etl2 part 2: pyspark transformations

Table of Contents

[Overview 2](#_Toc182754885)

[Diagram 2](#_Toc182754886)

[1. Start a Spark Session in Python 2](#_Toc182754887)

[2. Extract data from HDFS 3](#_Toc182754888)

[3. Exploratory analysis of the data sets 4](#_Toc182754889)

[4. Assembling the target data for the business database 6](#_Toc182754890)

[5. Data checks before exporting to MySQL database 9](#_Toc182754891)

# Overview

In this part of the overall project, Hadoop and Spark has been created and it is time to extract and transform data.

The data includes B2B sales of a company that sells computer hardware. In an effort to cut costs, the company would like to see which products are selling below retail price. They would also like to see which products sell above retail price. Finally, they would like to which regional team is performing the best.

The latest report have been uploaded to HDFS where the raw data resides. The following guide will look from the perspective of the data analyst who will extract the data and transform so it can be transferred to a SQL database for the business teams to create visualizations.

# Diagram

A computer and a cloud of information

Description automatically generated with medium confidence

# 1. Start a Spark Session in Python

|  |
| --- |
| Description: Spark must be initialized within Python to begin extracting the data. |
| 1. Run the following code to start the Spark session: |

# 2. Extract data from HDFS

|  |
| --- |
| Description: With the Spark session enabled, it is time to extract the data from HDFS. |
| 1. Set variables to represent the HDFS paths of each file:    2. Set variables for each data file that will be read from:    3. Verify the data read successfully: |

# 3. Exploratory analysis of the data sets

|  |
| --- |
| Description: The objective is to prep the data to be sent to the SQL database. Each file needs to be explored to find the right columns that can be used to make one data set that will help teams using the SQL database to answer the business requirements. |
| 1. Look up the columns of each data set by executing the following code:    2. Observe the results    From the results, it appears the ‘Sales Pipeline’ data set has the data necessary to meet the business requirements. However, it doesn’t capture the region generating sales. The regional detail is reflected in the ‘sales team’ data set.  The link between the two data sets is the ‘sales\_agent’ column. Using this column as a key, the regional office can be merged into the ‘sales pipeline’ data set according to which sales agent made the transaction.  4. With the ‘sales\_pipeline’ data being identified as being the data set to indicate sales for products, examine a sample of rows to see what these transactions look like by running the following code:    Two points of interest based on the results:  1) The ‘close\_value’ represents the money made off the transaction and is dependent on the ‘deal\_stage’ being won to receive it.  2) Revenue made off the same product is inconsistent. Ex. GTX Basic sold for $588 on one deal, while it $517 on a different deal.  Point 2 needs further exploring, so the ‘products’ data set must be looked at for product details. Run the following code:    Using the example product of GTX Basic, it can be seen it’s sales price is different from what it is actually being sold for. This would mean money is being made or lost on a deal even if a deal was won. To meet the business requirement of seeing which product is selling more than the sales price and which one is lower, an additional column will have to be created for the target data set to determine if the deal gained or loss money based on the selling price. |

# 4. Assembling the target data for the business database

|  |
| --- |
| Description: Exploratory analysis revealed that the ‘sales pipeline’ data set would be the best set to serve as the basis for the data needed by the business. Also, two columns will have to be added to it to meet the requirement, which would be:   * A column that shows the regional office who made the sale. * A column that indicated if the product made more or less than its original selling price. |
| 1. Filter data by deals that were won as those are they only ones who determined revenue outcome. Run the following code:    2. Merge ‘regional\_office’ column to the filtered ‘winning\_deals’ dataframe. Also, select the relevant columns needed for the target data set. Run the following code:    3. Merge the ‘sales\_price’ column from the ‘products’ data set to the ‘region\_merge’ data set. Run the following code:    4. Create a column called ‘gain\_loss’ that shows if the deal made or lost money. Run the following code: |

# 5. Data checks before exporting to MySQL database

|  |
| --- |
| Description: A check needs to be made to ensure the final data set is clean and ready for deployment into the business database. |
| 1. Check for null values in the data. Run the following code:    2. Check data types of each column to ensure they make sense for the SQL database where the data will be deployed. Run the following code:    Congrats! You have created the data needed to meet the business requirements and are ready to deploy it to the database! |