**E-COMMERCE APPLICATION** **ON IBM CLOUD**

**FOUNDARY**

PHASE 3 : **DEVELOPMENT PART 1**

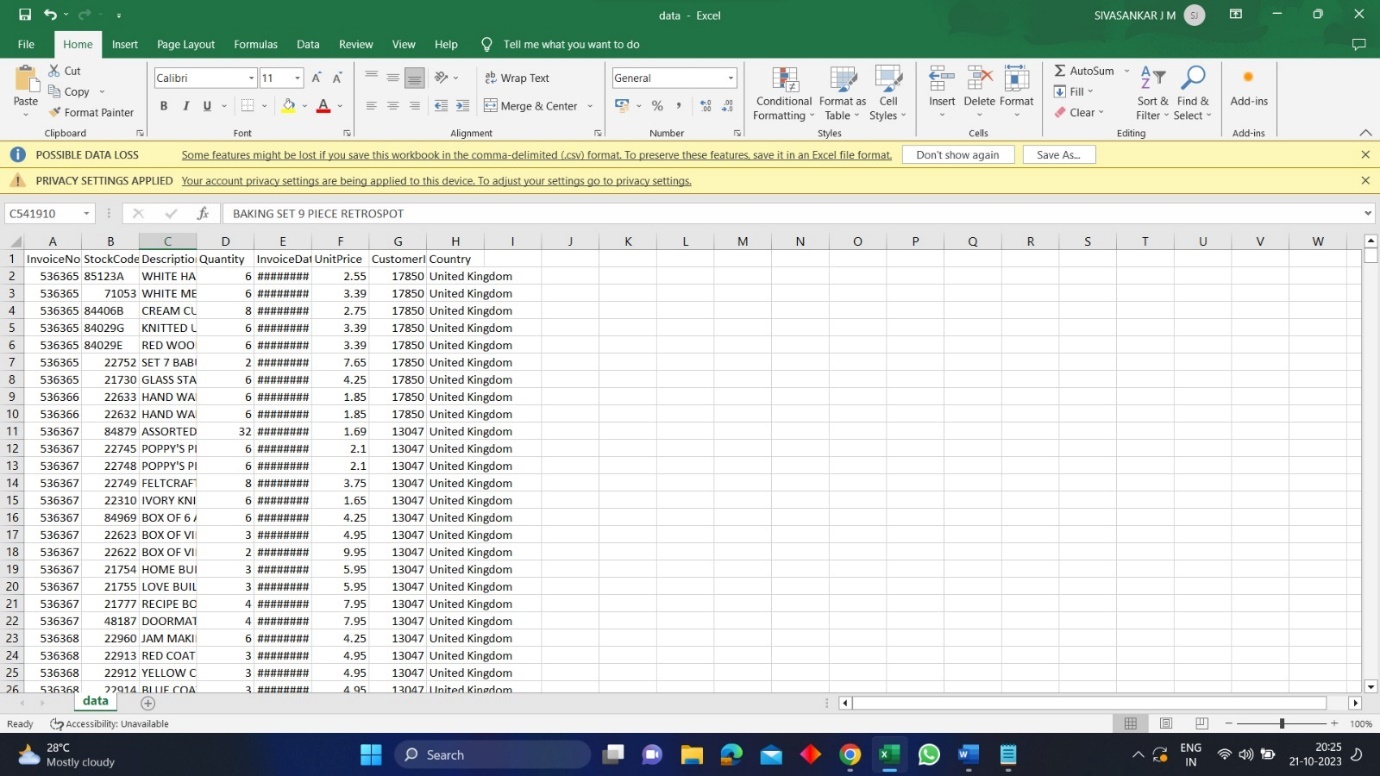
**PROBLEM DEFINITION**

The project at hand entails the creation of an artisanal e-commerce platform utilising IBM Cloud Foundry. The primary objective is to establish a platform that bridges skilled artisans with a global audience, showcasing their unique handmade products, and providing essential e-commerce functionalities such as secure shopping carts, seamless payment gateways, and an intuitive checkout process. This project encompasses designing the e-commerce platform, implementing necessary features, and ensuring a seamless user experience.

The instances that comprise the dataset :

* **Invoice No:** Invoice number. Nominal, a 6-digit integral number uniquely assigned to each transaction. If this code starts with letter 'c', it indicates a cancellation.
* **Stock Code:** Product (item) code. Nominal, a 5-digit integral number uniquely assigned to each distinct product.
* **Description:** Product (item) name. Nominal.
* **Quantity:** The quantities of each product (item) per transaction. Numeric.
* **Invoice Date:** Invoice Date and time. Numeric, the day and time when each transaction was generated.
* **Unit Price:** Unit price. Numeric, Product price per unit in sterling.
* **CustomerID:** Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.
* **Country:** Country name. Nominal, the name of the country where each customer resides.

SAMPLE DATASET COLLECTION :



**Basic EDA**

**Gaining a better understanding of the dataset**

import pandas as pd

data\_data=pd.read\_csv("/content/data.csv")

try:

df = pd.read\_csv('/kaggle/input/ecommerce-data/data.csv', encoding = 'latin')

except:

print("Error occurred while importing the data")

else:

print("data imported successfully")

data imported successfully

df.head()

|  | InvoiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 12/1/2010 8:26 | 2.55 | 17850.0 | United Kingdom |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 12/1/2010 8:26 | 3.39 | 17850.0 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 12/1/2010 8:26 | 2.75 | 17850.0 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 12/1/2010 8:26 | 3.39 | 17850.0 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 12/1/2010 8:26 | 3.39 | 17850.0 | United Kingdom |

In [4]:

*#Make column headings easier to work with and to read.*

try:

df.rename(index=str, columns={'InvoiceNo': 'invoice\_no',

'StockCode' : 'stock\_code',

'Description' : 'description',

'Quantity' : 'quantity',

'InvoiceDate' : 'invoice\_date',

'UnitPrice' : 'unit\_price',

'CustomerID' : 'cust\_id',

'Country' : 'country'}, inplace=True)

except:

print("Error")

df.head()

|  | invoice\_no | stock\_code | description | quantity | invoice\_date | unit\_price | cust\_id | country |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 12/1/2010 8:26 | 2.55 | 17850.0 | United Kingdom |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 12/1/2010 8:26 | 3.39 | 17850.0 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 12/1/2010 8:26 | 2.75 | 17850.0 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 12/1/2010 8:26 | 3.39 | 17850.0 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 12/1/2010 8:26 | 3.39 | 17850.0 | United Kingdom |

In [5]:

df.info()

<class 'pandas.core.frame.DataFrame'>

Index: 541909 entries, 0 to 541908

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 invoice\_no 541909 non-null object

1 stock\_code 541909 non-null object

2 description 540455 non-null object

3 quantity 541909 non-null int64

4 invoice\_date 541909 non-null object

5 unit\_price 541909 non-null float64

6 cust\_id 406829 non-null float64

7 country 541909 non-null object

dtypes: float64(2), int64(1), object(5)

memory usage: 37.2+ MB

In [6]:

df.describe()

Out[6]:

|  | quantity | unit\_price | cust\_id |
| --- | --- | --- | --- |
| count | 541909.000000 | 541909.000000 | 406829.000000 |
| mean | 9.552250 | 4.611114 | 15287.690570 |
| std | 218.081158 | 96.759853 | 1713.600303 |
| min | -80995.000000 | -11062.060000 | 12346.000000 |
| 25% | 1.000000 | 1.250000 | 13953.000000 |
| 50% | 3.000000 | 2.080000 | 15152.000000 |
| 75% | 10.000000 | 4.130000 | 16791.000000 |
| max | 80995.000000 | 38970.000000 | 18287.000000 |

**.**

In [7]:

*#Find NULL values in dataset*

df.isnull().sum().sort\_values(ascending=False)

Out[7]:

cust\_id 135080

description 1454

invoice\_no 0

stock\_code 0

quantity 0

invoice\_date 0

unit\_price 0

country 0

dtype: int64

In [8]:

*#Check datatypes for each column:*

**df**.**dtypes**

Out[8]:

**invoice\_no object**

**stock\_code object**

**description object**

**quantity int64**

**invoice\_date object**

**unit\_price float64**

**cust\_id float64**

**country object**

**dtype: object**

***Data Cleaning***

**The invoice date is currently set as an object so we will convert this to datetime as it is easier to work with.**

In [9]:

*#Cast invoice\_date to datetime*

**df['invoice\_date']** = **pd**.**to\_datetime(df['invoice\_date'])**

In [10]:

*#Check above was done correctly*

**df**.**dtypes**

Out[10]:

invoice\_no object

stock\_code object

description object

quantity int64

invoice\_date datetime64[ns]

unit\_price float64

cust\_id float64

country object

dtype: object

In [11]:

*#In this case, we will drop NULL values for simplicity*

try**:**

df\_new = df.dropna()

except**:**

**print("Error")**

else**:**

**print("Dropped NAs")**

Dropped NAs

Removing negative values

In [12]:

*#Round values to 2 decimal places*

**df\_new**.**describe()**.**round(2)**

Out[12]:

|  | quantity | unit\_price | cust\_id |
| --- | --- | --- | --- |
| count | 406829.00 | 406829.00 | 406829.00 |
| mean | 12.06 | 3.46 | 15287.69 |
| std | 248.69 | 69.32 | 1713.60 |
| min | -80995.00 | 0.00 | 12346.00 |
| 25% | 2.00 | 1.25 | 13953.00 |
| 50% | 5.00 | 1.95 | 15152.00 |
| 75% | 12.00 | 3.75 | 16791.00 |
| max | 80995.00 | 38970.00 | 18287.00 |

In [13]:

*#Remove negative quantities*

try**:**

df\_new =df\_new[df\_new.quantity > 0] *#removing quantity which is in negative.*

except**:**

**print("Error")**

else**:**

**print("Removed Negative Quantities")**

**Removed Negative Quantities**

In [14]:

*#Change customer ID to string datatype*

try:

df\_new['cust\_id'] = df\_new['cust\_id'].astype('str')

except:

print("Error")

else:

print("changed datatype")

changed datatype

In [15]:

*#Add total\_sales column*

df\_new['total\_sales'] = df\_new['quantity'] \* df\_new['unit\_price']

try:

df\_new = df\_new[['invoice\_no','invoice\_date','stock\_code','description',

'quantity','unit\_price','total\_sales','cust\_id','country']]

except**:**

**print("Error")**

else:

print("Column Added")

Column Added

In [16]:

*#Quick overview of current dataset*

df\_new.head()

Out[16]:

|  | invoice\_no | invoice\_date | stock\_code | description | quantity | unit\_price | total\_sales | cust\_id | country |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | 2010-12-01 08:26:00 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 2.55 | 15.30 | 17850.0 | United Kingdom |
| 1 | 536365 | 2010-12-01 08:26:00 | 71053 | WHITE METAL LANTERN | 6 | 3.39 | 20.34 | 17850.0 | United Kingdom |
| 2 | 536365 | 2010-12-01 08:26:00 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 2.75 | 22.00 | 17850.0 | United Kingdom |
| 3 | 536365 | 2010-12-01 08:26:00 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 3.39 | 20.34 | 17850.0 | United Kingdom |
| 4 | 536365 | 2010-12-01 08:26:00 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 3.39 | 20.34 | 17850.0 | United Kingdom |

In [17]:

*#Add Month, Day and Hour columns to the dataset*

df\_new.insert(loc=2, column='year\_month', value=df\_new['invoice\_date'].map(lambda x: 100\*x.year + x.month))

df\_new.insert(loc=3, column='month', value=df\_new.invoice\_date.dt.month)

*# +1 to make Monday=1.....until Sunday=7*

df\_new.insert(loc=4, column='day', value=(df\_new.invoice\_date.dt.dayofweek)+1)

df\_new.insert(loc=5, column='hour', value=df\_new.invoice\_date.dt.hour)

**df\_new**.**head()**

Out[17]:

|  | invoice\_no | invoice\_date | year\_month | month | day | hour | stock\_code | description | quantity | unit\_price | total\_sales | cust\_id | country |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 536365 | 2010-12-01 08:26:00 | 201012 | 12 | 3 | 8 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 2.55 | 15.30 | 17850.0 | United Kingdom |
| 1 | 536365 | 2010-12-01 08:26:00 | 201012 | 12 | 3 | 8 | 71053 | WHITE METAL LANTERN | 6 | 3.39 | 20.34 | 17850.0 | United Kingdom |
| 2 | 536365 | 2010-12-01 08:26:00 | 201012 | 12 | 3 | 8 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 2.75 | 22.00 | 17850.0 | United Kingdom |
| 3 | 536365 | 2010-12-01 08:26:00 | 201012 | 12 | 3 | 8 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 3.39 | 20.34 | 17850.0 | United Kingdom |
| 4 | 536365 | 2010-12-01 08:26:00 | 201012 | 12 | 3 | 8 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 3.39 | 20.34 | 17850.0 | United Kingdom |

In [18]:

**#garbage collector interface**

import gc

**#basic date and time**

import datetime

pip install pandas-profiling

*#diagnose data sparsity*

import missingno as msno

**#to generate profile reports from a pandas DataFrame**

pip install pandas-profiling

#Add Month, Day and Hour columns to the dataset

data\_data\_new.insert(loc=6, column ='year',value=data\_data\_new['invoice\_date'].map(lambda x: 100\*x.year + x.month))

value=data\_data\_new.invoice\_date.dt.month)

# +1 to make Monday=1.....until Sunday=7

DATABASE CREATION FOR STORING INFORMATION :

Designing a database for an e-commerce store involves creating tables to store information about products, categories, customers, orders, and more. Below is a simplified example using SQL syntax. Depending on your specific needs, you may need to expand or modify the schema accordingly.

-- Create a table for product categories

CREATE TABLE Categories (

CategoryID INT PRIMARY KEY,

CategoryName VARCHAR(255) NOT NULL

);

-- Create a table for products

CREATE TABLE Products (

ProductID INT PRIMARY KEY,

ProductName VARCHAR(255) NOT NULL,

CategoryID INT,

Price DECIMAL(10, 2) NOT NULL,

StockQuantity INT NOT NULL,

Description TEXT,

ImageURL VARCHAR(255),

FOREIGN KEY (CategoryID) REFERENCES Categories(CategoryID)

);

-- Create a table for customers

CREATE TABLE Customers (

CustomerID INT PRIMARY KEY,

FirstName VARCHAR(255) NOT NULL,

LastName VARCHAR(255) NOT NULL,

Email VARCHAR(255) NOT NULL,

Phone VARCHAR(20),

Address VARCHAR(255),

City VARCHAR(255),

Country VARCHAR(255),

PostalCode VARCHAR(20)

);

-- Create a table for orders

CREATE TABLE Orders (

OrderID INT PRIMARY KEY,

CustomerID INT,

OrderDate DATE NOT NULL,

TotalAmount DECIMAL(10, 2) NOT NULL,

Status VARCHAR(50) NOT NULL,

FOREIGN KEY (CustomerID) REFERENCES Customers(CustomerID)

);

-- Create a table for order details

CREATE TABLE OrderDetails (

OrderDetailID INT PRIMARY KEY,

OrderID INT,

ProductID INT,

Quantity INT NOT NULL,

Price DECIMAL(10, 2) NOT NULL,

FOREIGN KEY (OrderID) REFERENCES Orders(OrderID),

FOREIGN KEY (ProductID) REFERENCES Products(ProductID)

);

Explanation:

* The Categories table stores information about product categories.
* The Products table contains details about individual products, including the category they belong to.
* The Customers table stores customer information.
* The Orders table tracks orders placed by customers.
* The OrderDetails table contains details about the products included in each order.

This is a basic schema, and you may need to expand it based on additional features such as user authentication, reviews, ratings, and more. Also, consider indexing certain columns to improve query performance, and implement data validation and constraints to maintain data integrity.

SAMPLE ECOMMERCE WEBSITE DESIGN :

