Group Final Project:

Market Basket Analysis for Online Retail Industry

Group Epsilon:

Yu Qiu (qiu.yu1@northeastern.edu)

Chengjie Zhang (zhang.chengj@northeastern.edu)

Fatima Nurmakhamadova (nurmakhamadova.f@northeastern.edu)

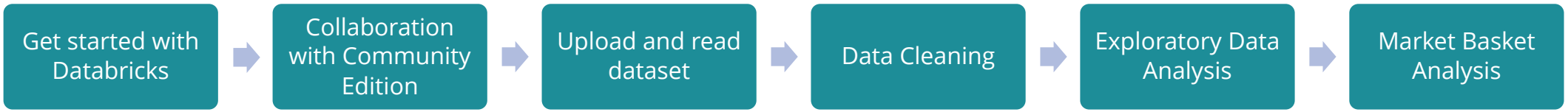
Class ALY6110. 80442:

Big Data and Data Management

June 27, 2022

AGENDA

1. **Summary**
2. **Content**



3. **Insights**
4. **Comments: Pros & Cons, Challenges**
5. **References**
6. **Q&A**

Summary

Summary

Purpose

Providing a comprehensive manual which could help analysts who have no experience using the big data management tool – Databricks to understand how to use it to tackle the real big data.

Business Problem

How to create a better product bundle sales strategy for online retail companies by using Market Basket Analysis?

- What are customer purchase time pattern from this store?
- What are the top 10 products selling the most?
- What are the top 10 Products Most Often Sold Together?
- Which country of customers placed most orders from this store?
- Market Basket Analysis: Apriori Algorithm & Association Rules

Dataset

A dataset from Kaggle: transaction records from January 12, 2009, to September 12, 2011 of a UK-based online sales store

- 2 Excel sheets
- 1,067,371 rows/observations
- 8 columns/variables in the dataset

Methodology

- Analysis Algorithm:
  Apriori Algorithm – Market Basket Analysis
- Big data management tool:

Content

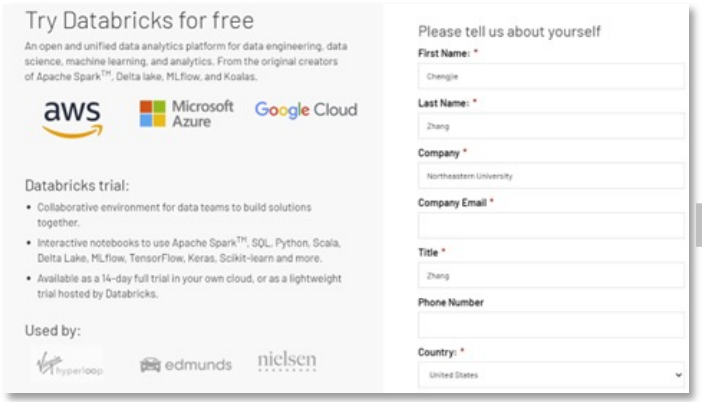
Get Started with Databricks

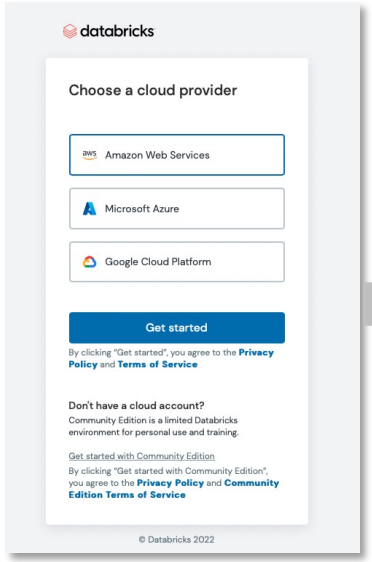
Step 1: Registration & Login (Community Edition)

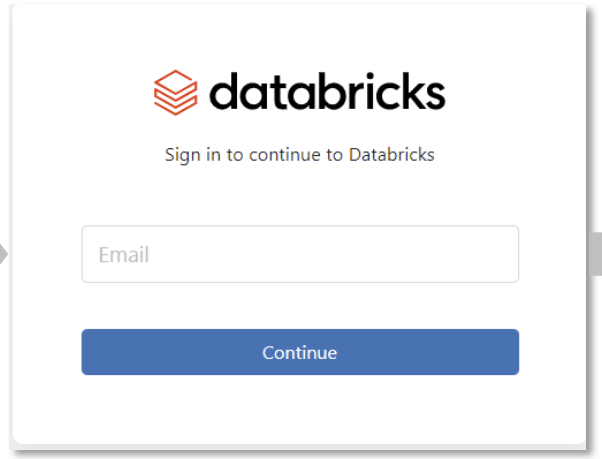
What is DataBricks

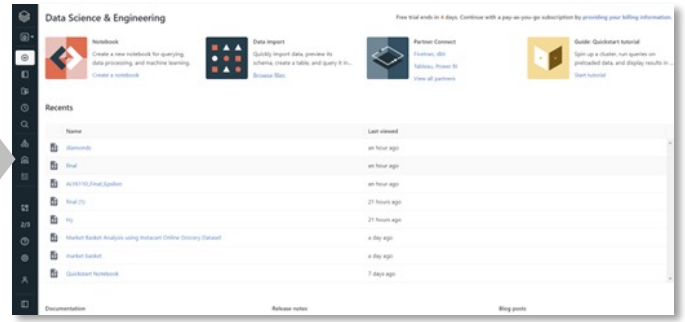
Databricks is to provide a separate space for processing data, independent of the hosting environment and Hadoop cluster management, and the entire process is done in the cloud

Start using *https://databricks.com/try-databricks*







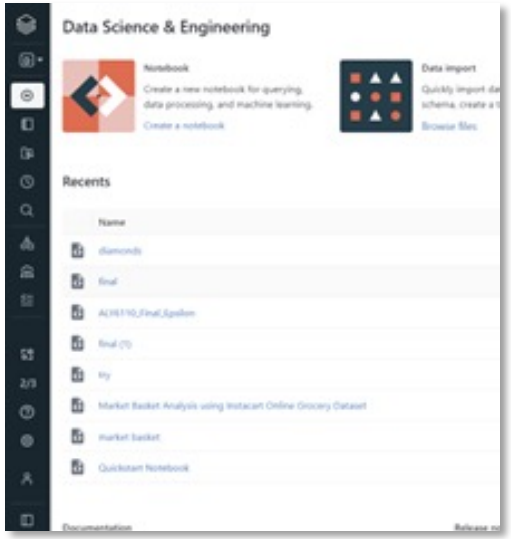


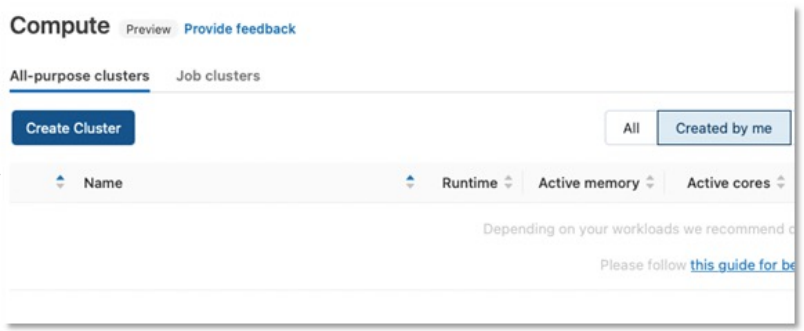
Register on website **Choose community** **Login by account** **Main page**

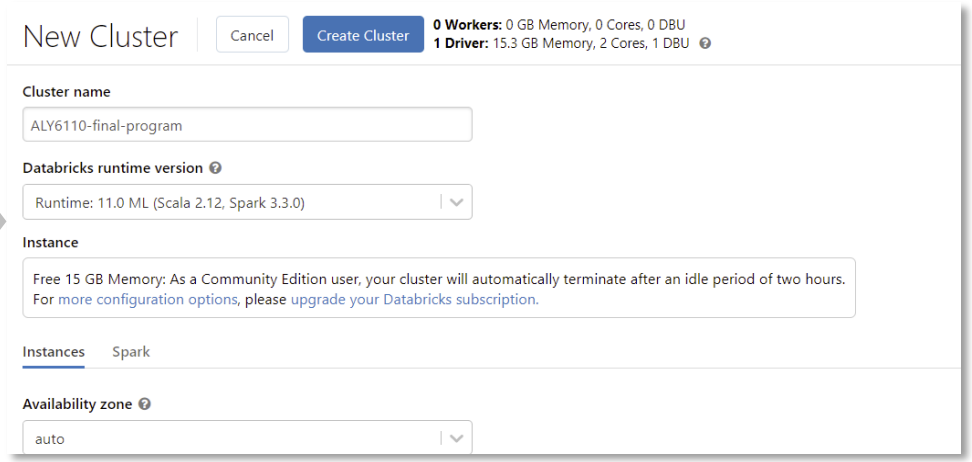
Step 2: Create a Cluster & a Notebook

Create a Cluster

Cluster represents the computing resources that run notebooks and codes, is used to store corresponding configuration information.

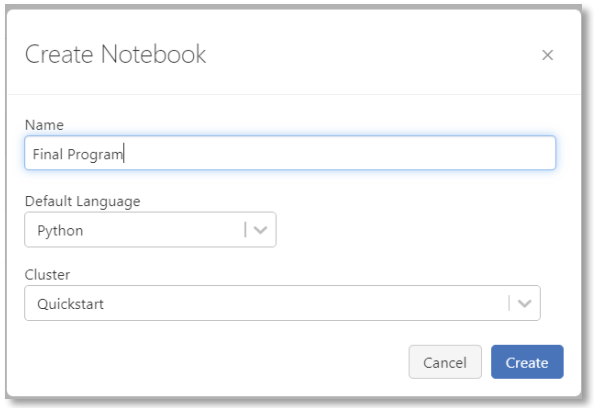


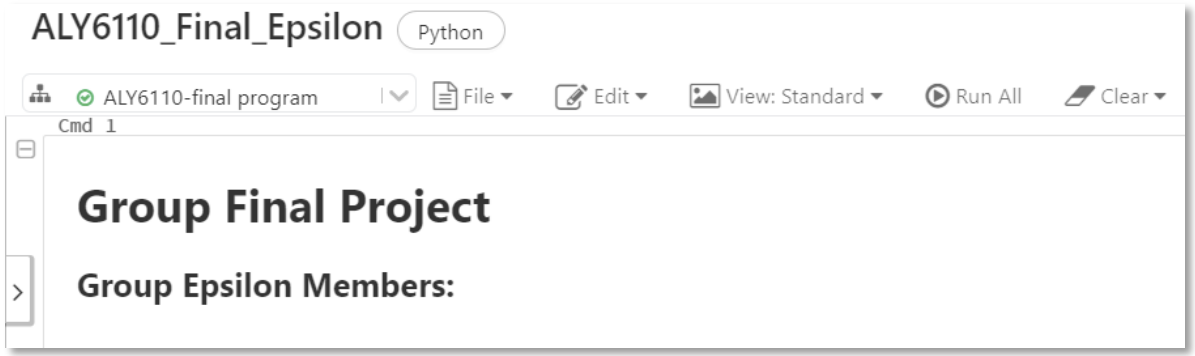




Create a Notebook

Notebook is a notepad that contains executable commands





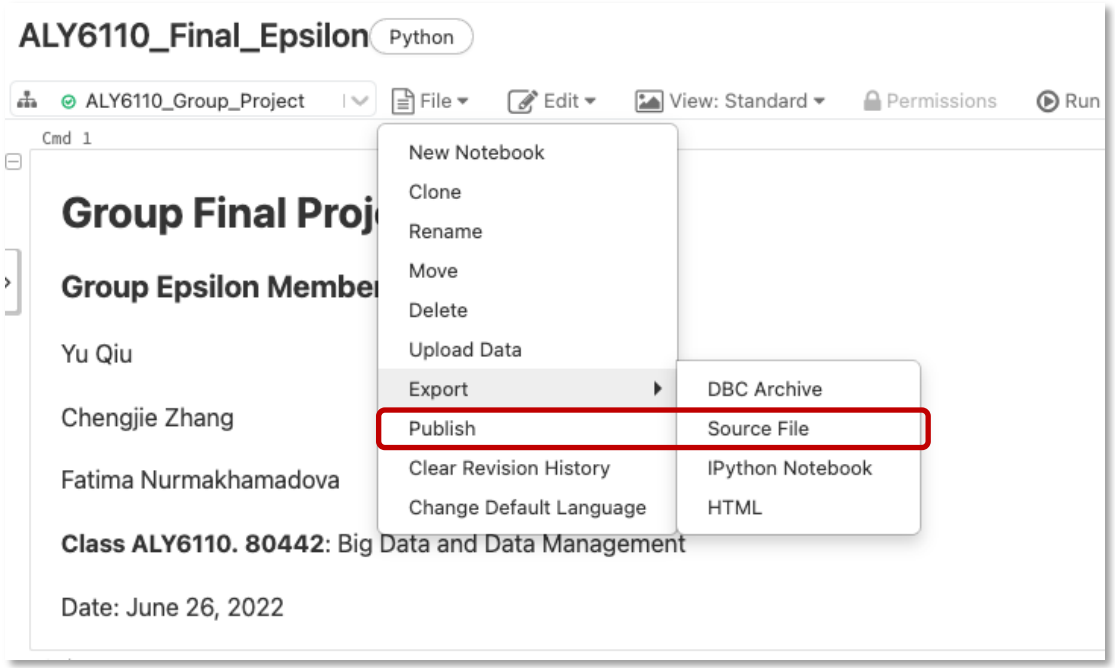
Content

Collaboration with Community Edition

Collaboration with Community Edition

Step 1 Download DBC File

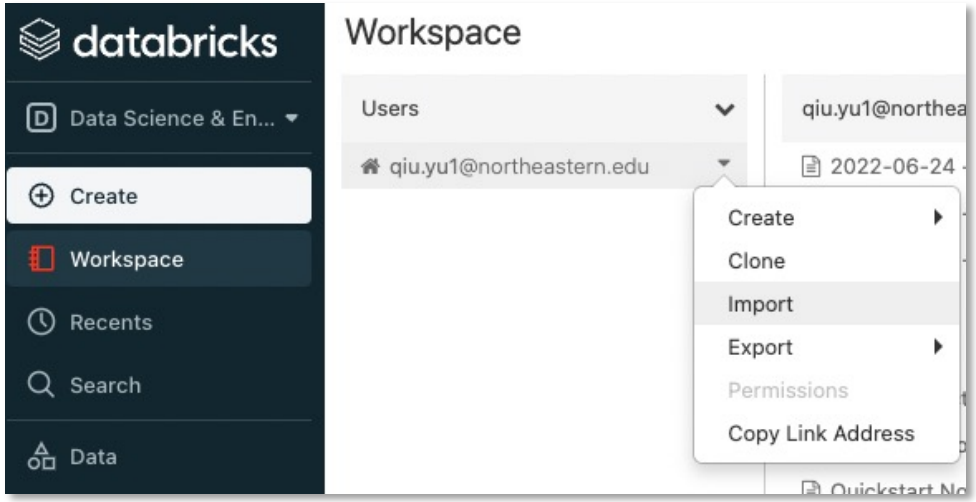
1. Go to the top menu of Notebook and click “File”
2. Chose Export and then click DBC Archive



Step 2: Share the DBC file with teammates

Step 3: Upload DBC Files

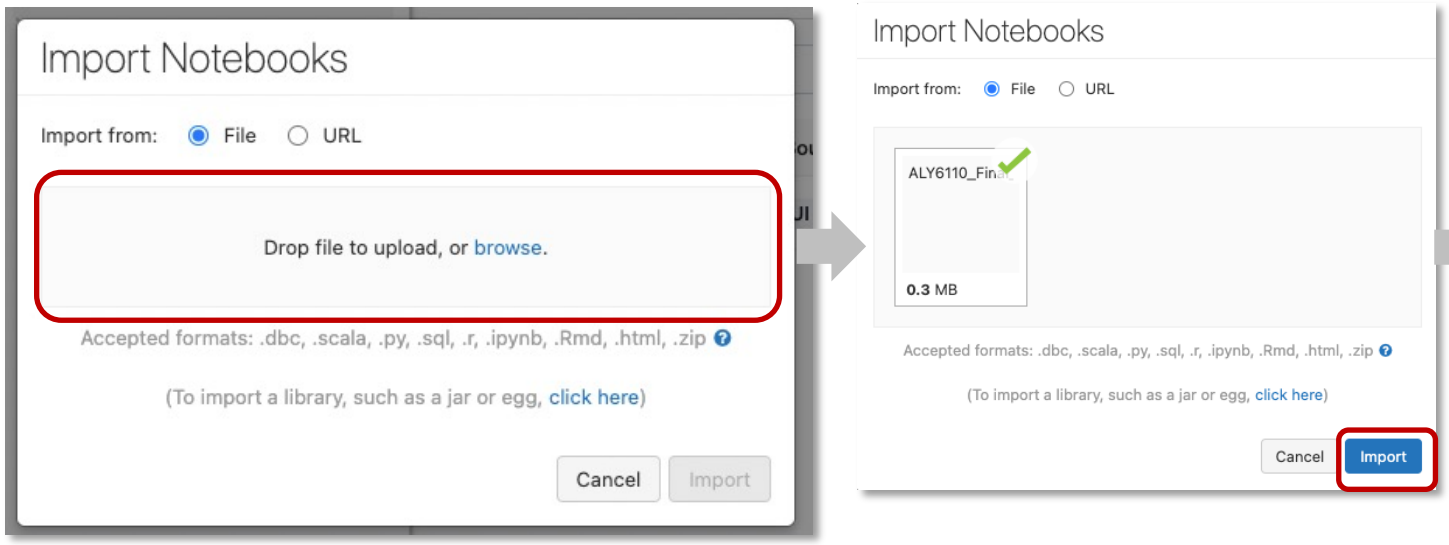
1. Go to Databricks Portal and click “Workspace” in the left menu bar;
2. Then click the arrow beside your user account and choose “Import”;



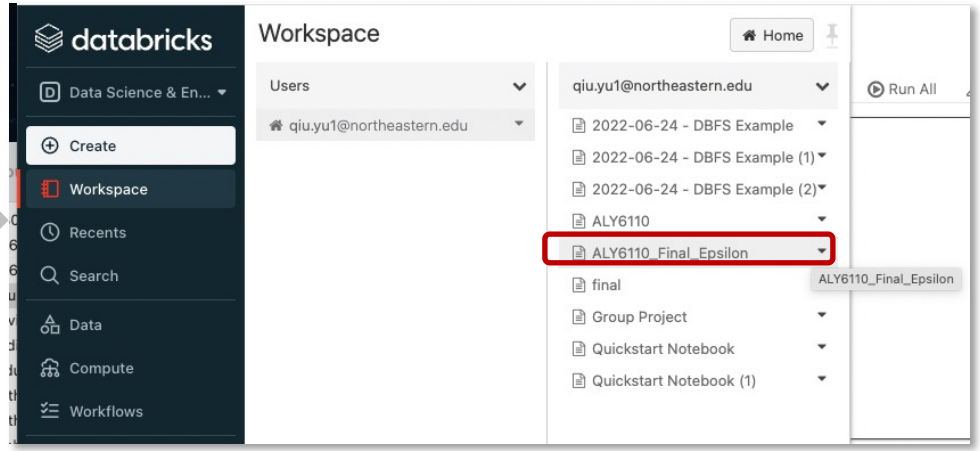
Collaboration with Community Edition

Step 3: Upload DBC Files

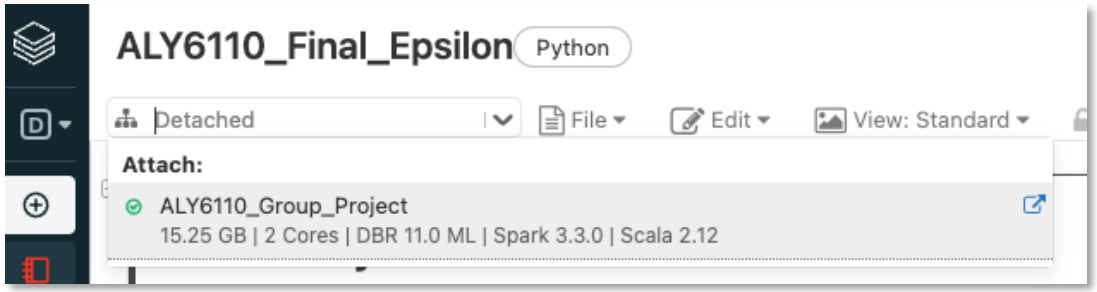
3. Drag the DBC file into the box “Drop file to upload”, or click “browse” to upload;



4. Double click the file and the Notebook will be opened in Databricks;



Step 4: Choose Cluster and Check File Path




```
# File location and type
#Please upload two attached files to DBFS, particularly in /FileStore/tables/
f1_loc = "/FileStore/tables/online_retail_2009_2010.csv"
f2_loc = "/FileStore/tables/online_retail_2010_2011.csv"
file_type = "csv"
```

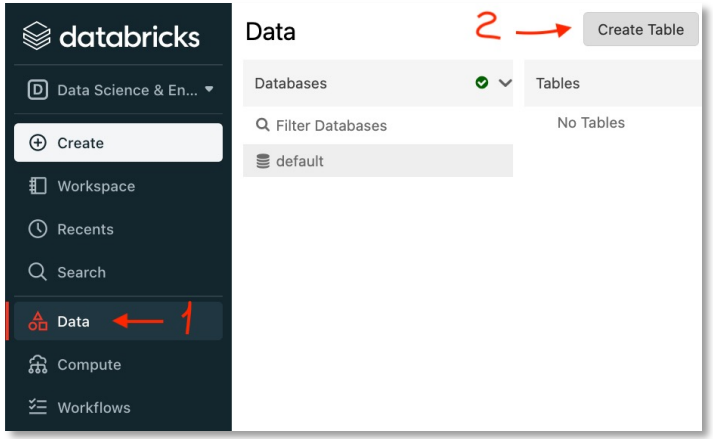
Content

Upload and read dataset

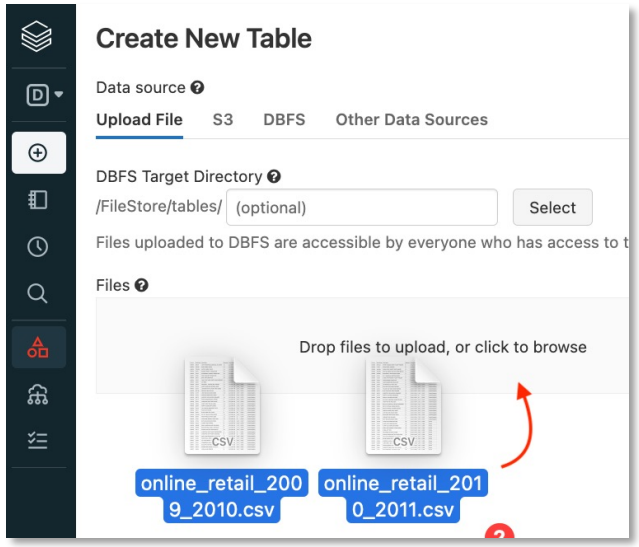
Step 3: Upload Dataset

Step 1: Upload the CSV Files

1. Choose “Data” in the left menu bar;
2. Then click to ’Create Table’

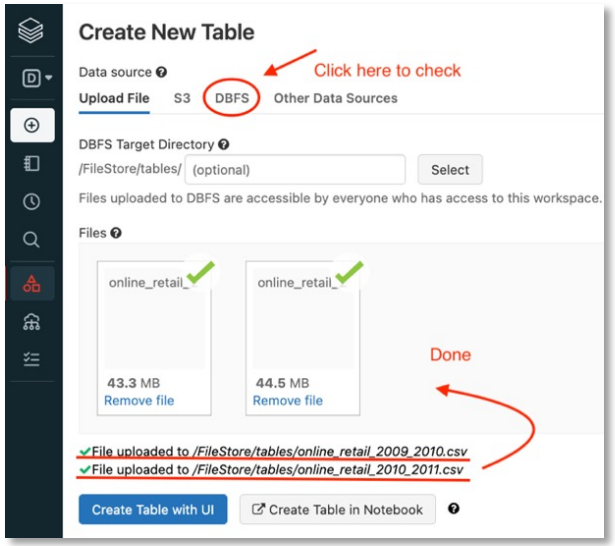


1. Drag the CSV files into the box “Drop file to upload”;



Step 2: Copy the File Path

1. Once it’s uploaded, the green stick will appear;
2. The ‘File uploaded to’ shows the file path it was uploaded to. Copy it



Step 4: Read Dataset in Notebook

Step 1 - Prepare Options for Reading the Files

1. Assign copied file paths for two files separately
2. Set necessary options

```
# File location and type
#Please upload two attached files to DBFS, particularly in /FileStore/tables/
f1_loc = "/FileStore/tables/online_retail_2009_2010.csv"
f2_loc = "/FileStore/tables/online_retail_2010_2011.csv"
file_type = "csv"
```

```
# CSV options of file parameters
infer_schema = "false" #true
first_row_is_header = "true"
delimiter = ","
```

Step 2 - Read a CSV file into Spark Data Frame

1. Assign copied file paths for two files separately
2. Set necessary options

```
# The applied options are for CSV files. For other file types, these will be ignored.
# Import csv file for 2009 - 2010 and read it into Spark data frame
df1 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(f1_loc)

# Import csv file for 2010 - 2011 and read it into Pyspark data frame
df2 = spark.read.format(file_type) \
  .option("inferSchema", infer_schema) \
  .option("header", first_row_is_header) \
  .option("sep", delimiter) \
  .load(f2_loc)
```

Step 3 – Convert to Pandas DF and Combine

```
# convert Spark data frames to pandas dataframes
df1_pandas = df1.toPandas()
df2_pandas = df2.toPandas()

#Join dataset from 2009-2010 and 2010-2011
data =pd.concat([df1_pandas, df2_pandas])
```

1. Convert each Spark DFs into Pandas DFs
2. Combine two data frames into one

Content

Data Cleaning

Data Cleaning

1 –Data Types

1. Convert **Invoice Date** to date time format
2. Convert **Quantity** to integer data type
3. Convert **Price** to float data type
4. Keep the rest as object data type

2 – Missing & Meaningless Values

1. Drop rows with missing values below 5%
2. Replace nulls in **Customer ID** with ‘non-members’ value
3. Drop values <= 0 in **Price** and **Quantity**
4. Drop doubtful values in **Description** (Product)

Cleaned Dataset Info

1,033,437 observations, and **8** features

Information about Variables

No.	Variables	Meaning	Data Type
1	**Invoice**	The invoice number made by a customer; those starts with “C” are the canceled transactions.	Categorical
2	**Stock Code**	Unique product code for each product	Categorical
3	**Description**	Product name	Categorical
4	**Quantity**	Number of products sold in each transaction	Numeric: discrete
5	**Invoice Date**	The date of invoices happened	Date & Time
6	**Price**	The price of the product	Numeric: continuous
7	**Customer ID**	Unique customer ID for each customer; customers who do not register will have not customer ID	Categorical
8	**Country**	The customers’ location	Categorical

Content

Exploratory Data Analysis

Pandas Data Frame ⇄ Spark Data Frame ⇄ SQL Tables

1. We can convert several forms of data Frame into each other by Scala codes.
2. If you need to create a SQL table you can create a “Tempview”, and then you can use SQL syntax to write code.
3. The SQL function on SparkSession enables applications to run SQL queries programmatically and returns the result as a Data Frame

```
# create a spark session
ss = SparkSession.builder.appName('Test').getOrCreate()

# Enable Arrow-based columnar data transfers
spark.conf.set("spark.sql.execution.arrow.enabled", "true")


# Create a Spark DataFrame from a pandas DataFrame using Arrow
data_clean_spark = spark.createDataFrame(data_clean)

# Convert the Spark DataFrame back to a pandas DataFrame using Arrow
data_clean_pandas = data_clean_spark.select("*").toPandas()
```

```
#use Spark dataframe to create one table in SQL called "dfs"
#The entry point into all functionality in Spark is the SparkSession class. To create
from pyspark.sql import SparkSession
#SparkConf passed to your SparkContext. SparkConf allows you to configure some of the
from pyspark import SparkConf

spark = SparkSession.builder.config(conf=SparkConf()).getOrCreate()
data_clean_pyspark2.createOrReplaceTempView("dfs")
```

```
%sql
select count(Invoice) as total_invoices, hour from dfs Group by hour order by hour
```

▸ (2) Spark Jobs

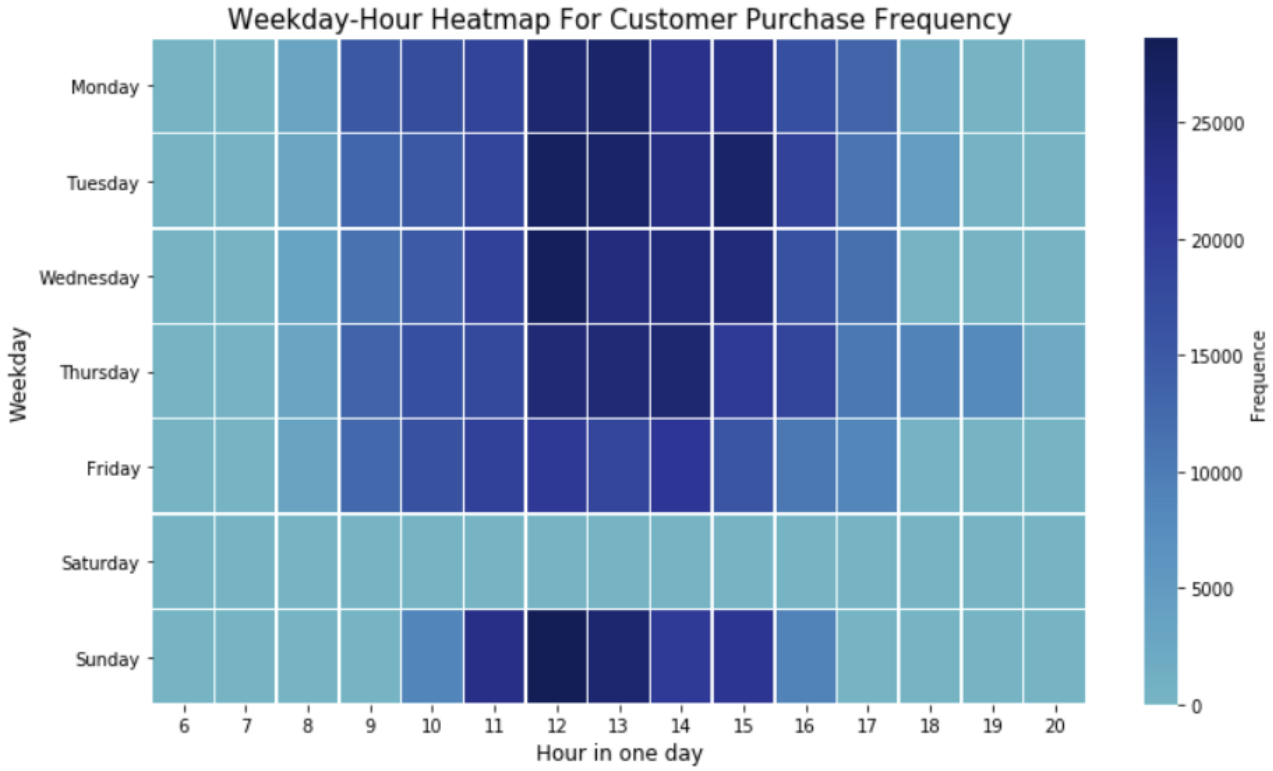
Table | Data Profile

	total_invoices	hour
1	1	6
2	1051	7
3	15532	8
4	66385	9
5	89781	10
6	117486	11
7	155430	12

Showing all 15 rows.

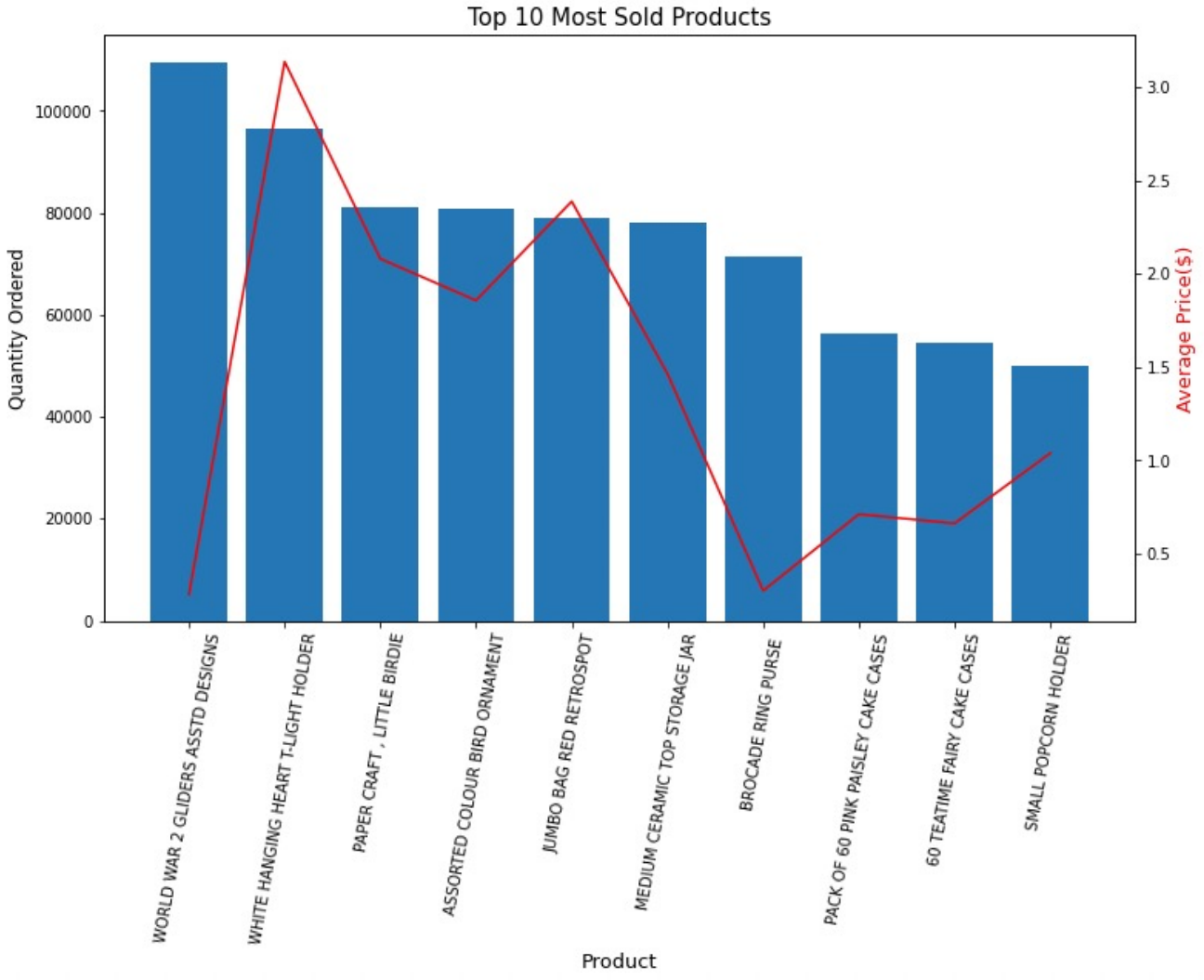
EDA: Which day and time period generated most transactions?

1. People are more likely to buy products at noon and afternoon time: 11am – 15pm
2. European people do not online shopping from this store on Saturday
3. Thursday has a longer shopping time period, while Sunday has the shortest



*We used Python & SQL syntax to create the above graph, detailed codes are provided in our report.

EDA: What are the top 10 products selling the most?



1. **Top 3 most sold products:**

- World War 2 Glider Assorted Designs > 100,000
- White Hanging Heart T-Light Holder < 100,000
- Paper craft, little birdie < 100,000

2. Among these top 10 **White Hanging Heart T-Light Holder** seems to generate **relatively higher revenue** to the store, as it has the highest unit price being among the top most sold

*We used Python syntax to create the above graph, detailed codes are provided in our report.

EDA: What are the top 10 Products Most Often Sold Together?

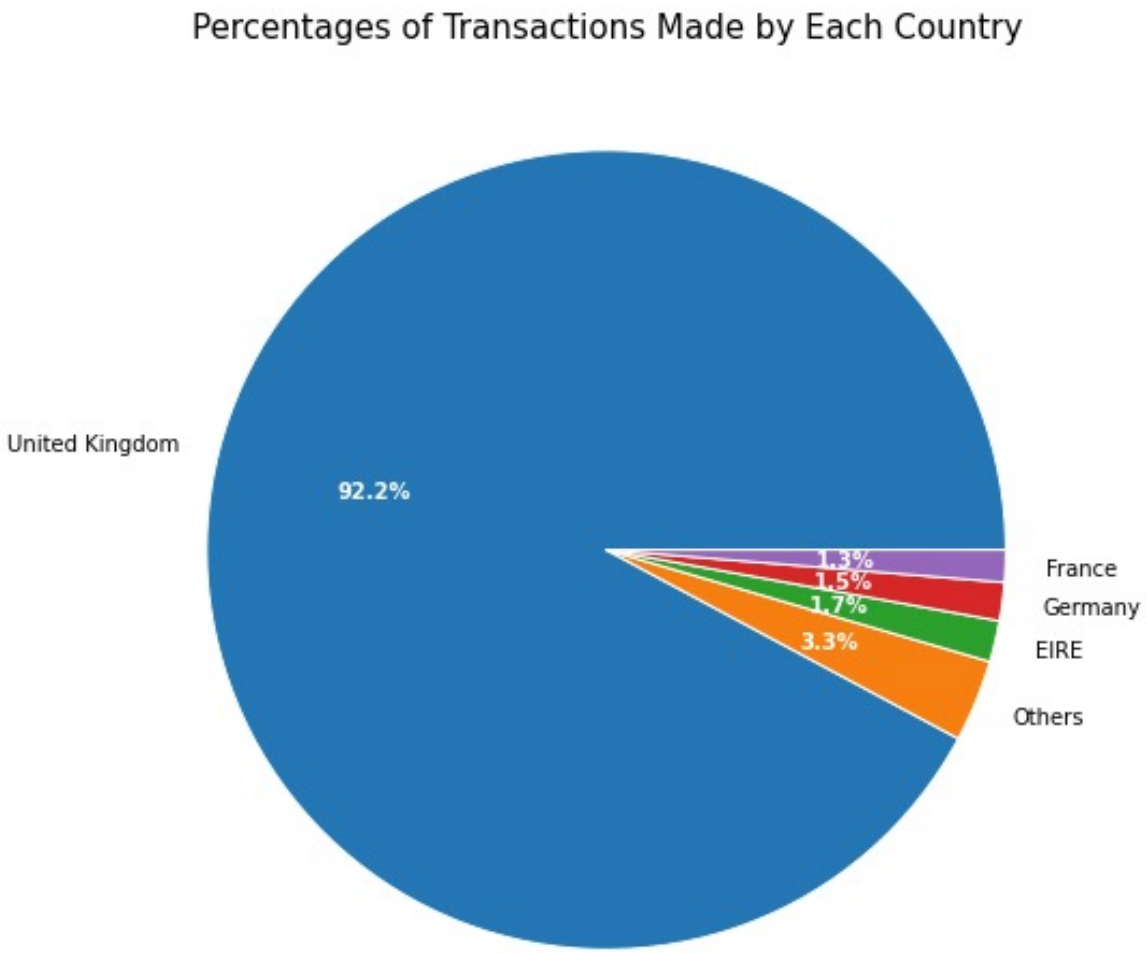
Count	Product Combination
2566	('KEY FOB ', 'KEY FOB ')
2516	('FRENCH BLUE METAL DOOR SIGN', 'FRENCH BLUE METAL DOOR SIGN')
1518	('KEY FOB ', ' SHED')
1501	('KEY FOB ', ' BACK DOOR ')
1128	('KEY FOB ', ' FRONT DOOR ')
1018	('KEY FOB ', ' GARAGE DESIGN')
977	('HOOK', 'MAGIC GARDEN')
920	('METAL SIGN', 'CUPCAKE SINGLE HOOK')
876	('COFFEE', 'SUGAR')
827	('RED HANGING HEART T-LIGHT HOLDER', 'WHITE HANGING HEART T-LIGHT HOLDER')

Most common combinations with the 2 product bundles

- **Key fob** product is found in most combinations
- Combination involves **complementing** products

*We used Python syntax to create the above graph, detailed codes are provided in our report.

EDA: Which country of customers placed most orders from this store?



1. Top 4 countries with most transactions:

United Kingdom (UK), EIRE (Ireland), Germany and France

2. **Transactions from UK** occupies the most part of all the transactions which is **over 92%.**

This online store might need try to attract more transactions from Non-UK countries to generate more revenue.

*We used Python syntax to create the above graph, detailed codes are provided in our report.

Content

Market Basket Analysis

Market Basket Analysis

Algorithm

Apriori Algorithm

Package need to be installed

Package: 'mlxtend'

Library:

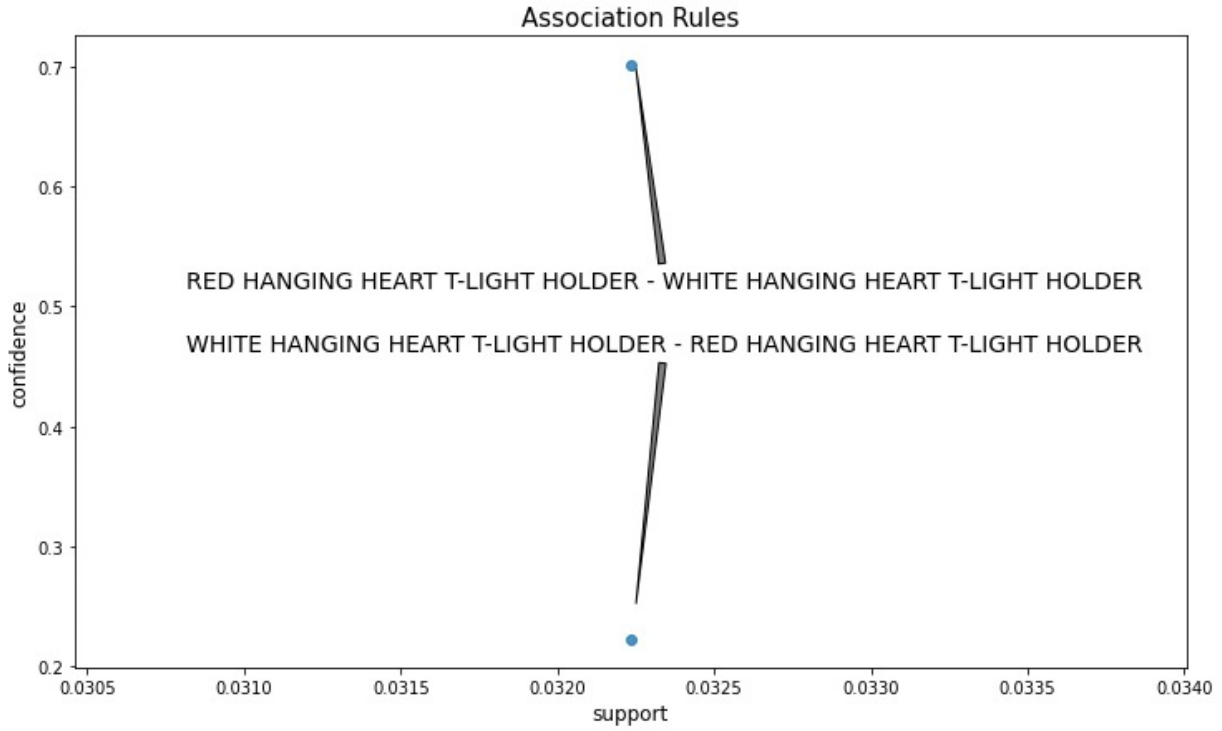
- **apriori**: to directly calculate the frequency of 2 products appear in same order
- **association_rules**: calculate the metrics: Lift, Confidence, support

Subset

UK Transactions

Result

antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage
(WHITE HANGING HEART T-LIGHT HOLDER)	(RED HANGING HEART T-LIGHT HOLDER)	0.14428	0.04595	0.032234	0.223412	4.862071	0.025604
(RED HANGING HEART T-LIGHT HOLDER)	(WHITE HANGING HEART T-LIGHT HOLDER)	0.04595	0.14428	0.032234	0.701502	4.862071	0.025604



- Strong association since Lift >1;
- 70% probability that people buy Red Holder with White Holder **together more** when they put buy **Red** Holder in carts;
- While only 22% probability that people buy **these 2 together when put White one in carts**;

Suggest to provide a discount on buying Red Holder when they put White Holder in Carts

*We used Python syntax to create the above graph, detailed codes are provided in our report.

Insights

Recommendations

- Increase the profit margins of **low-price items**
- Finding the **best combinations** of low-price items that can be sold together
- Try the **cross-promotions** to Non-UK customers and check whether this will help to generate more transactions
- Provide **promotions** during the non-peak times to release the website pressure
- Providing a **discount** on Red Holders when people put White Hanging Heart T-Light Holder in their carts

Future Analysis

- Analyze purchase time patterns of specific product categories
- Conduct Market Basket Analysis for other countries

Comments

Comments

Pros

- APIs to multiple programming languages: R, Scala, Python, SQL;
- Databricks can be connected to the Cloud we often use in our daily work and study, such as AWS, AZURE, etc.;
- When we select a table from SQL environment, it can be easily transformed into a graph;
- There are many useful packages are available in the environment, so we don't need to install many packages from pip install command;
- The speed to run the commands is much quicker than Jupiter Notebooks;

Cons

- There is a limited number of clusters could be created in Databricks community edition. The terminated cluster could not be restarted, so every time to reopen the notebook, we need to create a new cluster for the notebook.
- When we need to use different ways to analyze one dataframe, we need to establish it as a Spark dataframe, Python dataframe and a SQL table, though they are all the same.
- It needs a relatively long time to have the cluster fully started, usually 3- 5 minutes.
- Community Edition need more time to run the commands than Full Edition using Cloud AWS/ Azure;

Comments

Challenges

- Directly upload the original Excel file which contains 2 sheets in it and use Databricks to read and merge the 2 sheets.
- When we use InferSchema, we met problems in changing some variables types, and replacing NaN;
- MBA min support value could only be 0.03. When we set a smaller value for example 0.003, the result shows error. So, our analysis will not be very comprehensive

Python

```
#import apriori library which could be used to directly calculate the frequency of 2 products appear in same order
from mlxtend.frequent_patterns import apriori

#import association_rules which could calculate the metrics:Lift, Confidence, support
from mlxtend.frequent_patterns import association_rules

#Use Apriori algorithm to calculate the support with the threshold of minimun support 0.03
frequent_itemsets = apriori(basket, min_support= 0.003, use_colnames=True)

#set the rule metric as lift with minimun value of 1
rules = association_rules(frequent_itemsets, metric="lift", min_threshold=1)
```

```
/local_disk0/.ephemeral_nfs/envs/pythonEnv-8b092253-cd82-415c-87ee-23331ad769e8/lib/python3.9/site-packages/mlxtend/frequent_patterns/fpcommon.py:111: DeprecationWarning: DataFrames with non-bool types result in worse computationalperformance and their support might be discontinued in the future.Please use a DataFrame with bool type
  warnings.warn(
MemoryError: Unable to allocate 1.23 TiB for an array with shape (2325246, 2, 36238) and data type int64
Command took 3.65 seconds -- by qiu.yu1@northeastern.edu at 6/26/2022, 7:48:55 PM on ALY6110_Group_Project
```

Reference

- Cagirici, O. *Online Retail Dataset*. Kaggle.com. Retrieved 17 June 2022, from https://www.kaggle.com/datasets/ozlemilgun/online-retail-dataset?resource=download.
- *Clusters - Azure Databricks*. Docs.microsoft.com. (2022). Retrieved 18 June 2022, from https://docs.microsoft.com/en-us/azure/databricks/clusters/.
- Garg, A. (2022). *Apache Spark Tutorial - Learn Spark & Scala with Hadoop - Intellipaat*. Intellipaat Blog. Retrieved 10 June 2022, from https://intellipaat.com/blog/tutorial/spark-tutorial/.
- Kadlaskar, A. (2021). *Market basket Analysis | Guide on Market Basket Analysis*. Analytics Vidhya. Retrieved 11 June 2022, from https://www.analyticsvidhya.com/blog/2021/10/a-comprehensive-guide-on-market-basket-analysis/.
- *What is Databricks Runtime? - Databricks*. Databricks. Retrieved 18 June 2022, from https://databricks.com/glossary/what-is-databricks-runtime.

Q & A
Thank You