------------+
|bb50f2e236e5eea01...|           63885.0|
|6cdd53843498f9289...| 54730.20000000005|
|d6160fb7873f18409...|48899.340000000004|
|d1c427060a0f73f6b...| 47214.51000000006|
|99a4788cb24856965...|43025.560000000085|
|3dd2a17168ec895c7...| 41082.60000000005|
|25c38557cf793876c...| 38907.32000000001|
|5f504b3a1c75b73d6...|37733.899999999994|
|53b36df67ebb7c415…| 37683.42000000001|
|aca2eb7d00ea1a7b8…| 37608.90000000007|
|e0d64dcfaa3b6db5c…|          31786.82|
|d285360f29ac7fd97…|31623.809999999983|
|7a10781637204d8d1…|           30467.5|
|f1c7f353075ce59d8…|          29997.36|
|f819f0c84a64f02d3…|29024.479999999996|
|588531f8ec37e7d5f…|28291.989999999998|
|422879e10f4668299…|26577.219999999972|
|16c4e87b98a9370a9…|           25034.0|
|5a848e4ab52fd5445…|24229.029999999962|
|a62e25e09e05e6faf…|           24051.0|
+--------------------+------------------+
only showing top 20 rows
```

[31]:
```
#Average Delivery Time Analysis
delivery_df = orders_df.
  ↪select('order_id','order_purchase_timestamp','order_delivered_customer_date').
  ↪show()
```

```
+--------------------+------------------------+-----------------------------+
|            order_id|order_purchase_timestamp|order_delivered_customer_date|
+--------------------+------------------------+-----------------------------+
|e481f51cbdc54678b…|     2017-10-02 10:56:33|          2017-10-10 21:25:13|
|53cdb2fc8bc7dce0b…|     2018-07-24 20:41:37|          2018-08-07 15:27:45|
|47770eb9100c2d0c4…|     2018-08-08 08:38:49|          2018-08-17 18:06:29|
|949d5b44dbf5de918…|     2017-11-18 19:28:06|          2017-12-02 00:28:42|
|ad21c59c0840e6cb8…|     2018-02-13 21:18:39|          2018-02-16 18:17:02|
|a4591c265e18cb1dc…|     2017-07-09 21:57:05|          2017-07-26 10:57:55|
|136cce7faa42fdb2c…|     2017-04-11 12:22:08|                         NULL|
|6514b8ad8028c9f2c…|     2017-05-16 13:10:30|          2017-05-26 12:55:51|
|76c6e866289321a7c…|     2017-01-23 18:29:09|          2017-02-02 14:08:10|
|e69bfb5eb88e0ed6a…|     2017-07-29 11:55:02|          2017-08-16 17:14:30|
|e6ce16cb79ec1d90b…|     2017-05-16 19:41:10|          2017-05-29 11:18:31|
|34513ce0c4fab462a…|     2017-07-13 19:58:11|          2017-07-19 14:04:48|
|82566a660a982b15f…|     2018-06-07 10:06:19|          2018-06-19 12:05:52|
|5ff96c15d0b717ac6…|     2018-07-25 17:44:10|          2018-07-30 15:52:25|
|432aaf21d85167c2c…|     2018-03-01 14:14:28|          2018-03-12 23:36:26|
|dcb36b511fcac050b…|     2018-06-07 19:03:12|          2018-06-21 15:34:32|
|403b97836b0c04a62…|     2018-01-02 19:00:43|          2018-01-20 01:38:59|
|116f0b09343b49556…|     2017-12-26 23:41:31|          2018-01-08 22:36:36|
|85ce859fd6dc634de…|     2017-11-21 00:03:41|          2017-11-27 18:28:00|
|83018ec114eee8641…|     2017-10-26 15:54:26|          2017-11-08 22:22:00|
+--------------------+------------------------+-----------------------------+
only showing top 20 rows
```

```
[ ]:
```