DataSpark: Illuminating Insights for Global Electronics

1. Introduction

Problem Statement:

As part of Global Electronics' data analytics team, you are tasked with conducting a comprehensive Exploratory Data Analysis (EDA) to uncover valuable insights from the company's data. Your goal is to provide actionable recommendations that can enhance customer satisfaction, optimize operations, and drive overall business growth. Global Electronics, a leading retailer of consumer electronics, has provided you with several datasets containing information about their customers, products, sales, stores, and currency exchange rates. The company seeks to leverage this data to better understand their business and identify areas for improvement..

Business Use Cases:
By analyzing Global Electronics' customer, product, sales, and store data, we aim to identify key insights that will enhance marketing strategies, optimize inventory management, and improve sales forecasting. This will help tailor marketing campaigns, develop better products, plan effective promotions, and decide on store expansions and optimizations. Additionally, understanding the impact of currency exchange rates on sales will allow for better international pricing strategies. Overall, these insights will help Global Electronics increase customer satisfaction and drive business growth.

Data Set:

Link: DataSet

Project Overview:

The DataSpark project conducts Exploratory Data Analysis for Global Electronics, leveraging customer, sales, product, store, and currency data to uncover insights. The goal is to enhance customer satisfaction, optimize operations, and drive business growth through actionable recommendations.

Objectives:

- To Perform Data Cleaning and Preparation.
- To store the data in an SQL database.
- To Connect SQL to Power BI, import the data, and create interactive dashboards.
- Develop 10 SQL Queries to Formulate and execute 10 SQL queries to extract key insights from the data.

2. Tools and Technologies

- **Power BI:** For creating visualizations, dashboards, and analyzing data trends.
- **Jupyter Notebook:** For performing exploratory data analysis (EDA) and data preprocessing.
- **Pandas:** For data manipulation and cleaning in Python.
- **Matplotlib/Seaborn:** For creating visualizations in Python during the EDA process.
- **DAX (Data Analysis Expressions):** For creating custom measures and calculations in Power BI.
- **CSV Files:** For storing the cleaned datasets used in analysis.

3. Project Setup

Prerequisites:

- Python installed on your machine.
- Required Python libraries: numpy, seaborn, streamlit, pandas,matplotlib,plotly,.
- For advanced visualizations, dashboards, and data analysis download PowerBI.

Data Preprocessing and Exploratory Data Analysis

The code efficiently handles data preprocessing and EDA for the Customers, Exchange Rates, and Products datasets. It focuses on ensuring data cleanliness (handling missing values, duplicates, encoding issues), transforming columns to the correct data types, and visualizing important patterns for further analysis.

1. Data Import and Encoding Detection:

```
import chardet
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from datetime import datetime
```

Purpose:

- chardet: Automatically detects the file encoding, which is necessary for loading files with non-UTF-8 encoding.
- pandas: Handles data manipulation and loading CSV files into DataFrames.
- seaborn and matplotlib: Used for visualizing the data.
- datetime: Used for handling date and time operations, particularly for calculating customer age.

2. File Encoding Detection

```
files = [
    os.path.join(csv_folder_path, 'Customers.csv'),
    os.path.join(csv_folder_path, 'Sales.csv'),
    os.path.join(csv_folder_path, 'Products.csv'),
    os.path.join(csv_folder_path, 'Stores.csv'),
    os.path.join(csv_folder_path, 'Exchange_Rates.csv'
]

# Detect encoding of each file
for f in files:
    with open(f, 'rb') as file:
        res = chardet.detect(file.read())
        print(f"Detected encoding for {file} : {res['encoding']}")
```

Purpose:

- Reads the dataset files in binary mode to detect encoding using chardet. This ensures that the files are read correctly without any errors.

3.Function to calculate age and age_range

```
def get_age(date):
    today = datetime.now()
    return(today.year - date.year - ((today.month, today.day) < (date.month, date.day)))

def get_agerange(age):
    if age < 10:
        return 'Under 10'
    elif age < 20:
        return '10-19'
    elif age < 30:
        return '20-29'
    elif age < 40:
        return '30-39'
    elif age < 50:
        return '40-49'
    elif age < 60:
        return '50-59'
    else:
        return '60 and above'
```

4.Data Preprocessing

```
 Load Customer data onto Pandas dataframe
customers = pd.read_csv(r'.\Untitled Folder\Customers.csv', encoding='windows-1252')

print(customers.info(),"\n")
print(customers.head(),"\n")
print(customers.describe(include='all'),"\n")

# Checking for missing values
customers[customers.isna().any(axis=1)]
# Replacing null values in State Code for Naploli with its state code NA
customers.fillna('NA',inplace=True)

# Converting Birthday to Date format
customers['Birthday'] = pd.to_datetime(customers['Birthday'], format='%m/%d/%Y', errors='coerce')

# Removing spaces in column names for MySQL
customers.columns = customers.columns.str.replace(' ', '')

# Checking for Duplicates - No Duplicates found
customers.duplicated().sum()

# Calculating age and age_range for analysis/reporting purposes
```

```
customers['Age'] = customers['Birthday'].apply(get_age)
customers['AgeRange'] = customers['Age'].apply(get_agerange).astype('category')

# Renaming columns to have unique names
customers.rename(columns={
    'City'      : 'Cust_City',
    'StateCode' : 'Cust_StateCode',
    'State'     : 'Cust_State',
    'ZipCode'   : 'Cust_ZipCode',
    'Country'   : 'Cust_Country',
    'Continent' : 'Cust_Continent'
}, inplace=True)

customers.to_csv(r'.\Untitled Folder\Cleaned_Customers.csv', index=False)
print("Customers data cleaning completed ....")
```

5.Exploratory Data Analysis (EDA)

```
customers = pd.read_csv(r'.\Untitled Folder\Cleaned_Customers.csv')
customers.fillna('NA',inplace=True)
customers['Birthday'] = pd.to_datetime(customers['Birthday'], errors='coerce')

print("EDA on Cleaned Customer Data ..... \n")
print(customers.info(),"\n")
print(customers.head(),"\n")
print(customers.describe(include='all'),"\n")
print(customers.isnull().sum(),"\n")

plt.figure(figsize=(14, 10))

# 1. Gender Distribution
plt.subplot(2, 2, 1)
sns.countplot(x='Gender', data=customers)
plt.title('Customer Gender Distribution')
plt.xlabel('Gender')
plt.ylabel('Count')

# 2. Top 10 Cities (City Distribution)
plt.subplot(2, 2, 2)
sns.countplot(y='Cust_City', data=customers, order=customers['Cust_City'].value_counts().index[:10])
plt.title('Top 10 Cities by Customer Count')
plt.xlabel('Count')
plt.ylabel('City')

# 3. State Distribution
plt.subplot(2, 2, 3)
sns.countplot(y='Cust_State', data=customers, order=customers['Cust_State'].value_counts().index)
plt.title('State Distribution by Customer Count')
plt.xlabel('Count')
```

```
plt.ylabel('State')

# 4. Country Distribution
plt.subplot(2, 2, 4)
sns.countplot(y='Cust_Country', data=customers, order=customers['Cust_Country'].value_counts().index)
plt.title('Country Distribution by Customer Count')
plt.xlabel('Count')
plt.ylabel('Country')

plt.tight_layout()
plt.show()

# 5. Age by Gender
plt.figure(figsize=(8, 6))
sns.boxplot(x='Gender', y='Age', data=customers)
plt.title('Age Distribution by Gender')
plt.xlabel('Gender')
plt.ylabel('Age')
plt.show()

# 6. Age by State (Top 10 States by Customer Count)
plt.figure(figsize=(12, 8))
sns.boxplot(y='Cust_State', x='Age', data=customers,
order=customers['Cust_State'].value_counts().index[:10])
plt.title('Age Distribution by State')
plt.xlabel('Age')
plt.ylabel('State')
plt.show()
```

6.Merging DataSets

```
df1 = pd.merge(customers, sales, on='CustomerKey', how='left')
df2 = pd.merge(df1, products, on='ProductKey', how='left')
df3 = pd.merge(df2, stores, on='StoreKey', how='left')
final_merged = pd.merge(df3, exchange, left_on=['OrderDate', 'CurrencyCode'], right_on=['Date', 'Currency'])

final_merged.drop(columns=['Date', 'Currency'], inplace=True)

final_merged.info()
```

Function to Categorize Order Frequency by Customer

```
def categorize_frequency(count):
    if count >= 1 and count <= 4:
        return 'Occasional'
    elif count >= 5 and count <= 10:
        return 'Moderate'
    elif count > 10:
        return 'Frequent'
    else:
        return 'Unknown'
```

Function to Categorize Order Frequency by Customer

```
def categorize_frequency(count):
    if count >= 1 and count <= 4:
        return 'Occasional'
    elif count >= 5 and count <= 10:
        return 'Moderate'
    elif count > 10:
        return 'Frequent'
    else:
        return 'Unknown'
```

7.EDA To Business Questions

Customer purchasing behavior across age groups and geographical locations

The gender distribution appears to be evenly balanced with male and female customers represented more or less equally. This suggests that the retailer's products and marketing strategies may be appealing to both genders. The customer base is predominantly composed of individuals aged 60 and above, and the youngest individuals aged between 20 and 29, represents a significantly smaller portion. The remaining age groups, spanning from 30 to 59 years old, exhibit a relatively uniform distribution. This distribution suggests a diverse customer base with a substantial portion of older individuals, potentially indicating a mature market segment.The majority of customers are from North America followed by Europe and Australia. Toronto is leading amongst other cities along with other cities in the US, followed by the ones in Canada including Montreal, Calgary and Atlanta. All of these cities located in Nourth America, confirms the previous observation.

```
plt.figure(figsize=(16, 12))

# 1. Customers By AgeGroup and Gender
gender_count = final_merged.groupby(['AgeRange', 'Gender']).size().reset_index(name='Count')

plt.subplot(2, 2, 1)
sns.barplot(data=gender_count, x='AgeRange', y='Count', hue='Gender')
plt.title('Customers By AgeGroup and Gender')
plt.xlabel('Age Group')
plt.ylabel('Number of Customers')

# 2. Customers By AgeGroup and Continent
cust_count = final_merged.groupby(['AgeRange', 'Cust_Continent']).size().reset_index(name='Count')

plt.subplot(2, 2, 2)
sns.barplot(data=cust_count, x='AgeRange', y='Count', hue='Cust_Continent')
plt.title('Customers By Age Group and Continent')
plt.xlabel('Age Group')
plt.ylabel('Number of Customers')

# 3. Customer By Gender and Country
gender_count = final_merged.groupby(['Gender', 'Cust_Country']).size().reset_index(name='Count')
plt.subplot(2, 2, 3)
sns.barplot(data=gender_count, x='Cust_Country', y='Count', hue='Gender')
plt.title('Customers By Gender and Country')
```

```
plt.xlabel('Country')
plt.ylabel('Number of Customers')
plt.xticks(rotation=45)

# 4. Top 10 Customer Cities
city_count = final_merged['Cust_City'].value_counts()
top_10_cities = city_count.head(10)

plt.subplot(2, 2, 4)
top_10_cities.plot(kind='bar')
plt.xlabel('City')
plt.ylabel('Number of Customers')
plt.title('Top 10 Customer Cities')
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()
```

Correlations between customer demographics (age, gender) and product preferences or purchase frequency

Both males and females within the same age group show similar purchasing patterns across different product categories. This indicates a gender-neutral trend in purchasing behavior, suggesting that gender may not be a significant factor influencing product preferences among customers.Regardless of age and gender, customers demonstrate consistent preferences for various product categories. The 'normalized purchase frequency heatmap' shows same correlation. This consistency suggests that product preferences are not strongly correlated with demographic factors such as age or gender but indicating universal appeal of certain products across diverse customer segments.Analysis of the 'normalized purchase frequency heatmap' shows highest purchase of purchase computers, followed by cell phones, music, movies, and audio books. These product categories appear as the most favored among customers of all age groups and genders, indicating their widespread appeal and demand among the customer base. However, the purchase frequency heatmap highlights a trend among customers aged 60 and above, who demonstrate higher purchasing frequency compared to other age groups. This observation can be attributed to the larger size of this age group, suggesting that while their purchasing frequency may be higher, it may not indicate a stronger preference for specific products over other groups.

```
cust_orderfreq = final_merged.groupby(['AgeRange', 'Gender', 'Category']).size().reset_index(name='order_freq')
cust_orderfreq_df = cust_orderfreq.reset_index()

pivot_df = cust_orderfreq_df.pivot(index=['AgeRange', 'Gender'], columns='Category', values='order_freq')
pivot_df_normalized = pivot_df.div(pivot_df.sum(axis=1), axis=0) * 100

fig, axes = plt.subplots(1, 2, figsize=(16, 12))
for i, dataframe in enumerate([pivot_df, pivot_df_normalized]):
    sns.heatmap(dataframe, cmap='YlGnBu', annot=True, fmt='.02f', linewidths=0.5, ax=axes[i])
    if i == 0:
        axes[i].set_title("Purchase Frequency by Age and Gender")
    else:
        axes[i].set_title("Normalized Purchase Frequency by Age and Gender")
    axes[i].set_xlabel("Product Category")
    axes[i].set_ylabel("Demographics (Age Group, Gender)")
    axes[i].tick_params(axis='x', rotation=90)
```

Correlation betn Store Size and age of the Store on Revenue

```
df_store = final_merged[(final_merged['SquareMeters'] != 0) & (final_merged['StoreAge'] != 0)]

corr = df_store[['SquareMeters', 'StoreAge', 'RevenueUSD']].corr()
print(corr)

plt.figure(figsize=(4, 2))
sns.heatmap(corr, annot=True, cmap='YlGnBu', fmt='.02f', linewidths=0.5)


plt.title('Revenue By Store Size and Age')
plt.show()
```

Top selling products, brands or categories across different regions and demographics

Leading Brands: Contos dominates followed by Wide World Importers, Southridge Video and Adventure Works. Contoso's lead indicates strong brand recognition and customer preference. Marketing and product quality likely contribute to this brand's success**.**

Top Category and Subcategory: Computers are the most popular category followed by Cell Phones. Bluetooth Headphones is a subctegory in demand. This indicates strong preference for technology and electronics among Customers. Next in the list is Music, Movies and Audio Books which suggests a substantial market for entertainment content**.**

Top Products: The highest-selling products are desktop computers from "Adventure Works Desktop" and "WWI Desktop" series.The focus on desktop computers suggests a high demand for these products among customers, potentially due to specific features or competitive pricing.

```
plt.figure(figsize=(23, 14))

# 1. Top Selling Brand
brand_counts = final_merged['Brand'].value_counts()
top_brands = brand_counts.nlargest(10)

plt.subplot(2, 2, 1)
top_brands.plot(kind='bar')
plt.xlabel('Brand')
plt.ylabel('Quantity Sold')
plt.title('Top 10 Brands')
plt.xticks(rotation=90)

# 2. Top Selling Categories
category_counts = final_merged['Category'].value_counts()
top_category = category_counts.nlargest(10)

plt.subplot(2, 2, 2)
top_category.plot(kind='bar')
plt.xlabel('Category')
plt.ylabel('Quantity Sold')
plt.title('Top 10 Categories')
plt.xticks(rotation=90)
```

```
# 3. Top Selling Sub-categories
scategory_counts = final_merged['Subcategory'].value_counts()
top_scategory = scategory_counts.nlargest(10)

plt.subplot(2, 2, 3)
top_scategory.plot(kind='bar')
plt.xlabel('Subcategory')
plt.ylabel('Quantity Sold')
plt.title('Top 10 Subcategories')
plt.xticks(rotation=90)

# 4. Top Selling Products
product_counts = final_merged['ProductName'].value_counts()
top_product = product_counts.nlargest(10)

plt.subplot(2, 2, 4)
top_product.plot(kind='bar')
plt.xlabel('Product')
plt.ylabel('Quantity Sold')
plt.title('Top 10 Products')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```

Overall Store Profit by Day, Week, Month, Year

The sales trend shows a decline in April every year. However, it gradually increased, peaking in Dec continuing onto Jan and Feb in the subsequent yearOverall, there is an upward trend since 2017 until it reached its peak in 2020. Post April 2020, it is visible that sales has not seen its expected upward trend

```
import pandas as pd

# Assuming your DataFrame is named final_merged and it has 'OrderDate' and 'RevenueUSD' columns
# Make sure 'OrderDate' is in datetime format
final_merged['OrderDate'] = pd.to_datetime(final_merged['OrderDate'])

# Resample by day and calculate total revenue per day
daily_revenue = final_merged.groupby('OrderDate')['RevenueUSD'].sum().reset_index()

# Calculate the average revenue per day
avg_revenue = daily_revenue['RevenueUSD'].median()  # Median to handle potential outliers
print(f'Average Revenue Per Day: ${avg_revenue:.2f}')

# Resample by week and calculate total revenue per week
weekly_revenue = final_merged.resample('W', on='OrderDate')['RevenueUSD'].sum().reset_index()

# Calculate the average revenue per week
avg_weekly_revenue = weekly_revenue['RevenueUSD'].median()
print(f'Average Revenue Per Week: ${avg_weekly_revenue:.2f}')

# Resample by month and calculate total revenue per month
monthly_revenue = final_merged.resample('M', on='OrderDate')['RevenueUSD'].sum().reset_index()
```

```
# Calculate the average revenue per month
avg_monthly_revenue = monthly_revenue['RevenueUSD'].median()
print(f'Average Revenue Per Month: ${avg_monthly_revenue:.2f}')

# Resample by year and calculate total revenue per year
yearly_revenue = final_merged.resample('Y', on='OrderDate')['RevenueUSD'].sum().reset_index()

# Calculate the average revenue per year
avg_yearly_revenue = yearly_revenue['RevenueUSD'].mean()
print(f'Average Revenue Per Year: ${avg_yearly_revenue:.2f}')
plt.figure(figsize=(12, 6))

plt.plot(monthly_revenue['OrderDate'], monthly_revenue['RevenueUSD'], linestyle='-', color='r')
plt.title('Monthly Revenue')
plt.xlabel('Order Date')
plt.ylabel('Revenue(USD)')
plt.grid(True)
plt.xticks(rotation=45)

# Extract month from the OrderDate
final_merged['Month'] = final_merged['OrderDate'].dt.month

# Filter data for April (Month 4)
for i in range(1,13):
    month_data = final_merged[final_merged['Month'] == i]

    # Calculate total revenue for the month
    total_revenue_month = month_data['RevenueUSD'].sum()

    print(f'Total Revenue for Month {i}: ${total_revenue_month:,.2f}')
```

8.Connecting With SQL And Storing In Database

Installing Packages

```
!pip install mysql-connector-python
!pip install SQLAlchemy
```

Connecting with Sql

```
import sqlalchemy
from sqlalchemy import create_engine, text

# Database connection credentials
host = "localhost"  # Usually "localhost" if running locally
user = "root"
password = "silviya123"
database = "GlobalElectronics"

# Create SQLAlchemy Engine for MySQL connection
```

```
engine = create_engine(f'mysql+mysqlconnector://{user}:{password}@{host}/{database}')

# Example query to test the connection
with engine.connect() as connection:
    result = connection.execute(text("SHOW TABLES"))
    for row in result:
        print(row)
```

Create Customers Table

```
create_customer = """
CREATE TABLE IF NOT EXISTS customers (
    CustomerKey INT PRIMARY KEY NOT NULL,
    Gender VARCHAR(10),
    Name VARCHAR(255),
    City VARCHAR(255),
    StateCode VARCHAR(255),
    State VARCHAR(255),
    ZipCode VARCHAR(20),
    Country VARCHAR(100),
    Continent VARCHAR(100),
    Birthday DATE,
    Age INT,
    AgeRange VARCHAR(20)
)"""

with engine.connect() as connection:
    connection.execute(text(create_customer))

# Push the Customer data into Mysql
customers.to_sql('customers', con=engine, if_exists='replace', index=False)
```

Create Exchange Rates Table

```
create_exchange_rate = """
CREATE TABLE IF NOT EXISTS ExchangeRates(
    Date DATE,
    Currency VARCHAR(3),
    Exchange FLOAT,
    PRIMARY KEY (Date, Currency)
)"""

with engine.connect() as connection:
    connection.execute(text(create_exchange_rate))

# Push the ExchangeRates data into Mysql
exchange.to_sql('exchangerates', con=engine, if_exists='replace', index=False)
```

Create Products Table

```
create_products = """
CREATE TABLE IF NOT EXISTS Products (
    ProductKey INT PRIMARY KEY NOT NULL,
    ProductName VARCHAR(255),
    Brand VARCHAR(100),
```

```
    Color VARCHAR(50),
    UnitCostUSD FLOAT,
    UnitPriceUSD FLOAT,
    SubcategoryKey INT,
    Subcategory VARCHAR(255),
    CategoryKey INT,
    Category VARCHAR(255)
)
"""
with engine.connect() as connection:
    connection.execute(text(create_products))

# Push the Products data into Mysql
products.to_sql('products', con=engine, if_exists='replace', index=False)
```

Create Sales Table

```
create_sales = """
CREATE TABLE IF NOT EXISTS Sales (
    OrderNumber INT,
    LineItem INT,
    OrderDate DATE,
    DeliveryDate DATE,
    CustomerKey INT,
    StoreKey INT,
    ProductKey INT,
    Quantity INT,
    CurrencyCode VARCHAR(10),
    PRIMARY KEY (OrderNumber, LineItem)
)
"""
with engine.connect() as connection:
    connection.execute(text(create_sales))

# Push the Sales data into Mysql
sales.to_sql('sales', con=engine, if_exists='replace', index=False)
```

Create Sales Table

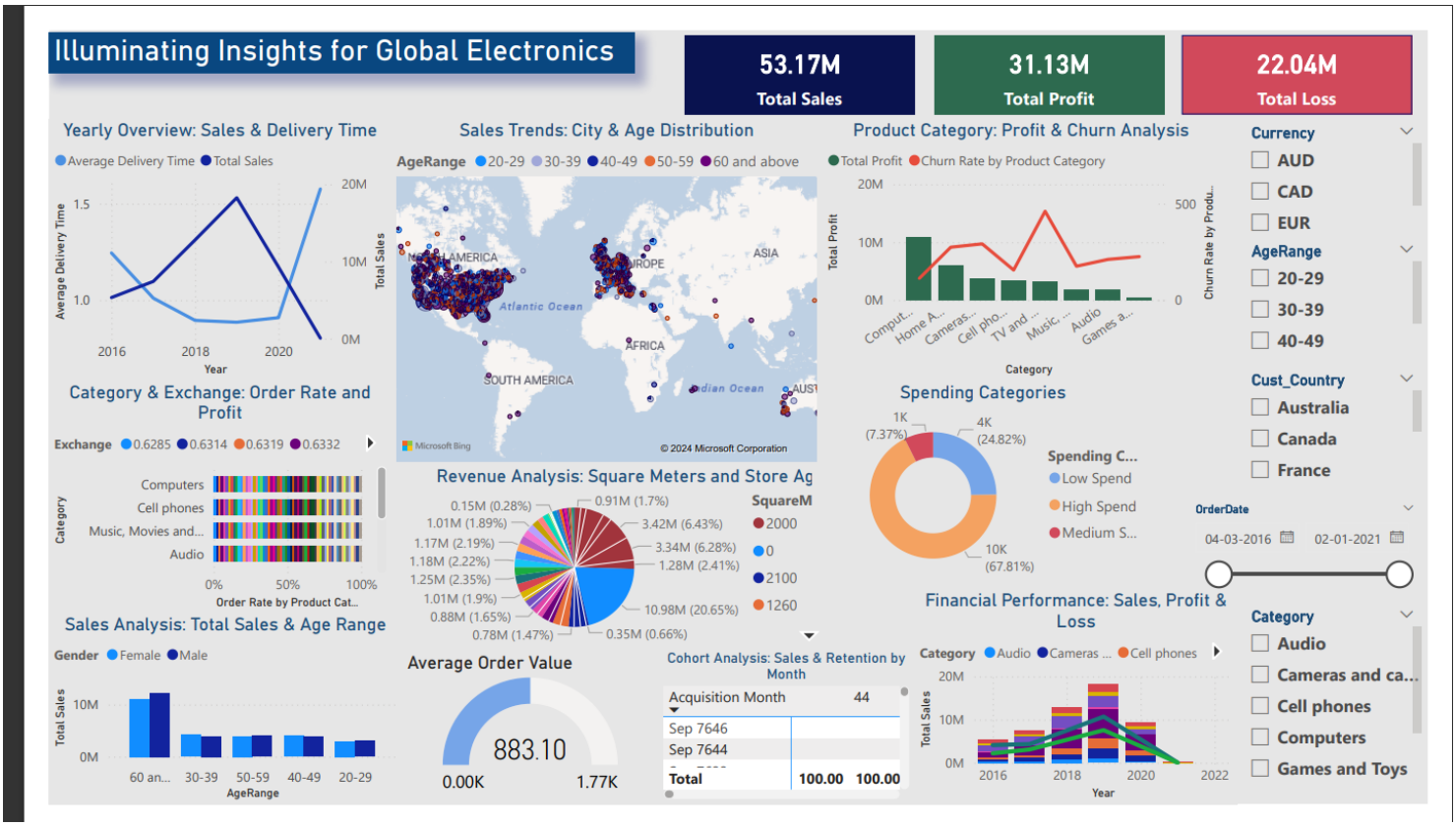
```
create_sales = """
CREATE TABLE IF NOT EXISTS Sales (
    OrderNumber INT,
    LineItem INT,
    OrderDate DATE,
    DeliveryDate DATE,
    CustomerKey INT,
    StoreKey INT,
    ProductKey INT,
    Quantity INT,
    CurrencyCode VARCHAR(10),
    PRIMARY KEY (OrderNumber, LineItem)
)
"""
with engine.connect() as connection:
```

```
    connection.execute(text(create_sales))

# Push the Sales data into Mysql
sales.to_sql('sales', con=engine, if_exists='replace', index=False)
```

9.Visualization Using PowerBI:

Screenshots :



Results:

Upon completion of the project, learners are expected to deliver a comprehensive Exploratory Data Analysis (EDA) report for Global Electronics, featuring clean and integrated datasets, in-depth insights into customer demographics, purchasing behaviors, product performance, store operations, and currency impact on sales. The report will include visually compelling visualizations and actionable recommendations tailored to enhance marketing strategies, optimize inventory management, improve sales forecasting, guide product development, and inform store expansion and operational decisions. This analysis will empower Global Electronics to increase customer satisfaction, maximize revenue, and drive overall business growth**.**

Project Deliverables:

- Data Cleaning and Preparation: To Ensure all datasets are clean, integrated, and ready for analysis.
- Exploratory Data Analysis (EDA): To Perform EDA to uncover trends, patterns, and insights.
- Visualizations: To Create visualizations to effectively communicate key findings.
- Report: To Summarize the analysis and provide actionable recommendations for Global Electronics.

Project Evaluation Metrics

1.Data Quality

Assess completeness, accuracy, consistency, and timeliness of the datasets to ensure reliable insights.

2,Database Design
Evaluate the schema for normalization, scalability, and indexing to optimize data integrity and retrieval performance.

3.Visualization Quality:
Review the clarity, interactivity, and relevance of visualizations to ensure effective communication of insights.

4. Application Usability:
Gather user feedback on UX, accessibility, and documentation to enhance user experience and navigation within the Streamlit application.

5. Code Quality:
Ensure adherence to coding standards, readability, testing effectiveness, and proper use of version control for maintainability and collaboration.

Technical Tags

- Data Cleaning and Preprocessing
- Exploratory Data Analysis (EDA)
- Python Programming
- Data Management using SQL
- Data Visualization with Power BI
- Domain Knowledge in Retail Analytics

Conclusion

The DataSpark project effectively uncovered valuable insights in the electronics retail sector by employing data cleaning, preprocessing, and exploratory data analysis techniques. Utilizing Python and SQL for data management, along with Power BI and Tableau for visualization, the project addressed key business questions related to customer behavior and sales trends. The findings provide actionable recommendations for optimizing operations and enhancing customer satisfaction, supporting informed decision-making and strategic growth in the competitive electronics market.

Future Work

- Enhanced Data Integration
- Advanced Predictive Analytics
- Real-Time Data Processing
- Customer Feedback Loop

References

- Links to resources and documentation used in theproject.

  EDA Guide