Retail grocery analysis

Documented By

Aruna Balamurugan

F07064

**RETAIL GROCERY ANALYSIS**

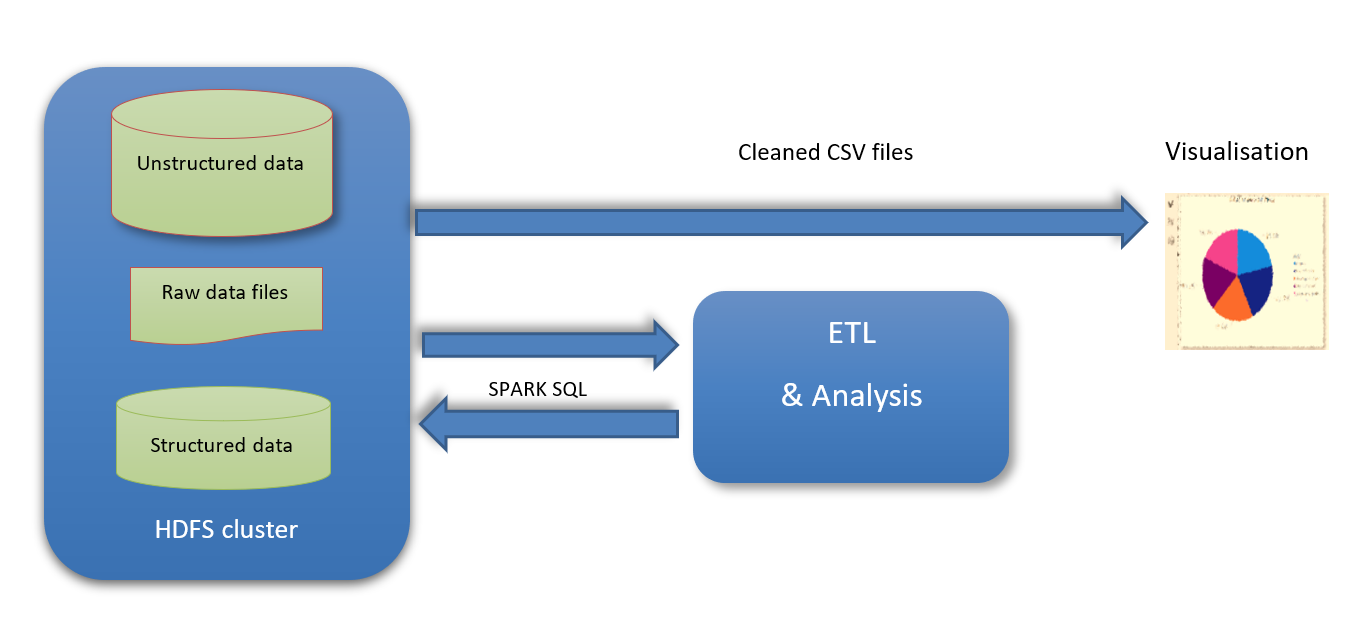
**Objective:**

* To perform analysis on a retail grocery company.
* To extract and build ETL pipeline and produce aggregated data into data lake.
* To analyse and provide business insights on improving sales volume and customer retention.

**Solution:**

* Extract unstructured raw data from HDFS located in: /user/insofe/retail/data/aws
* Use SparkSQL to read CSV file by specifying path location.
* Create schema and dataframes for each schema.
* Check for null values and redundancies.
* Merge dataframes and perform aggregations. One final dataframe for cleaned data.
* Remove duplicate columns
* Write dataframes into CSV file and load it back on HDFS. ETL pipelining is now achieved.
* Load the output CSV file into local machine and import to Power BI for visualisation.
* Remove duplicate rows and columns, null values, perform necessary transformation for analysis.
* Analyse for product-order insights by creating visuals.

**Solution Architecture:**



**Steps:**

1. Open Terminal in JupyterLab
2. Connect to Hadoop by using ssh connection:
   1. Provide IP address and password.
   2. Connect to HDFS by using hadoop fs -ls /path location

A screenshot of a computer

Description automatically generated

* 1. Use ls -ltr for listing all files and directories in the cluser

Graphical user interface, text, chat or text message

Description automatically generated

* 1. Exit the connection.
  2. Use scp cluster\_location local\_path . for loading all CSV files into local machine (Optional)

Text

Description automatically generated

1. Open another terminal to start connection and use Pyspark.

(Note: PySpark is used inside the connection throughout the ETL pipeline).

Text

Description automatically generated

1. Start a spark session by importing necessary packages.
2. Use spark.read.csv for reading the loaded CSV file:

Text

Description automatically generated

Fig.1: df.show() is used to display data in aisle.csv

1. Schema Creation:

* StructType() is used to create schema by defining fields to allow nullable(True) or not (False)
* To convert schema to dataframe, use df=spark.read.csv(“path”,header=True,schema=schema)

This dataframe df reads file located in “path”, defines schema as mentioned in “schema” variable above and keeps the first row as the header.

* 1. aisle.csv

A screenshot of a computer

Description automatically generated with medium confidence

Fig.2: schema\_ais = schema for aisle; df\_ais = dataframe for aisle schema.

* 1. department.csv

A screenshot of a computer

Description automatically generated with medium confidence

Fig.3: schema\_dept = schema for dept data; df\_dept = dataframe for dept schema.

* 1. order.csv

A screenshot of a computer

Description automatically generated

Fig.4: schema\_orders = schema for orders data; df\_order = dataframe for orders schema.

* 1. prior\_order.csv

A screenshot of a computer

Description automatically generated with medium confidence

Fig.5: schema\_prior\_order = schema for prior\_order data; df\_po = dataframe for prior\_order schema.

* 1. product.csv

A screenshot of a computer

Description automatically generated with medium confidence

Fig.6: schema\_pdt = schema for product data; df\_pdt = dataframe for product schema.

* 1. train\_order.csv

A screenshot of a computer

Description automatically generated with medium confidence

Fig.7: schema\_train\_order = schema for train\_orders data; df\_to = dataframe for train\_order schema

1. Displaying column names:

Use print(df.columns) to print column names as a list:

A screenshot of a computer

Description automatically generated

1. Display the datatypes of the columns:

Use df.dypes for diplaying data types of all columns in a dataframe.

For loop is used to display column and its datatypes one after the other, separated by a comma.

A screenshot of a computer

Description automatically generated with medium confidence

1. Check for Null values in the columns:

Use df.column\_name.isNull() in each column of a dataframe to display all the null values present.

Use df.filter() to display according to the specified condition (null values, in this case)

Use count() to display the count of null values in each column.

1. Orders.csv (df\_order)

A screenshot of a computer

Description automatically generated with medium confidence

1. Department.csv (df\_dept) and aisle.csv (df\_ais)

A screenshot of a computer

Description automatically generated

1. Prior\_order.csv (df\_po) and train\_orders.csv (df\_to)

A screenshot of a computer

Description automatically generated

1. Products.csv (df\_pdt)

A screenshot of a computer

Description automatically generated with medium confidence

1. Merging of datasets:

Use df.join() to merge two or more datasets.

Join conditions used:

1. Left Join Product with aisle using aisle\_id.
2. Left join above joined product with department using department\_id.
3. Inner join above new joined product with prior\_orders and train\_orders with product\_id.
4. Left join Orders with prior\_order and train\_orders using order\_id.
5. Product – Left Join - aisle:

A screenshot of a computer

Description automatically generated with medium confidence

1. Product\_aisle – Left Join - department:

Graphical user interface, text

Description automatically generated

1. Prior\_order – Inner Join – joined\_product :

Graphical user interface, text

Description automatically generated

1. Train\_order – Inner Join – joined\_product:

Graphical user interface

Description automatically generated with medium confidence

1. Order – Left Join – prior\_orders:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Order – Left Join – Train\_orders:

A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence

1. Aggregation of datasets:

Use df1.union(df2) to combine two or more datasets.

* 1. Combining prior\_orders and train\_orders.

A picture containing graphical user interface

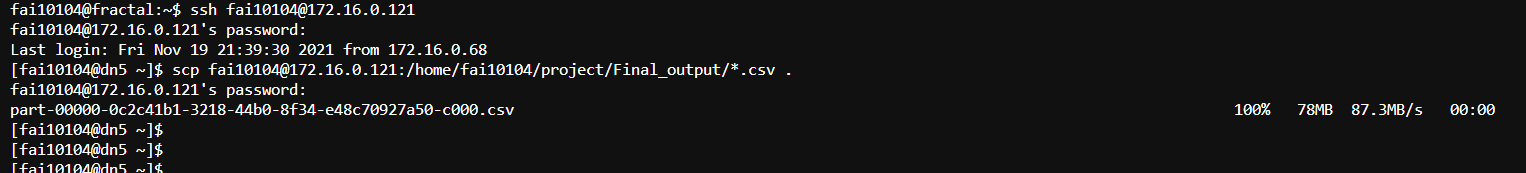
Description automatically generated

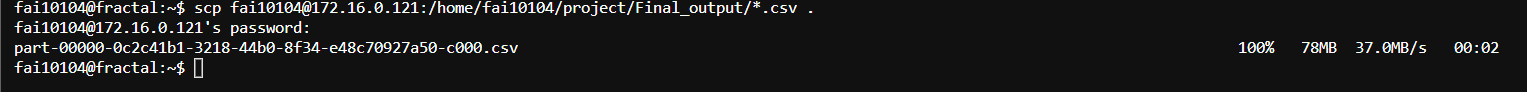
* 1. Combine all the datasets into one dataframe.

A screenshot of a computer

Description automatically generated with medium confidence

1. Loading CSV file into cluster and local machine:





1. Visualisation:

Transformation made before loading:

* 1. Removed duplicated rows and columns
  2. Removed empty/null values
  3. Created new table and relationship with the other table.

Graphical user interface

Description automatically generated with medium confidence

Graphical user interface, application

Description automatically generated

* 1. Created visuals:

Chart, pie chart

Description automatically generated

Insight: ‘Produce’ department accounted 41.9% of top 5 orders.

Chart, funnel chart

Description automatically generated

Insight: Fresh Fruits accounted around 87 thousand orders and frozen meals having lower sales. Improving cereals, frozen meals sales can drive business orders.

Graphical user interface, chart

Description automatically generated

Chart, bar chart

Description automatically generated

Insight: The highest orders are made around 11AM in an average week. 8AM-6PM is peak hours on average. Sunday has the highest sales while Wednesday has the lowest. Improving sales on middle of weekdays can increase business growth.

Graphical user interface

Description automatically generated

Insight: Customer\_number = 164055 contributes the most accounting 3060 orders. Offering sales discounts, freebies for other customers can increase sales orders.

Chart, bar chart

Description automatically generated

Insight: Banana has the highest reorders accounting 9800 orders.

Chart

Description automatically generated

Insight: Selecting by department slicer, pantry for example provides corresponding sales orders.

**Conclusion:**

The sales orders can be increased by focusing on beverages and frozen department. Irregular customers can avail more discounts and offers than regular customers, on less peak hours on least sales day of the week (Wednesday). This can maintain business growth throughout.

**Github repository:**

1. Repository creation:

Graphical user interface, text, application, email

Description automatically generated

1. Upload supporting files:

Graphical user interface, text, application, email

Description automatically generated

Github\_link: [https://github.com/arunahb/aruna\_fractal](https://protect2.fireeye.com/v1/url?k=997da7a1-f93a03f3-997a07d5-86f4d0ed5d79-d2d780af18bcb5cb&q=1&e=1d11fcc0-94c9-4bb6-9689-d23e1154f506&u=https%3A%2F%2Fgithub.com%2Farunahb%2Faruna_fractal)

**Files submitted:**

aruna\_document, aruna\_pbi, pyspark\_code, aruna\_ouput.csv