Module 1 - Data Ingestion and Exploration

March 7, 2025

0.0.1 Step 1: Setting Up the Spark Environment

In a real-world company setup, we wouldn't use Google Colab directly. Instead, we would:

1. **1. Deploy a Spark Cluster** (like AWS EMR, GCP Dataproc, or an on-prem Hadoop cluster, Azure HD Insight).
2. **2. Store Data in HDFS** instead of local storage.

- Load data from Kaggle i.e. Data Source (#!/bin/bash curl -L -o ~/olist/brazilian-ecommerce.zip\ https://www.kaggle.com/api/v1/datasets/download/olistbr/brazilian-ecommerce)
- `!unzip brazilian-ecommerce.zip -d ~/olist/data/`

3. **3. Use PySpark** to interact with data.

[ ]:
```
#!/bin/bash
curl -L -o ~/Downloads/brazilian-ecommerce.zip\
  https://www.kaggle.com/api/v1/datasets/download/olistbr/brazilian-ecommerce
```

[ ]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
.appName('OlistData') \
.getOrCreate()
```

[2]:
```
spark
```

[2]:
```
<pyspark.sql.session.SparkSession at 0x7f07f8292740>
```

[1]:
```
!hadoop fs -ls /data/olist/
```

```
Found 9 items
-rw-r--r--   2 mayank0953 hadoop    9033957 2025-02-21 10:51
/data/olist/olist_customers_dataset.csv
-rw-r--r--   2 mayank0953 hadoop   61273883 2025-02-21 10:51
/data/olist/olist_geolocation_dataset.csv
-rw-r--r--   2 mayank0953 hadoop   15438671 2025-02-21 10:51
/data/olist/olist_order_items_dataset.csv
-rw-r--r--   2 mayank0953 hadoop    5777138 2025-02-21 10:51
/data/olist/olist_order_payments_dataset.csv
```

```
-rw-r--r--   2 mayank0953 hadoop   14451670 2025-02-21 10:51
/data/olist/olist_order_reviews_dataset.csv
-rw-r--r--   2 mayank0953 hadoop   17654914 2025-02-21 10:51
/data/olist/olist_orders_dataset.csv
-rw-r--r--   2 mayank0953 hadoop    2379446 2025-02-21 10:51
/data/olist/olist_products_dataset.csv
-rw-r--r--   2 mayank0953 hadoop     174703 2025-02-21 10:51
/data/olist/olist_sellers_dataset.csv
-rw-r--r--   2 mayank0953 hadoop       2613 2025-02-21 10:51
/data/olist/product_category_name_translation.csv
```

[2]:
```
hdfs_path = '/data/olist/'
```

[3]:
```
customers_df = spark.read.csv(hdfs_path + 'olist_customers_dataset.
 ↪csv',header=True,inferSchema=True)
```

[4]:
```
customers_df.show(5)
```

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

[5]:
```
customers_df = spark.read.csv(hdfs_path + 'olist_customers_dataset.
 ↪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)
category_translation_df = spark.read.csv(hdfs_path +␣
 ↪'product_category_name_translation.csv',header=True,inferSchema=True)
```

[6]:
```
customers_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)
```

[7]:
```
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)
```

[8]:
```
# Data Lekage or Drop

print(f'Customers : {customers_df.count()} rows')
print(f'Orders : {orders_df.count()} rows')
```

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

[9]:
```
customers_df.columns
```

[9]:
```
['customer_id',
 'customer_unique_id',
 'customer_zip_code_prefix',
 'customer_city',
 'customer_state']
```

[12]:
```
from pyspark.sql.functions import col

# Check for nulls in critical fields
customers_df.select([col(c).isNull().alias(c) for c in customers_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|
|      false|             false|                  false|        false|
false|
|      false|             false|                  false|        false|
false|
+-----------+------------------+-----------------------+-------------+---------
-----+
only showing top 20 rows
```

[13]:
```
from pyspark.sql.functions import col,when ,count

# Check for nulls in critical fields
customers_df.select([count(when(col(c).isNull(),1)).alias(c) for c in␣
 ↪customers_df.columns]).show()
```

```
+-----------+------------------+-----------------------+-------------+---------
-----+
|customer_id|customer_unique_id|customer_zip_code_prefix|customer_city|customer_
state|
+-----------+------------------+-----------------------+-------------+---------
-----+
|          0|                 0|                      0|            0|
0|
+-----------+------------------+-----------------------+-------------+---------
-----+
```

[14]:
```
# Duplicate Values

customers_df.groupBy('customer_id').count().filter('count>1').show()
```

```
[Stage 37:==============================>                          (1 + 1) / 2]
+-----------+-----+
|customer_id|count|
+-----------+-----+
+-----------+-----+
```

[ ]:
```

```

[17]:
```
# Customer Distribution by state

customers_df.groupBy('customer_state').count().orderBy('count',ascending=False).
  ↪show()
```

```
+--------------+-----+
|customer_state|count|
+--------------+-----+
|            SP|41746|
|            RJ|12852|
|            MG|11635|
|            RS| 5466|
|            PR| 5045|
|            SC| 3637|
|            BA| 3380|
|            DF| 2140|
|            ES| 2033|
|            GO| 2020|
|            PE| 1652|
|            CE| 1336|
|            PA|  975|
|            MT|  907|
|            MA|  747|
|            MS|  715|
|            PB|  536|
|            PI|  495|
|            RN|  485|
|            AL|  413|
+--------------+-----+
only showing top 20 rows
```

[18]:
```
orders_df.show()
```

```
+--------------------+--------------------+-----------+-----------------------
+-------------------+---------------------------+---------------------------
+-----------------------------+
|            order_id|
customer_id|order_status|order_purchase_timestamp|  order_approved_at|order_deli
vered_carrier_date|order_delivered_customer_date|order_estimated_delivery_date|
+--------------------+--------------------+-----------+-----------------------
+-------------------+---------------------------+---------------------------
+-----------------------------+
|e481f51cbdc54678b…|9ef432eb625129730…|   delivered|     2017-10-02
10:56:33|2017-10-02 11:07:15|         2017-10-04 19:55:00|          2017-10-10
21:25:13|          2017-10-18 00:00:00|
|53cdb2fc8bc7dce0b…|b0830fb4747a6c6d2…|   delivered|     2018-07-24
20:41:37|2018-07-26 03:24:27|         2018-07-26 14:31:00|          2018-08-07
```

```
15:27:45|          2018-08-13 00:00:00|
|47770eb9100c2d0c4…|41ce2a54c0b03bf34…|   delivered|     2018-08-08
08:38:49|2018-08-08 08:55:23|         2018-08-08 13:50:00|          2018-08-17
18:06:29|          2018-09-04 00:00:00|
|949d5b44dbf5de918…|f88197465ea7920ad…|   delivered|     2017-11-18
19:28:06|2017-11-18 19:45:59|         2017-11-22 13:39:59|          2017-12-02
00:28:42|          2017-12-15 00:00:00|
|ad21c59c0840e6cb8…|8ab97904e6daea886…|   delivered|     2018-02-13
21:18:39|2018-02-13 22:20:29|         2018-02-14 19:46:34|          2018-02-16
18:17:02|          2018-02-26 00:00:00|
|a4591c265e18cb1dc…|503740e9ca751ccdd…|   delivered|     2017-07-09
21:57:05|2017-07-09 22:10:13|         2017-07-11 14:58:04|          2017-07-26
10:57:55|          2017-08-01 00:00:00|
|136cce7faa42fdb2c…|ed0271e0b7da060a3…|    invoiced|     2017-04-11
12:22:08|2017-04-13 13:25:17|                    null|
null|          2017-05-09 00:00:00|
|6514b8ad8028c9f2c…|9bdf08b4b3b52b552…|   delivered|     2017-05-16
13:10:30|2017-05-16 13:22:11|         2017-05-22 10:07:46|          2017-05-26
12:55:51|          2017-06-07 00:00:00|
|76c6e866289321a7c…|f54a9f0e6b351c431…|   delivered|     2017-01-23
18:29:09|2017-01-25 02:50:47|         2017-01-26 14:16:31|          2017-02-02
14:08:10|          2017-03-06 00:00:00|
|e69bfb5eb88e0ed6a…|31ad1d1b63eb99624…|   delivered|     2017-07-29
11:55:02|2017-07-29 12:05:32|         2017-08-10 19:45:24|          2017-08-16
17:14:30|          2017-08-23 00:00:00|
+--------------------+--------------------+------------+-------------------------
+--------------------+----------------------------+----------------------------
+-----------------------------+
only showing top 10 rows
```

[20]:
```
# Order - Order status distribution

orders_df.groupBy('order_status').count().orderBy('count',ascending=False).
  ↪show()
```

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

[ ]:

[ ]:

[ ]:

[21]:
```
# Payments

payments_df.show()
```

```
+--------------------+-----------------+------------+--------------------+-----
--------+
|
order_id|payment_sequential|payment_type|payment_installments|payment_value|
+--------------------+-----------------+------------+--------------------+-----
--------+
|b81ef226f3fe1789b…|                1| credit_card|                   8|
99.33|
|a9810da82917af2d9…|                1| credit_card|                   1|
24.39|
|25e8ea4e93396b6fa…|                1| credit_card|                   1|
65.71|
|ba78997921bbcdc13…|                1| credit_card|                   8|
107.78|
|42fdf880ba16b47b5…|                1| credit_card|                   2|
128.45|
|298fcdf1f73eb413e…|                1| credit_card|                   2|
96.12|
|771ee386b001f0620…|                1| credit_card|                   1|
81.16|
|3d7239c394a212faa…|                1| credit_card|                   3|
51.84|
|1f78449c87a54faf9…|                1| credit_card|                   6|
341.09|
|0573b5e23cbd79800…|                1|      boleto|                   1|
51.95|
|d88e0d5fa41661ce0…|                1| credit_card|                   8|
188.73|
|2480f727e869fdeb3…|                1| credit_card|                   1|
141.9|
|616105c9352a9668c…|                1| credit_card|                   1|
75.78|
|cf95215a722f3ebf2…|                1| credit_card|                   5|
102.66|
|769214176682788a9…|                1| credit_card|                   4|
105.28|
```

```
|12e5cfe0e4716b59a…|                  1| credit_card|                  10|
157.45|
|61059985a6fc0ad64…|                  1| credit_card|                   1|
132.04|
|79da3f5fe31ad1e45…|                  1| credit_card|                   1|
98.94|
|8ac09207f415d55ac…|                  1| credit_card|                   4|
244.15|
|b2349a3f20dfbeef6…|                  1| credit_card|                   3|
136.71|
+--------------------+-----------------+------------+--------------------+-----
--------+
only showing top 20 rows
```

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

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

[ ]:
```

```

[ ]:
```
#Top selling Products
```

[25]:
```
order_item_df.show()
```

```
+--------------------+-------------+--------------------+--------------------+--
-----------------+------+-------------+
|            order_id|order_item_id|          product_id|
seller_id|shipping_limit_date| price|freight_value|
+--------------------+-------------+--------------------+--------------------+--
-----------------+------+-------------+
|00010242fe8c5a6d1…|
1|4244733e06e7ecb49…|48436dade18ac8b2b…|2017-09-19 09:45:35|  58.9|
13.29|
|00018f77f2f0320c5…|
1|e5f2d52b802189ee6…|dd7ddc04e1b6c2c61…|2017-05-03 11:05:13| 239.9|
19.93|
|000229ec398224ef6…|
1|c777355d18b72b67a…|5b51032eddd242adc…|2018-01-18 14:48:30| 199.0|
```

```
17.87|
|00024acbcdf0a6daa…|
1|7634da152a4610f15…|9d7a1d34a50524090…|2018-08-15 10:10:18| 12.99|
12.79|
|00042b26cf59d7ce6…|
1|ac6c3623068f30de0…|df560393f3a51e745…|2017-02-13 13:57:51| 199.9|
18.14|
|00048cc3ae777c65d…|
1|ef92defde845ab845…|6426d21aca402a131…|2017-05-23 03:55:27|  21.9|
12.69|
|00054e8431b9d7675…|
1|8d4f2bb7e93e6710a…|7040e82f899a04d1b…|2017-12-14 12:10:31|  19.9|
11.85|
|000576fe39319847c…|
1|557d850972a7d6f79…|5996cddab893a4652…|2018-07-10 12:30:45| 810.0|
70.75|
|0005a1a1728c9d785…|
1|310ae3c140ff94b03…|a416b6a846a117243…|2018-03-26 18:31:29|145.95|
11.65|
|0005f50442cb953dc…|
1|4535b0e1091c278df…|ba143b05f0110f0dc…|2018-07-06 14:10:56| 53.99|
11.4|
|00061f2a7bc09da83…|
1|d63c1011f49d98b97…|cc419e0650a3c5ba7…|2018-03-29 22:28:09| 59.99|
8.88|
|00063b381e2406b52…|
1|f177554ea93259a5b…|8602a61d680a10a82…|2018-07-31 17:30:39|  45.0|
12.98|
|0006ec9db01a64e59…|
1|99a4788cb24856965…|4a3ca9315b744ce9f…|2018-07-26 17:24:20|  74.0|
23.32|
|0008288aa423d2a3f…|
1|368c6c730842d7801…|1f50f920176fa81da…|2018-02-21 02:55:52|  49.9|
13.37|
|0008288aa423d2a3f…|
2|368c6c730842d7801…|1f50f920176fa81da…|2018-02-21 02:55:52|  49.9|
13.37|
|0009792311464db53…|
1|8cab8abac59158715…|530ec6109d11eaaf8…|2018-08-17 12:15:10|  99.9|
27.65|
|0009c9a17f916a706…|
1|3f27ac8e699df3d30…|fcb5ace8bcc92f757…|2018-05-02 09:31:53| 639.0|
11.34|
|000aed2e25dbad2f9…|
1|4fa33915031a8cde0…|fe2032dab1a61af87…|2018-05-16 20:57:03| 144.0|
8.77|
|000c3e6612759851c…|
1|b50c950aba0dcead2…|218d46b86c1881d02…|2017-08-21 03:33:13|  99.0|
```

```
13.71|
|000e562887b1f2006…|
1|5ed9eaf534f6936b5…|8cbac7e12637ed9cf…|2018-02-28 12:08:37|  25.0|
16.11|
+--------------------+------------+--------------------+--------------------+--
----------------+------+------------+
only showing top 20 rows
```

[27]:
```
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 57:============================>                            (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
```

[ ]:
```

```

[30]:
```
# Average Delivery Time Analysis

delivery_df = orders_df.
 ↪select('order_id','order_purchase_timestamp','order_delivered_customer_date')
```

[31]:
```
delivery_df.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
```

[37]:
```
from pyspark.sql.functions import datediff,to_date

delivery_detail_df = delivery_df.
 ↪withColumn('delivery_time',datediff(col('order_delivered_customer_date'),col('order_purchas


delivery_detail_df.show()
```

```
+--------------------+-----------------------+----------------------------+---
----------+
|
order_id|order_purchase_timestamp|order_delivered_customer_date|delivery_time|
+--------------------+-----------------------+----------------------------+---
----------+
```

```
|e481f51cbdc54678b…|     2017-10-02 10:56:33|          2017-10-10 21:25:13|
8|
|53cdb2fc8bc7dce0b…|     2018-07-24 20:41:37|          2018-08-07 15:27:45|
14|
|47770eb9100c2d0c4…|     2018-08-08 08:38:49|          2018-08-17 18:06:29|
9|
|949d5b44dbf5de918…|     2017-11-18 19:28:06|          2017-12-02 00:28:42|
14|
|ad21c59c0840e6cb8…|     2018-02-13 21:18:39|          2018-02-16 18:17:02|
3|
|a4591c265e18cb1dc…|     2017-07-09 21:57:05|          2017-07-26 10:57:55|
17|
|136cce7faa42fdb2c…|     2017-04-11 12:22:08|                          null|
null|
|6514b8ad8028c9f2c…|     2017-05-16 13:10:30|          2017-05-26 12:55:51|
10|
|76c6e866289321a7c…|     2017-01-23 18:29:09|          2017-02-02 14:08:10|
10|
|e69bfb5eb88e0ed6a…|     2017-07-29 11:55:02|          2017-08-16 17:14:30|
18|
|e6ce16cb79ec1d90b…|     2017-05-16 19:41:10|          2017-05-29 11:18:31|
13|
|34513ce0c4fab462a…|     2017-07-13 19:58:11|          2017-07-19 14:04:48|
6|
|82566a660a982b15f…|     2018-06-07 10:06:19|          2018-06-19 12:05:52|
12|
|5ff96c15d0b717ac6…|     2018-07-25 17:44:10|          2018-07-30 15:52:25|
5|
|432aaf21d85167c2c…|     2018-03-01 14:14:28|          2018-03-12 23:36:26|
11|
|dcb36b511fcac050b…|     2018-06-07 19:03:12|          2018-06-21 15:34:32|
14|
|403b97836b0c04a62…|     2018-01-02 19:00:43|          2018-01-20 01:38:59|
18|
|116f0b09343b49556…|     2017-12-26 23:41:31|          2018-01-08 22:36:36|
13|
|85ce859fd6dc634de…|     2017-11-21 00:03:41|          2017-11-27 18:28:00|
6|
|83018ec114eee8641…|     2017-10-26 15:54:26|          2017-11-08 22:22:00|
13|
+--------------------+------------------------+------------------------------+---
----------+
only showing top 20 rows
```

[38]:
```
delivery_detail_df.orderBy('delivery_time',ascending=False).show()
```

```
+--------------------+------------------------+------------------------------+---
```

```
----------+
|
order_id|order_purchase_timestamp|order_delivered_customer_date|delivery_time|
+--------------------+-----------------------+-----------------------------+---
----------+
|ca07593549f1816d2…|     2017-02-21 23:31:27|          2017-09-19 14:36:39|
210|
|1b3190b2dfa9d789e…|     2018-02-23 14:57:35|          2018-09-19 23:24:07|
208|
|440d0d17af552815d…|     2017-03-07 23:59:51|          2017-09-19 15:12:50|
196|
|285ab9426d6982034…|     2017-03-08 22:47:40|          2017-09-19 14:00:04|
195|
|2fb597c2f772eca01…|     2017-03-08 18:09:02|          2017-09-19 14:33:17|
195|
|0f4519c5f1c541dde…|     2017-03-09 13:26:57|          2017-09-19 14:38:21|
194|
|47b40429ed8cce3ae…|     2018-01-03 09:44:01|          2018-07-13 20:51:31|
191|
|2fe324febf907e3ea…|     2017-03-13 20:17:10|          2017-09-19 17:00:07|
190|
|c27815f7e3dd0b926…|     2017-03-15 23:23:17|          2017-09-19 17:14:25|
188|
|2d7561026d542c8db…|     2017-03-15 11:24:27|          2017-09-19 14:38:18|
188|
|437222e3fd1b07396…|     2017-03-16 11:36:00|          2017-09-19 16:28:58|
187|
|dfe5f68118c257614…|     2017-03-17 12:32:22|          2017-09-19 18:13:19|
186|
|6e82dcfb5eada6283…|     2017-05-17 19:09:02|          2017-11-16 10:56:45|
183|
|2ba1366baecad3c35…|     2017-02-28 14:56:37|          2017-08-28 16:23:46|
181|
|d24e8541128cea179…|     2017-06-12 13:14:11|          2017-12-04 18:36:29|
175|
|3566eabb132f8d647…|     2017-03-29 13:57:55|          2017-09-19 15:07:09|
174|
|ed8e9faf1b75f43ee…|     2017-11-29 15:10:14|          2018-05-21 18:22:18|
173|
|2fa29503f2ebd9f53…|     2017-03-31 15:03:51|          2017-09-19 18:24:46|
172|
|df6d8b7768a047c29…|     2017-04-04 10:46:22|          2017-09-19 15:08:19|
168|
|525e11b26fdb7f414…|     2017-04-04 16:19:10|          2017-09-19 14:58:10|
168|
+--------------------+-----------------------+-----------------------------+---
----------+
only showing top 20 rows
```

[ ]:
```
delivery_detail_df
```