By

Name: Mukul Aggarwal

Email: msd23007@iiitl.ac.in (mailto:msd23007@iiitl.ac.in)

Enrolment Number: MSD23007

Extraction

Connecting to SQL Server on GCP

4

```
jdbcHostname = "34.72.112.128"
jdbcPort = 3306
jdbcDatabase = "grocery_data"
jdbcUsername = "Mukul"
jdbcPassword = "Aesl$775"
```

5

```
jdbcUrl = f"jdbc:mysql://{jdbcHostname}:{jdbcPort}/{jdbcDatabase}?user={jdbcUsername}&password={jdbcPassword}"
```

6

```
connectionProperties = {
    "user" : jdbcUsername,
    "password" : jdbcPassword,
    "driver" : "com.mysql.jdbc.Driver"
}

df = spark.read.jdbc(jdbcUrl, "Transactions", properties=connectionProperties)
# Loaded Transactions table from MySQL
df.show()
```

```
|         2300|2015-09-19|          pip fruit\r|
|         1187|2015-12-12|  other vegetables\r|
|         3037|2015-02-01|         whole milk\r|
|         4941|2015-02-14|         rolls/buns\r|
|         4501|2015-05-08|  other vegetables\r|
|         3803|2015-12-23|         pot plants\r|
|         2762|2015-03-20|         whole milk\r|
|         4119|2015-02-12|     tropical fruit\r|
|         1340|2015-02-24|       citrus fruit\r|
|         2193|2015-04-14|               beef\r|
|         1997|2015-07-21|        frankfurter\r|
|         4546|2015-09-03|            chicken\r|
|         4736|2015-07-21|             butter\r|
|         1959|2015-03-30|fruit/vegetable j...|
|         1974|2015-05-03|packaged fruit/ve...|
|         2421|2015-09-02|          chocolate\r|
|         1513|2015-08-03|      specialty bar\r|
|         1905|2015-07-07|  other vegetables\r|
+-------------+----------+--------------------+
only showing top 20 rows
```

Transformation

Removing \r at the end of itemDescription

9

```
from pyspark.sql.functions import regexp_replace
processed_df = df.withColumn("itemDescription", regexp_replace(df["itemDescription"], r"\r", ""))
processed_df.show()
```

```
|         2300|2015-09-19|           pip fruit|
|         1187|2015-12-12|    other vegetables|
|         3037|2015-02-01|          whole milk|
|         4941|2015-02-14|          rolls/buns|
|         4501|2015-05-08|    other vegetables|
|         3803|2015-12-23|          pot plants|
|         2762|2015-03-20|          whole milk|
|         4119|2015-02-12|      tropical fruit|
|         1340|2015-02-24|        citrus fruit|
|         2193|2015-04-14|                beef|
|         1997|2015-07-21|         frankfurter|
|         4546|2015-09-03|             chicken|
|         4736|2015-07-21|              butter|
|         1959|2015-03-30|fruit/vegetable j...|
|         1974|2015-05-03|packaged fruit/ve...|
|         2421|2015-09-02|           chocolate|
|         1513|2015-08-03|       specialty bar|
|         1905|2015-07-07|    other vegetables|
+-------------+----------+--------------------+
only showing top 20 rows
```

Extracting Day, Month, Year and Week from Date

11

```
from pyspark.sql.functions import year, month, dayofmonth, weekofyear

processed_df = processed_df.withColumn("Year", year(processed_df["Date"])) \
                           .withColumn("Month", month(processed_df["Date"])) \
                           .withColumn("Day", dayofmonth(processed_df["Date"])) \
                           .withColumn("WeekOfYear", weekofyear(processed_df["Date"]))

processed_df.show()
```

```
|         2300|2015-09-19|           pip fruit|2015|    9| 19|        38|
|         1187|2015-12-12|    other vegetables|2015|   12| 12|        50|
|         3037|2015-02-01|          whole milk|2015|    2|  1|         5|
|         4941|2015-02-14|          rolls/buns|2015|    2| 14|         7|
|         4501|2015-05-08|    other vegetables|2015|    5|  8|        19|
|         3803|2015-12-23|          pot plants|2015|   12| 23|        52|
|         2762|2015-03-20|          whole milk|2015|    3| 20|        12|
|         4119|2015-02-12|      tropical fruit|2015|    2| 12|         7|
|         1340|2015-02-24|        citrus fruit|2015|    2| 24|         9|
|         2193|2015-04-14|                beef|2015|    4| 14|        16|
|         1997|2015-07-21|         frankfurter|2015|    7| 21|        30|
|         4546|2015-09-03|             chicken|2015|    9|  3|        36|
|         4736|2015-07-21|              butter|2015|    7| 21|        30|
|         1959|2015-03-30|fruit/vegetable j...|2015|    3| 30|        14|
|         1974|2015-05-03|packaged fruit/ve...|2015|    5|  3|        18|
|         2421|2015-09-02|           chocolate|2015|    9|  2|        36|
|         1513|2015-08-03|       specialty bar|2015|    8|  3|        32|
|         1905|2015-07-07|    other vegetables|2015|    7|  7|        28|
+-------------+----------+--------------------+----+-----+---+----------+
only showing top 20 rows
```

12

```
processed_df.describe().show()
```

```
+-------+-----------------+-------------------+------------------+-----------------+------------------+-----------
-------+
|summary|    Member_number|    itemDescription|              Year|            Month|               Day|        Wee
kOfYear|
+-------+-----------------+-------------------+------------------+-----------------+------------------+-----------
-------+
|  count|            38765|              38765|             38765|            38765|             38765|
38765|
|   mean|  3003.64186766413|               NULL| 2014.528517993035|6.487604798142654|15.743196182123048| 26.6410937
7015349|
| stddev|1153.6110310565432|               NULL|0.4991925003201702|3.419042070046702|  8.81681415804242|14.91888736
9064297|
|    min|              1000|Instant food prod...|              2014|                1|                 1|
1|
|    max|              5000|           zwieback|              2015|               12|                31|
53|
+-------+------------------+--------------------+------------------+-----------------+------------------+-----------
-------+
```

Load

Loading into a delta table

Delta Table: A Delta Table is a type of table in the Delta Lake format, an open-source storage layer that brings ACID transactions, schema enforcement, and data versioning to data lakes. Delta Tables provide reliable data storage and management, enabling robust data pipelines and analytics.

15

```
processed_df.write.format("delta").mode("overwrite").saveAsTable("Groceries_Transactions")
```

Checking if the data has been stored succesfully in Delta Table

17

```
spark.sql("SELECT COUNT(*) AS RowCount FROM Groceries_Transactions").show()
```

```
+--------+
|RowCount|
+--------+
|   38765|
+--------+
```

Data Loading Succesful!