

Retail Business Analytics.

Description

Customers can purchase products or services from Amazon for consumption and usage. Amazon usually sells products and services in-store; however, some may be sold online or over the phone and shipped to the customer. Clothing, medicine, supermarket, and convenience stores are examples of their retail operations.

Objective:

The objective is to analyze the “**retail_db**” dataset, provide reports on the total completed orders, and perform customer and product analytics.

Step 1: Upload the “data-files” file to the HDFS

- 1.1 Download the relevant dataset from the "Course Resources" section or the project description
- 1.2 Upload the dataset to the “FTP” lab from your local system
- 1.3 To move the dataset to “HDFS” from the “Webconsole” use the put command

Commands:

```
hdfs dfs -put data-files /user/swapnasamirshukla1988gmail/retail_ba_dataset
```

Refresh	Download	Cut	Copy	Paste	Rename	Delete	Logout
/							
	Name	Size	Date	Time			
	data-files		11/12/22	09:19			
	insta-cart		15/12/22	09:12			

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$ hdfs dfs -put data-files /user/swapnasamirshukla1988gmail/retail_ba_dataset
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$
```

Step 2: Perform the below tasks on the uploaded dataset using PySpark:

- Login to the Pyspark shell

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$ pyspark3
Python 3.6.8 (default, Nov 16 2020, 16:55:22)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux
Type "help", "copyright", "credits" or "license()" for more information.
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
24/02/04 15:08:35 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
/opt/cloudera/parcels/CDH-7.2.15-1.cdh7.2.15.p1.26792553/lib/spark3/python/pyspark/context.py:238: FutureWarning: Python 3.6 support is deprecated in Spark 3.2.
FutureWarning
Welcome to
      _ _ _ _ _
     / / _ _ _
    / / _ _ _
   / / _ _ _
  / / _ _ _
 / / _ _ _
/_/_/_/_/_
version 3.2.1.7.2.15.1-1

Using Python version 3.6.8 (default, Nov 16 2020 16:55:22)
Spark context Web UI available at http://sl-cdp-prod-en21.cdp-env.gne4-rutx.cloudera.site:4041
Spark context available as 'sc' (master = local[*], app id = local-1707059315807).
SparkSession available as 'spark'.
>>>
```

Task 2.2:

Explore the customer records saved in the "customers-tab-delimited" directory on HDFS

REQUIREMENT:

2.2.1 Show the client information for those who live in California

2.2.2 The final output must be in text format

2.2.3 Save the results in the result/scenario1/solution folder

2.2.4 Only records with the state value "CA" should be included in the result

2.2.5 Only the customer's entire name should be included in the output

Example: "Robert Hudson"

Read the file, assuming it is tab-separated and does not have a header

```
file_path = "retail_ba_dataset/customers-tab-delimited/part-m-00000"
df = spark.read.option("header", "false").option("sep", "\t").csv(file_path)
df.printSchema()
df.show(10)
```

```
>>> file_path = "retail_ba_dataset/customers-tab-delimited/part-m-00000"
>>> df = spark.read.option("header", "false").option("sep", "\t").csv(file_path)
>>> df.show(10)
+---+---+---+---+---+---+---+---+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|
+---+---+---+---+---+---+---+---+
| 1|Richard|Hernandez|XXXXXXXXXX|XXXXXXXXXX|6303 Heather Plaza|Brownsville|TX|78521|
| 2|Mary|Barrett|XXXXXXXXXX|XXXXXXXXXX|9526 Noble Embers...|Littleton|CO|80126|
| 3|Ann|Smith|XXXXXXXXXX|XXXXXXXXXX|3422 Blue Pioneer...|Caguas|PR|00725|
| 4|Mary|Jones|XXXXXXXXXX|XXXXXXXXXX|8324 Little Common|San Marcos|CA|92069|
| 5|Robert|Hudson|XXXXXXXXXX|XXXXXXXXXX|10 Crystal River ...|Caguas|PR|00725|
| 6|Mary|Smith|XXXXXXXXXX|XXXXXXXXXX|3151 Sleepy Quail...|Passaic|NJ|07055|
| 7|Melissa|Wilcox|XXXXXXXXXX|XXXXXXXXXX|9453 High Concession|Caguas|PR|00725|
| 8|Megan|Smith|XXXXXXXXXX|XXXXXXXXXX|3047 Foggy Forest...|Lawrence|MA|01841|
| 9|Mary|Perez|XXXXXXXXXX|XXXXXXXXXX|3616 Quaking Street|Caguas|PR|00725|
|10|Melissa|Smith|XXXXXXXXXX|XXXXXXXXXX|8598 Harvest Beac...|Stafford|VA|22554|
+---+---+---+---+---+---+---+---+
only showing top 10 rows
```

```
from pyspark.sql.functions import col
# Show people residing in California
df_filtered = df.filter(col("_c6") == "CA")
# Show the filtered DataFrame
df_filtered.show()
```

```
>>> from pyspark.sql.functions import col
>>> df_filtered = df.filter(col("_c6") == "CA")
>>> df_filtered.show()
+---+---+---+---+---+---+---+---+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|
+---+---+---+---+---+---+---+---+
| 4|Mary|Jones|XXXXXXXXXX|XXXXXXXXXX|8324 Little Common|San Marcos|CA|92069|
|14|Katherine|Smith|XXXXXXXXXX|XXXXXXXXXX|5666 Hazy Pony Sq...|Pico Rivera|CA|90660|
|15|Jane|Luna|XXXXXXXXXX|XXXXXXXXXX|673 Burning Glen|Fontana|CA|92336|
|38|Robert|Smith|XXXXXXXXXX|XXXXXXXXXX|2734 Hazy Butterf...|Martinez|CA|94553|
|35|Margaret|Wright|XXXXXXXXXX|XXXXXXXXXX|9456 Sleepy Jetty|Oceanside|CA|92056|
|40|Mary|Smith|XXXXXXXXXX|XXXXXXXXXX|7358 Rocky Villas|Long Beach|CA|90805|
|44|Howard|Smith|XXXXXXXXXX|XXXXXXXXXX|1356 Easy Plaza|Napá|CA|94558|
|50|Mary|Kim|XXXXXXXXXX|XXXXXXXXXX|938 Rustic Pine R...|San Bernardino|CA|92410|
|59|Douglas|James|XXXXXXXXXX|XXXXXXXXXX|2386 Green Lane|Sunnyvale|CA|94086|
|70|Mary|Simmons|XXXXXXXXXX|XXXXXXXXXX|5553 Cinder Harbour|Los Angeles|CA|90042|
|72|Frank|Gillespie|XXXXXXXXXX|XXXXXXXXXX|5332 Heather Hill...|Vista|CA|92084|
|76|Joseph|Young|XXXXXXXXXX|XXXXXXXXXX|7605 Tawny Horse ...|Los Angeles|CA|90016|
|89|Sean|Smith|XXXXXXXXXX|XXXXXXXXXX|9126 Wishing Expr...|Escondido|CA|92027|
|106|Lauren|Freeman|XXXXXXXXXX|XXXXXXXXXX|2783 Foggy Mews|Napá|CA|94558|
|114|Alice|Hanner|XXXXXXXXXX|XXXXXXXXXX|4566 Burning Deer...|Bellflower|CA|90706|
|115|Mary|Smith|XXXXXXXXXX|XXXXXXXXXX|1613 Broad Beach|West Covina|CA|91790|
|125|Mary|Gallagher|XXXXXXXXXX|XXXXXXXXXX|9831 Sunny Cloud ...|Oxnard|CA|93030|
|139|Daniel|Maxwell|XXXXXXXXXX|XXXXXXXXXX|9468 Red Corner|San Diego|CA|92104|
|149|Shirley|McLain|XXXXXXXXXX|XXXXXXXXXX|9729 Emerald Pony...|Colton|CA|92324|
|156|Mary|Smith|XXXXXXXXXX|XXXXXXXXXX|2291 Thunder Leaf...|Los Angeles|CA|90001|
+---+---+---+---+---+---+---+---+
only showing top 20 rows
```

```

from pyspark.sql.functions import concat_ws, col

# Concat First and second names to get full name
full_name = concat_ws(" ", col("_c1"), col("_c2"))

# Filter for clients in California (assuming _c7 is the state column)
ca_clients = df.filter(col("_c7") == "CA").select(full_name.alias("Full Name"))

# Show the result
ca_clients.show()

```

```

>>> from pyspark.sql.functions import concat_ws, col
>>>
>>> full_name = concat_ws(" ", col("_c1"), col("_c2"))
>>> ca_clients = df.filter(col("_c7") == "CA").select(full_name.alias("Full Name"))
>>> ca_clients.show()
+-----+
|      Full Name|
+-----+
|      Mary Jones|
|Katherine Smith|
|      Jane Luna|
|      Robert Smith|
|Margaret Wright|
|      Mary Smith|
|      Howard Smith|
|      Mary Kim|
|      Douglas James|
|      Mary Simmons|
|Frank Gillespie|
|      Joseph Young|
|      Sean Smith|
|      Lauren Freeman|
|      Alice Warner|
|      Mary Smith|
|      Mary Gallagher|
|      Daniel Maxwell|
|Shirley McClain|
|      Mary Smith|
+-----+
only showing top 20 rows

```

```

# Specify the path where you want to save the results
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution"

# Save the DataFrame as a text file
# coalesce(1) is used to save the output into a single file.
ca_clients.coalesce(1).write.mode("overwrite").text(output_path)

```

```

>>> output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution"
>>> ca_clients.coalesce(1).write.mode("overwrite").text(output_path)
>>>

```

Check the hdfs location to see the file

hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution

```
sl-cdp-prod-en10 login: swapnasamirshukla1988gmail
Password:
Last login: Sun Feb  4 15:31:03 on pts/43

[CloudEra]
=====
*
:

Password for swapnasamirshukla1988gmail@CDP-ENV.GNE4-RUTX.CLOUDERA.SITE:
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1
Found 1 items
drwxr-xr-x - swapnasamirshukla1988gmail swapnasamirshukla1988gmail      0 2024-02-04 15:37 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail      0 2024-02-04 15:37 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 26096 2024-02-04 15:37 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/part-00000-fbf80159-9c66-4aa4-9c6f-879289198745-c000.txt
```

The file is stored as part-00000-fbf80159-9c66-4aa4-9c6f-879289198745-c000.txt

Rename file using mv command

hdfs dfs -mv /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/part-00000-fbf80159-9c66-4aa4-9c6f-879289198745-c000.txt
/user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/retai_sol1.txt

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -mv /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/part-00000-fbf80159-9c66-4aa4-9c6f-879289198745-c000.tx
t /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/retai_sol1.txt
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail      0 2024-02-04 15:37 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 26096 2024-02-04 15:37 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario1/solution/retai_sol1.txt
```

Task 2.3:

Explore the order records saved in the “orders parquet” directory on HDFS

REQUIREMENT:

- 2.3.1 Show all orders with the order status value "COMPLETE"
- 2.3.2 The output should be in JSON format
- 2.3.3 Save the data in the "result/scenario2/solution" directory on HDFS
- 2.3.4 The "order date" column should be in the "YYYY-MM-DD" format
- 2.3.5 Use GZIP compression to compress the output
- 2.3.6 Only the column names listed below should be included in the output:
 - 2.3.6.1 Order number
 - 2.3.6.2 Order date
 - 2.3.6.3 Current situation

```
file_path = "retail_ba_dataset/orders_parquet/741ca897-c70e-4633-b352-5dc3414c5680.parquet"
df = spark.read.parquet (file_path)
df.printSchema()
df.show(10)
```

```
>>> file_path = "retail_ba_dataset/orders_parquet/741ca897-c70e-4633-b352-5dc3414c5680.parquet"
>>> df = spark.read.parquet (file_path)
24/02/04 17:13:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/02/04 17:13:38 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
>>> df.printSchema()
root
 |-- order_id: integer (nullable = true)
 |-- order_date: long (nullable = true)
 |-- order_customer_id: integer (nullable = true)
 |-- order_status: string (nullable = true)
>>> df.show(10)
+-----+-----+-----+-----+
|order_id|order_date|order_customer_id|order_status|
+-----+-----+-----+-----+
|1|1374710400000|11599|CLOSED|
|2|1374710400000|256|PENDING_PAYMENT|
|3|1374710400000|12111|COMPLETE|
|4|1374710400000|8827|CLOSED|
|5|1374710400000|11318|COMPLETE|
|6|1374710400000|7130|COMPLETE|
|7|1374710400000|4530|COMPLETE|
|8|1374710400000|2911|PROCESSING|
|9|1374710400000|5657|PENDING_PAYMENT|
|10|1374710400000|5648|PENDING_PAYMENT|
+-----+-----+-----+-----+
only showing top 10 rows
```

```
# Filter for orders with the status "COMPLETE"
df_filtered = df.filter(df["order_status"] == "COMPLETE")
```

```
>>> df_filtered = df.filter(df["order_status"] == "COMPLETE")
>>> df_filtered.show(10)
+-----+-----+-----+-----+
|order_id|order_date|order_customer_id|order_status|
+-----+-----+-----+-----+
|3|1374710400000|12111|COMPLETE|
|5|1374710400000|11318|COMPLETE|
|6|1374710400000|7130|COMPLETE|
|7|1374710400000|4530|COMPLETE|
|15|1374710400000|2568|COMPLETE|
|17|1374710400000|2667|COMPLETE|
|22|1374710400000|333|COMPLETE|
|26|1374710400000|7562|COMPLETE|
|28|1374710400000|656|COMPLETE|
|32|1374710400000|3960|COMPLETE|
+-----+-----+-----+-----+
only showing top 10 rows
```

```
from pyspark.sql.functions import from_unixtime, col
```

```
# Convert order_date from milliseconds to "YYYY-MM-DD" format
```

```
df_with_converted_date = df_filtered.withColumn("order_date", from_unixtime(col("order_date") / 1000, "yyyy-MM-dd"))
```

```
>>> from pyspark.sql.functions import from_unixtime, col
>>> df_with_converted_date = df_filtered.withColumn("order_date", from_unixtime(col("order_date") / 1000, "yyyy-MM-dd"))
>>> df_with_converted_date.show(10)
+-----+-----+-----+-----+
|order_id|order_date|order_customer_id|order_status|
+-----+-----+-----+-----+
|3|2013-07-25|12111|COMPLETE|
|5|2013-07-25|11318|COMPLETE|
|6|2013-07-25|7130|COMPLETE|
|7|2013-07-25|4530|COMPLETE|
|15|2013-07-25|2568|COMPLETE|
|17|2013-07-25|2667|COMPLETE|
|22|2013-07-25|333|COMPLETE|
|26|2013-07-25|7562|COMPLETE|
|28|2013-07-25|656|COMPLETE|
|32|2013-07-25|3960|COMPLETE|
+-----+-----+-----+-----+
only showing top 10 rows
```

```
# Selecting only the required columns
```

```
required_columns = df_with_converted_date.select(
    col("order_id").alias("Order number"),
    col("order_date").alias("Order date"),
    col("order_status").alias("Current situation")
)
```

```
>>> required_columns = df_with_converted_date.select(
...     col("order_id").alias("Order number"),
...     col("order_date").alias("Order date"),
...     col("order_status").alias("Current situation")
... )
>>>
>>> required_columns.show(10)
+-----+-----+-----+-----+
|Order number|Order date|Current situation|
+-----+-----+-----+-----+
|3|2013-07-25|COMPLETE|
|5|2013-07-25|COMPLETE|
|6|2013-07-25|COMPLETE|
|7|2013-07-25|COMPLETE|
|15|2013-07-25|COMPLETE|
|17|2013-07-25|COMPLETE|
|22|2013-07-25|COMPLETE|
|26|2013-07-25|COMPLETE|
|28|2013-07-25|COMPLETE|
|32|2013-07-25|COMPLETE|
+-----+-----+-----+-----+
only showing top 10 rows
```

Specify the path where you want to save the results

```
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution"
```

Save the data in JSON format with GZIP compression

```
required_columns.write.mode("overwrite").option("compression", "gzip").json(output_path)
```

```
>> output_path = " /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution"
>> required_columns.write.mode("overwrite").option("compression", "gzip").json(output_path)
>> █
```

Check the hdfs location to see the file

```
hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution
```

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 17:59 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 79705 2024-02-04 17:59 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/part-00000-785622e8-80b0-4ac6-9fe7-67165f99b7de-c000.json.gz
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -mv /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/part-00000-785622e8-80b0-4ac6-9fe7-67165f99b7de-c000.json.gz /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/retai_sol2.json.gz
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 17:59 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 79705 2024-02-04 17:59 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/retai_sol2.json.gz
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ █
```

The file is stored as part-00000-785622e8-80b0-4ac6-9fe7-67165f99b7de-c000.json.gz

Rename file using mv command

```
hdfs dfs -mv /user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/part-00000-785622e8-80b0-4ac6-9fe7-67165f99b7de-c000.json.gz
/user/swapnasamirshukla1988gmail/retail_ba/result/scenario2/solution/retai_sol2.json.gz
```

Task 2.4

Explore the customer records saved in the "customers-tab-delimited" directory on HDFS

REQUIREMENT:

- 2.4.1 Produce a list of all consumers who live in the city of "Caguas"
- 2.4.2 Save the data in the result/scenario3/solution directory on HDFS
- 2.4.3 The result should only contain records with the value "Caguas" for the customer city
- 2.4.4 Use snappy compression to compress the output
- 2.4.5 Save the file in the orc format

```
# Read the file, assuming it is tab-separated and does not have a header
```

```
file_path = "retail_ba_dataset/customers-tab-delimited/part-m-00000"
df = spark.read.option("header", "false").option("sep", "\t").csv(file_path)
df.printSchema()
df.show(10)
```

```
>>> file_path = "retail_ba_dataset/customers-tab-delimited/part-m-00000"
>>> df = spark.read.option("header", "false").option("sep", "\t").csv(file_path)
>>> df.printSchema()
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)
 |-- _c3: string (nullable = true)
 |-- _c4: string (nullable = true)
 |-- _c5: string (nullable = true)
 |-- _c6: string (nullable = true)
 |-- _c7: string (nullable = true)
 |-- _c8: string (nullable = true)

>>> df.show(10)
+-----+-----+-----+-----+-----+-----+-----+-----+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|
+-----+-----+-----+-----+-----+-----+-----+-----+
| 1|Richard|Hernandez|XXXXXXXXXX|XXXXXXXXXX|6303 Heather Plaza|Brownsville|TX|78521|
| 2| Mary|Barrett|XXXXXXXXXX|XXXXXXXXXX|9526 Noble Embers...|Littleton|CO|80126|
| 3| Ann|Smith|XXXXXXXXXX|XXXXXXXXXX|3422 Blue Pioneer...|Caguas|PR|00725|
| 4| Mary|Jones|XXXXXXXXXX|XXXXXXXXXX|8324 Little Common|San Marcos|CA|92069|
| 5| Robert|Hudson|XXXXXXXXXX|XXXXXXXXXX|10 Crystal River ...|Caguas|PR|00725|
| 6| Mary|Smith|XXXXXXXXXX|XXXXXXXXXX|3151 Sleepy Quail...|Passaic|NJ|07055|
| 7|Melissa|Wilcox|XXXXXXXXXX|XXXXXXXXXX|9453 High Concession|Caguas|PR|00725|
| 8| Megan|Smith|XXXXXXXXXX|XXXXXXXXXX|3047 Foggy Forest...|Lawrence|MA|01841|
| 9| Mary|Perez|XXXXXXXXXX|XXXXXXXXXX|3616 Quaking Street|Caguas|PR|00725|
|10|Melissa|Smith|XXXXXXXXXX|XXXXXXXXXX|8598 Harvest Beac...|Stafford|VA|22554|
+-----+-----+-----+-----+-----+-----+-----+-----+
only showing top 10 rows
```



```
# Filter for customers who live in "Caguas"
customers_in_caguas = df.filter(col("_c6") == "Caguas")
```

```
>>> customers_in_caguas = df.filter(col("_c6") == "Caguas")
>>> customers_in_caguas.show(5)
+---+-----+-----+-----+-----+-----+-----+-----+
|_c0|_c1|_c2|_c3|_c4|_c5|_c6|_c7|_c8|
+---+-----+-----+-----+-----+-----+-----+-----+
| 3|Ann|Smith|XXXXXXXXXX|XXXXXXXXXX|3422 Blue Pioneer...|Caguas|PR|00725|
| 5|Robert|Hudson|XXXXXXXXXX|XXXXXXXXXX|10 Crystal River ...|Caguas|PR|00725|
| 7|Melissa|Wilcox|XXXXXXXXXX|XXXXXXXXXX|9453 High Concession|Caguas|PR|00725|
| 9|Mary|Perez|XXXXXXXXXX|XXXXXXXXXX|3616 Quaking Street|Caguas|PR|00725|
|11|Mary|Huffman|XXXXXXXXXX|XXXXXXXXXX|3169 Stony Woods|Caguas|PR|00725|
+---+-----+-----+-----+-----+-----+-----+-----+
only showing top 5 rows
```

```
# Save the filtered data in ORC format with Snappy compression to the specified HDFS directory
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario3/solution"
customers_in_caguas.write.mode("overwrite").option("compression", "snappy").orc(output_path)
```

```
>>> output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario3/solution"
>>> customers_in_caguas.write.mode("overwrite").option("compression", "snappy").orc(output_path)
>>> []
```

```
# Check the hdfs location to see the file
```

```
hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario3/solution
```

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario3/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 18:19 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario3/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 86324 2024-02-04 18:19 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario3/solution/part-00000-e90defb9-6794-45c4-ae02-36e1be7f9286-c000.snappy.orc
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$
```

Task 2.5

Explore all the category records stored in the “categories” directory on HDFS

REQUIREMENT:

- 2.5.1 Save the result files in CSV format
- 2.5.2 Save the data in the result/scenario4/solution directory on HDFS
- 2.5.3 Use lz4 compression to compress the output

```
file_path = "retail_ba_dataset/categories/ part-m-00000"
df = spark.read.option("header", "false").option("sep", ",").csv(file_path)
df.printSchema()
df.show(10)
```

```
>>> file_path = "retail_ba_dataset/categories/part-m-00000"
>>> df = spark.read.option("header", "false").option("sep", ",").csv(file_path)
>>> df.printSchema()
root
 |-- _c0: string (nullable = true)
 |-- _c1: string (nullable = true)
 |-- _c2: string (nullable = true)

>>> df.show(10)
+---+---+-----+
|_c0|_c1|          _c2|
+---+---+-----+
| 1| 2|      Football|
| 2| 2|        Soccer|
| 3| 2|Baseball & Softball|
| 4| 2|      Basketball|
| 5| 2|        Lacrosse|
| 6| 2|Tennis & Racquet|
| 7| 2|        Hockey|
| 8| 2|    More Sports|
| 9| 3|Cardio Equipment|
|10| 3|Strength Training|
+---+---+-----+
only showing top 10 rows
```

```
# Save the filtered data in text format with lz4 compression to the specified HDFS directory
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario4/solution"
df.write.mode("overwrite").option("compression", "gzip").csv(output_path)
```

```
>>> output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario4/solution"
>>> df.write.mode("overwrite").option("compression", "gzip").csv(output_path)
>>> 
```

*Spark's built-in CSV data source does not directly support LZ4 compression. One might need to save the data uncompressed and then compress it using external tools, or choose a compression codec supported by Spark for CSV, such as **bzip2**, **gzip** or **deflate**.*

```
# Check the hdfs location to see the file
```

```
hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario4/solution
```

```
ssword for swapnasamirshukla1988gmail@CDP-ENV.GNE4-RUTX.CLOUDERA.SITE:
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario4/solution
round 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 18:42 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario4/solution/ SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 575 2024-02-04 18:42 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario4/solution/part-00000-bfbb1b16-f621-4922-b6e4-578ebb3834a9-c000.csv.gz
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$ 
```

Task 2.6

Explore all product records that are saved in the “products_avro” database

REQUIREMENT:

- 2.6.1 Only products with a price of more than 1000.0 should be included in the output
- 2.6.2 Save the output files in parquet format
- 2.6.3 Remove data from the table if the product price is lesser than 1000.0
- 2.6.4 Save the data in the result/scenario5/solution directory on HDFS
- 2.6.5 Use snappy compression to compress the output

Path to the Avro files

```
file_paths = [  
    "retail_ba_dataset/products_avro/part-m-00000.avro",  
    "retail_ba_dataset/products_avro/part-m-00001.avro",  
    "retail_ba_dataset/products_avro/part-m-00002.avro",  
    "retail_ba_dataset/products_avro/part-m-00003.avro"  
]
```

Read the Avro files into a DataFrame

```
df = spark.read.format("avro").load(file_paths)
```

Remove completely duplicated rows

```
df_deduplicated = df.dropDuplicates()
```

```
>>> file_paths = [  
...     "retail_ba_dataset/products_avro/part-m-00000.avro",  
...     "retail_ba_dataset/products_avro/part-m-00001.avro",  
...     "retail_ba_dataset/products_avro/part-m-00002.avro",  
...     "retail_ba_dataset/products_avro/part-m-00003.avro"  
... ]  
>>> df = spark.read.format("avro").load(file_paths)  
>>> df_deduplicated = df.dropDuplicates()  
>>> df_deduplicated.printSchema()  
root  
 |-- product_id: integer (nullable = true)  
 |-- product_category_id: integer (nullable = true)  
 |-- product_name: string (nullable = true)  
 |-- product_description: string (nullable = true)  
 |-- product_price: float (nullable = true)  
 |-- product_image: string (nullable = true)  
  
>>> df_deduplicated.show(5)  
+-----+-----+-----+-----+-----+-----+  
|product_id|product_category_id|product_name|product_description|product_price|product_image|  
+-----+-----+-----+-----+-----+-----+  
|1033|46|YETI Tundra 45 Ch...| |349.99|http://images.acm...|  
|1034|47|Nike+ Fuelband SE| |99.0|http://images.acm...|  
|1042|47|Under Armour Hust...| |54.99|http://images.acm...|  
|1126|50|Nike Men's New Yo...| |34.0|http://images.acm...|  
|1279|57|PUMA Men's evoPOW...| |189.99|http://images.acm...|  
+-----+-----+-----+-----+-----+-----+  
only showing top 5 rows
```

Filter for products with price more than 1000.0

```
filtered_df = df_deduplicated.filter(col("product_price") >= 1000.0)
```

```
>>> filtered_df = df_deduplicated.filter(col("product_price") >= 1000.0)  
>>> filtered_df.show()  
+-----+-----+-----+-----+-----+-----+  
|product_id|product_category_id|product_name|product_description|product_price|product_image|  
+-----+-----+-----+-----+-----+-----+  
|1048|47|Spalding Beast 60...| |1099.99|http://images.acm...|  
|496|22|SOLE F85 Treadmill| |1799.99|http://images.acm...|  
|199|10|SOLE F85 Treadmill| |1799.99|http://images.acm...|  
|66|4|SOLE F85 Treadmill| |1799.99|http://images.acm...|  
|208|10|SOLE E35 Elliptical| |1999.99|http://images.acm...|  
+-----+-----+-----+-----+-----+-----+
```

```
# Save the filtered data in parquet format with snappy compression to the specified HDFS directory
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario5/solution"
filtered_df.write.mode("overwrite").option("compression", "snappy").parquet(output_path)
```

```
>> output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario5/solution"
>> filtered_df.write.mode("overwrite").option("compression", "snappy").parquet(output_path)
>> []
```

Check the hdfs location to see the file

```
hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario5/solution
```

```
Password for swapnasamirshukla1988gmail@CDP-ENV-GNE4-RUTX.CLOUDERA.SITE:
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario5/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 19:16 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario5/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 2412 2024-02-04 19:16 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario5/solution/part-00000-725426c9-
05e-4939-9496-d30227e0ab94-c000.snappy.parquet
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$
```

Task 2.7

Explore the "products_avro" stored in product records

REQUIREMENT:

- 2.7.1 Only products with a price of more than 1000.0 should be in the output
- 2.7.2 The pattern "Treadmill" appears in the product name
- 2.7.3 Save the output files in parquet format
- 2.7.4 Save the data in the result/scenario6/solution directory on HDFS
- 2.7.5 Use GZIP compression to compress the output

Path to the Avro files

```
file_paths = [  
    "retail_ba_dataset/products_avro/part-m-00000.avro",  
    "retail_ba_dataset/products_avro/part-m-00001.avro",  
    "retail_ba_dataset/products_avro/part-m-00002.avro",  
    "retail_ba_dataset/products_avro/part-m-00003.avro"  
]
```

Read the Avro files into a DataFrame

```
df = spark.read.format("avro").load(file_paths)
```

Remove completely duplicated rows

```
df_deduplicated = df.dropDuplicates()
```

```
>>> file_paths = [  
...     "retail_ba_dataset/products_avro/part-m-00000.avro",  
...     "retail_ba_dataset/products_avro/part-m-00001.avro",  
...     "retail_ba_dataset/products_avro/part-m-00002.avro",  
...     "retail_ba_dataset/products_avro/part-m-00003.avro"  
... ]  
>>> df = spark.read.format("avro").load(file_paths)  
>>> df_deduplicated = df.dropDuplicates()  
>>> df_deduplicated.printSchema()  
root  
 |-- product_id: integer (nullable = true)  
 |-- product_category_id: integer (nullable = true)  
 |-- product_name: string (nullable = true)  
 |-- product_description: string (nullable = true)  
 |-- product_price: float (nullable = true)  
 |-- product_image: string (nullable = true)  
  
>>> df_deduplicated.show(5)  
+-----+-----+-----+-----+-----+-----+  
|product_id|product_category_id|product_name|product_description|product_price|product_image|  
+-----+-----+-----+-----+-----+-----+  
|1033|46|YETI Tundra 45 Ch...|349.99|http://images.acm...|  
|1034|47|Nike+ Fuelband SE|99.0|http://images.acm...|  
|1042|47|Under Armour Hust...|54.99|http://images.acm...|  
|1126|50|Nike Men's New Yo...|34.0|http://images.acm...|  
|1279|57|PUMA Men's evoPOW...|189.99|http://images.acm...|  
+-----+-----+-----+-----+-----+-----+  
only showing top 5 rows
```

Filter for products with price more than 1000.0

```
filtered_df = df_deduplicated.filter(col("product_price") >= 1000.0)
```

```
>>> filtered_df = df_deduplicated.filter(col("product_price") >= 1000.0)  
>>> filtered_df.show()  
+-----+-----+-----+-----+-----+-----+  
|product_id|product_category_id|product_name|product_description|product_price|product_image|  
+-----+-----+-----+-----+-----+-----+  
|1048|47|Spalding Beast 60...|1099.99|http://images.acm...|  
|496|22|SOLE F85 Treadmill|1799.99|http://images.acm...|  
|199|10|SOLE F85 Treadmill|1799.99|http://images.acm...|  
|66|4|SOLE F85 Treadmill|1799.99|http://images.acm...|  
|208|10|SOLE E35 Elliptical|1999.99|http://images.acm...|  
+-----+-----+-----+-----+-----+-----+
```

Now, filter for product names containing 'Treadmill'

```
treadmill_df = filtered_df.filter(col("product_name").contains("Treadmill"))
```

```
>>> treadmill_df = filtered_df.filter(col("product_name").contains("Treadmill"))
```

```
>>> treadmill_df.show()
```

product_id	product_category_id	product_name	product_description	product_price	product_image
496	22	SOLE F85 Treadmill		1799.99	http://images.acm...
199	10	SOLE F85 Treadmill		1799.99	http://images.acm...
66	4	SOLE F85 Treadmill		1799.99	http://images.acm...

Save the filtered data in parquet format with gzip compression to the specified HDFS directory

```
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario6/solution"
```

```
filtered_df.write.mode("overwrite").option("compression", "gzip").parquet(output_path)
```

```
>>> output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario6/solution"
```

```
>>> filtered_df.write.mode("overwrite").option("compression", "gzip").parquet(output_path)
```

```
>>>
```

Check the hdfs location to see the file

```
hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario6/solution
```

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario6/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 19:26 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario6/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 2556 2024-02-04 19:26 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario6/solution/part-00000-070cfaf6-833f-4ce9-8056-8402ca31a407-c000.gz.parquet
[swapnasamirshukla1988gmail@sl-cdp-prod-en10 ~]$
```

Task 2.8

Explore the order records that are saved in the "orders parquet" table on HDFS

REQUIREMENT:

- 2.8.1 Output all PENDING orders in July 2013
- 2.8.2 Output files should be in JSON format
- 2.8.3 Save the data in the result/scenario7/solution directory on HDFS.
- 2.8.4 Only entries with the order status value of "PENDING" should be included in the result
- 2.8.5 Order date should be in the YYYY-MM-DD format
- 2.8.6 Use snappy compression to compress the output, which should just contain the order date and order status

```
file_path = "retail_ba_dataset/orders_parquet/741ca897-c70e-4633-b352-5dc3414c5680.parquet"
df = spark.read.parquet (file_path)
df.printSchema()
df.show(10)
```

```
>>> file_path = "retail_ba_dataset/orders_parquet/741ca897-c70e-4633-b352-5dc3414c5680.parquet"
>>> df = spark.read.parquet (file_path)
24/02/04 17:13:38 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
24/02/04 17:13:38 WARN shortcircuit.DomainSocketFactory: The short-circuit local reads feature cannot be used because libhadoop cannot be loaded.
>>> df.printSchema()
root
 |-- order_id: integer (nullable = true)
 |-- order_date: long (nullable = true)
 |-- order_customer_id: integer (nullable = true)
 |-- order_status: string (nullable = true)
>>> df.show(10)
+-----+-----+-----+-----+
|order_id|order_date|order_customer_id|order_status|
+-----+-----+-----+-----+
|1|1374710400000|11599|CLOSED|
|2|1374710400000|256|PENDING_PAYMENT|
|3|1374710400000|12111|COMPLETE|
|4|1374710400000|8827|CLOSED|
|5|1374710400000|11318|COMPLETE|
|6|1374710400000|7130|COMPLETE|
|7|1374710400000|4530|COMPLETE|
|8|1374710400000|2911|PROCESSING|
|9|1374710400000|5657|PENDING_PAYMENT|
|10|1374710400000|5648|PENDING_PAYMENT|
+-----+-----+-----+-----+
only showing top 10 rows
```

```
from pyspark.sql.functions import from_unixtime, col
```

```
# Convert order_date from milliseconds to "YYYY-MM-DD" format
```

```
df_with_converted_date = df.withColumn("order_date", from_unixtime(col("order_date") / 1000, "yyy-MM-dd"))
```

```
>>> df_with_converted_date = df.withColumn("order_date", from_unixtime(col("order_date") / 1000, "yyy-MM-dd"))
>>> df_with_converted_date.show()
+-----+-----+-----+-----+
|order_id|order_date|order_customer_id|order_status|
+-----+-----+-----+-----+
|1|2013-07-25|11599|CLOSED|
|2|2013-07-25|256|PENDING_PAYMENT|
|3|2013-07-25|12111|COMPLETE|
|4|2013-07-25|8827|CLOSED|
|5|2013-07-25|11318|COMPLETE|
|6|2013-07-25|7130|COMPLETE|
|7|2013-07-25|4530|COMPLETE|
|8|2013-07-25|2911|PROCESSING|
|9|2013-07-25|5657|PENDING_PAYMENT|
|10|2013-07-25|5648|PENDING_PAYMENT|
|11|2013-07-25|918|PAYMENT_REVIEW|
|12|2013-07-25|1837|CLOSED|
|13|2013-07-25|9149|PENDING_PAYMENT|
|14|2013-07-25|9842|PROCESSING|
|15|2013-07-25|2568|COMPLETE|
|16|2013-07-25|7276|PENDING_PAYMENT|
|17|2013-07-25|2667|COMPLETE|
|18|2013-07-25|1205|CLOSED|
|19|2013-07-25|9488|PENDING_PAYMENT|
|20|2013-07-25|9198|PROCESSING|
+-----+-----+-----+-----+
only showing top 20 rows
```

```
pending_orders_july_2013 = df_with_converted_date \
    .filter(col("order_status") == "PENDING") \
    .filter(col("order_date").between("2013-07-01", "2013-07-31"))
```

```
... pending_orders_july_2013 = df_with_converted_date \
...     .filter(col("order_status") == "PENDING") \
...     .filter(col("order_date").between("2013-07-01", "2013-07-31"))
>>> pending_orders_july_2013.show(5)
```

order_id	order_date	order_customer_id	order_status
21	2013-07-25	2711	PENDING
36	2013-07-25	5649	PENDING
39	2013-07-25	8214	PENDING
42	2013-07-25	9776	PENDING
44	2013-07-25	10500	PENDING

only showing top 5 rows

```
pending_orders = df_with_converted_date \
    .filter(col("order_status") == "PENDING")
```

```
>>> pending_orders = df_with_converted_date \
...     .filter(col("order_status") == "PENDING")
>>>
>>> pending_orders.show()
```

order_id	order_date	order_customer_id	order_status
21	2013-07-25	2711	PENDING
36	2013-07-25	5649	PENDING
39	2013-07-25	8214	PENDING
42	2013-07-25	9776	PENDING
44	2013-07-25	10500	PENDING
49	2013-07-25	1871	PENDING
55	2013-07-25	2052	PENDING
68	2013-07-25	4320	PENDING
85	2013-07-25	1485	PENDING
96	2013-07-25	8683	PENDING
97	2013-07-25	10784	PENDING
121	2013-07-26	2074	PENDING
132	2013-07-26	289	PENDING
158	2013-07-26	12345	PENDING
167	2013-07-26	1347	PENDING
181	2013-07-26	7473	PENDING
188	2013-07-26	2889	PENDING
189	2013-07-26	10177	PENDING
190	2013-07-26	11115	PENDING
206	2013-07-26	8994	PENDING

only showing top 20 rows


```
pending_orders = df_with_converted_date \
    .filter(col("order_status") == "PENDING") \
    .select("order_date", "order_status")
```

```
>>> pending_orders = df_with_converted_date \
...     .filter(col("order_status") == "PENDING") \
...     .select("order_date", "order_status")
>>> pending_orders.show()
+-----+-----+
|order_date|order_status|
+-----+-----+
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-25|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
|2013-07-26|    PENDING|
+-----+-----+
only showing top 20 rows
```

Save the filtered data in json format with snappy compression to the specified HDFS directory

```
output_path = "/user/swapnasamirshukla1988gmail/retail_ba/result/scenario7/solution"
```

```
pending_orders.write.mode("overwrite").option("compression", "gzip").json(output_path)
```

it's important to note that JSON itself doesn't support Snappy compression directly within the Spark DataFrame API. Snappy compression is typically used with columnar storage formats like Parquet or ORC, which are better suited for compressing tabular data.

Check the hdfs location to see the file

```
hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario7/solution
```

```
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$
[swapnasamirshukla1988gmail@sl-cdp-prod-en21 ~]$ hdfs dfs -ls /user/swapnasamirshukla1988gmail/retail_ba/result/scenario7/solution
Found 2 items
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 0 2024-02-04 19:56 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario7/solution/_SUCCESS
-rw-rw-r-- 2 swapnasamirshukla1988gmail swapnasamirshukla1988gmail 4417 2024-02-04 19:56 /user/swapnasamirshukla1988gmail/retail_ba/result/scenario7/solution/part-00000-63d0f1a7-f
dcc-4738-846d-ae4d161199a4-c000.json.gz
```