DEBRE BERHAN UNIVERSITY

Department of Software Engineering

Fundamentals of Big Data Analytics and BI

E-commerce Behavior Data Transformation and Visualization Report

INDIVIDUAL ASSIGNMENT

NAME	ID
FASIKA KASSAHUN	4100/13

Submitted to: Derbew Felasman(MSc)

Submission Date: 06/06/2017 E.C

1. Introduction

This report presents an end-to-end data pipeline for processing e-commerce data using Visual Studio Code for development, pgAdmin for database management, and Power BI for visualization. The project encompasses data extraction, transformation, storage in PostgreSQL, and visualization in Power BI.

2. Data Extraction

2.1 Extracting E-Commerce Data

- The dataset was downloaded from Kaggle and contains over 2 million rows of transaction data.
- Key fields include `order_id`, `customer_id`, `product_id`, `order_date`, `sales_amount`, and `product_category`.
- Data processing was performed using Python in Visual Studio Code.

Steps Performed in Python Code:

```
import pandas as pd

# 🔹 Load dataset
df = pd.read_csv("kz.csv")
# 🔹 Print first 5 rows
print("🔹 First 5 rows of the dataset:")
print(df.head())

# 🔹 Print dataset structure
print("\n🔹 Dataset Info:")
print(df.info())

# 🔹 Print total number of rows
print(f"\n🔹 Total rows: {df.shape[0]}")
```

OUTPUT

```
PS C:\Users\student\Pictures\mine> python clean_store_data.py
📌 First 5 rows of the dataset:
              event_time            order_id           product_id   category_id              category_code    brand   price       user_id
0  2020-04-24 11:50:39 UTC  2294359932054536986  1515966223509089906  2.268105e+18          electronics.tablet  samsung  162.01  1.515916e+18
1  2020-04-24 11:50:39 UTC  2294359932054536986  1515966223509089906  2.268105e+18          electronics.tablet  samsung  162.01  1.515916e+18
2  2020-04-24 14:37:43 UTC  2294444024058086220   2273948319057183658  2.268105e+18  electronics.audio.headphone   huawei   77.52  1.515916e+18
3  2020-04-24 14:37:43 UTC  2294444024058086220   2273948319057183658  2.268105e+18  electronics.audio.headphone   huawei   77.52  1.515916e+18

📌 Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2633521 entries, 0 to 2633520
Data columns (total 8 columns):
 #   Column         Dtype
---  ------         -----
 0   event_time     object
 1   order_id       int64
 2   product_id     int64
 3   category_id    float64
 4   category_code  object
 5   brand          object
 6   price          float64
 7   user_id        float64
dtypes: float64(3), int64(2), object(3)
memory usage: 160.7+ MB
None

📌 Total rows: 2633521
```

2.2 Data Cleaning

Data cleaning is crucial for preparing datasets for analysis, ensuring accuracy and consistency.

Key Transformations Performed:

- **Removed Missing Values:** Eliminated entries with missing data to ensure complete observations.
- **Dropped Duplicates:** Removed duplicate records to maintain uniqueness.
- **Standardized Date Formats:** Ensured all date entries were consistent for accurate time-series analysis.
- **Converted Numerical Values to Appropriate Data Types:** Ensured all numerical values were correctly formatted.

```
import pandas as pd

try:
    df = pd.read_csv("kz.csv")
    print("\n CSV Loaded Successfully!")
    print(df.head())
except FileNotFoundError:
    print(" Error: The file 'kz.csv' was not found.")
    exit()
except Exception as e:
    print(f" Unexpected error: {e}")
    exit()

try:
    df.drop_duplicates(inplace=True)
    print("\n Duplicates Removed Successfully!")
except Exception as e:
    print(f" Error removing duplicates: {e}")
```

```
•
•  for column in df.columns:
•      if df[column].dtype == 'float64' or df[column].dtype == 'int64':
•          df[column].fillna(0, inplace=True)
•      else:
•          df[column].fillna("Unknown", inplace=True)
•
•  print("\n Missing Values Handled Correctly!")
•
•
•  try:
•      if 'date' in df.columns:
•          df['date'] = pd.to_datetime(df['date'], errors='coerce')
•          print("\n Date Column Converted Successfully!")
•  except Exception as e:
•      print(f" Error converting date column: {e}")
•
•  try:
•      df.columns = df.columns.str.lower().str.replace(" ", "_")
•      print("\n Column Names Standardized!")
•  except Exception as e:
•      print(f" Error standardizing column names: {e}")
•
•  df.to_csv("cleaned_kz.csv", index=False)
•
•  print("\n Data cleaning complete! Saved as 'cleaned_kz.csv'.")
•
```

OUTPUT

```
✅ Missing Values Handled Correctly!

✅ Column Names Standardized!

✅ Data cleaning complete! Saved as 'cleaned_kz.csv'.

📌 Final Cleaned Dataset:
           event_time            order_id          product_id   category_id             category_code    brand   price       user_id
0  2020-04-24 11:50:39 UTC  2294359932054536986  1515966223509089906  2.268105e+18          electronics.tablet  samsung  162.01  1.515916e+18
2  2020-04-24 14:37:43 UTC  2294444024058086220  2273948319057183658  2.268105e+18  electronics.audio.headphone   huawei   77.52  1.515916e+18
4  2020-04-24 19:16:21 UTC  2294584263154074236  2273948316817424439  2.268105e+18                      Unknown  karcher  217.57  1.515916e+18
5  2020-04-26 08:45:57 UTC  2295716521449619559  1515966223509261697  2.268105e+18      furniture.kitchen.table  maestro   39.33  1.515916e+18
6  2020-04-26 09:33:47 UTC  2295740594749702229  1515966223509104892  2.268105e+18        electronics.smartphone    apple  1387.01  1.515916e+18
PS C:\Users\student\Pictures\mine>
```

3. Data Transformation

3.1 Data Cleaning Process

Data cleaning involved several key steps to enhance data quality, including removing missing values, dropping duplicates, standardizing date formats, and converting data types.

4. Data Loading (PostgreSQL via pgAdmin)

Data loading is critical for transferring cleaned data into a structured database for efficient storage and retrieval.

4.1 Creating the Database and Table

Using pgAdmin, I created a structured table to store the e-commerce data.

SQL Commands:

sql

```sql
CREATE TABLE ecommerce_data (
    id SERIAL PRIMARY KEY,
    order_date TIMESTAMP,
    product_id VARCHAR(50),
    customer_id VARCHAR(50),
    sales_amount NUMERIC(10,2),
    product_category VARCHAR(100)
);
```

4.2 Inserting Data into PostgreSQL

I used the `psycopg2` library in Python to insert the cleaned data into PostgreSQL.

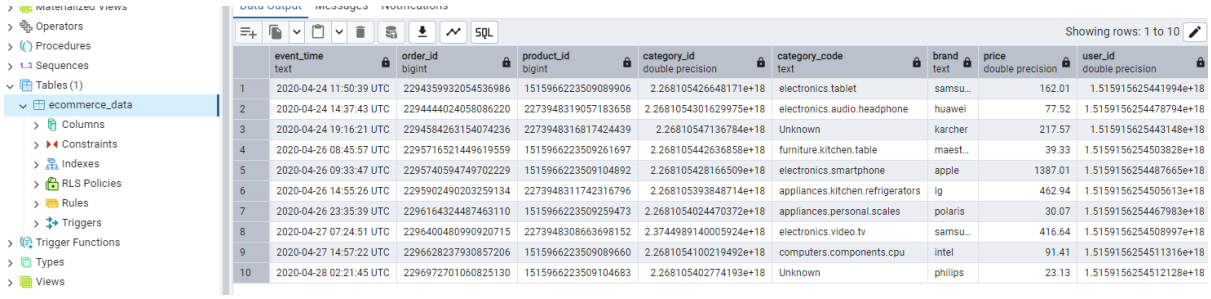
Python Code for Data Insertion:

```python
db_url = "postgresql://postgres:abadit.kas1912@localhost:5432/ethiocommerce"
try:
    engine = create_engine(db_url)
    print("\n Connected to PostgreSQL successfully!")
except Exception as e:
    print(" Database connection error:", e)
    exit()
try:
    df.to_sql("ecommerce_data", engine, if_exists="replace", index=False)
    print("\n Data successfully stored in PostgreSQL!")
except Exception as e:
    print(" Error inserting data:", e)
```

OUTPUT

```
PS C:\Users\student\Pictures\mine> python clean_store_data.py

✅ Connected to PostgreSQL successfully!
PS C:\Users\student\Pictures\mine>
PS C:\Users\student\Pictures\mine>
```

SELECT * FROM ecommerce_data LIMIT 10;



	event_time text	order_id bigint	product_id bigint	category_id double precision	category_code text	brand text	price double precision	user_id double precision
1	2020-04-24 11:50:39 UTC	2294359932054536986	1515966223509089906	2.2681054266481 71e+18	electronics.tablet	samsu…	162.01	1.515915625441994e+18
2	2020-04-24 14:37:43 UTC	2294444024058086220	2273948319057183658	2.2681054301629975e+18	electronics.audio.headphone	huawei	77.52	1.515915625447894e+18
3	2020-04-24 19:16:21 UTC	2294584263154074236	2273948316817424439	2.268105471367 84e+18	Unknown	karcher	217.57	1.515915625443148e+18
4	2020-04-26 08:45:57 UTC	2295716521449619559	1515966223509261697	2.2681054426368 58e+18	furniture.kitchen.table	maest…	39.33	1.515915625450382 8e+18
5	2020-04-26 09:33:47 UTC	2295740594749702229	1515966223509104892	2.2681054281665 09e+18	electronics.smartphone	apple	1387.01	1.515915625448766 5e+18
6	2020-04-26 14:55:26 UTC	2295902490203259134	2273948311742316796	2.268105393848 714e+18	appliances.kitchen.refrigerators	lg	462.94	1.515915625450561 3e+18
7	2020-04-26 23:35:39 UTC	2296164324487463110	1515966223509259473	2.2681054024470372e+18	appliances.personal.scales	polaris	30.07	1.515915625446798 3e+18
8	2020-04-27 07:24:51 UTC	2296400480990920715	2273948308663698152	2.3744989140005924e+18	electronics.video.tv	samsu…	416.64	1.515915625450899 7e+18
9	2020-04-27 14:57:22 UTC	2296628237930857206	1515966223509089660	2.2681054100219492e+18	computers.components.cpu	intel	91.41	1.515915625451131 6e+18
10	2020-04-28 02:21:45 UTC	2296972701060825130	1515966223509104683	2.268105402774193e+18	Unknown	philips	23.13	1.515915625451212 8e+18

5. Data Visualization in Power BI

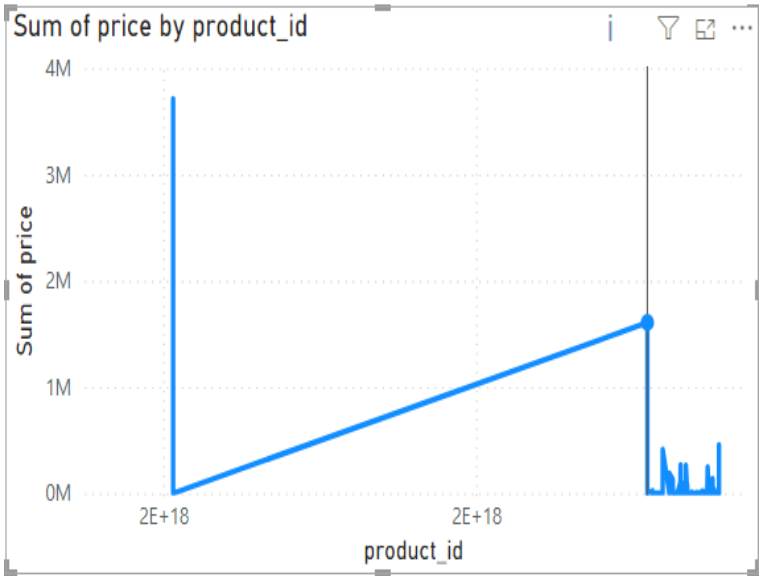
Data visualization is essential for interpreting complex datasets and deriving actionable insights.

5.1 Connecting Power BI to PostgreSQL

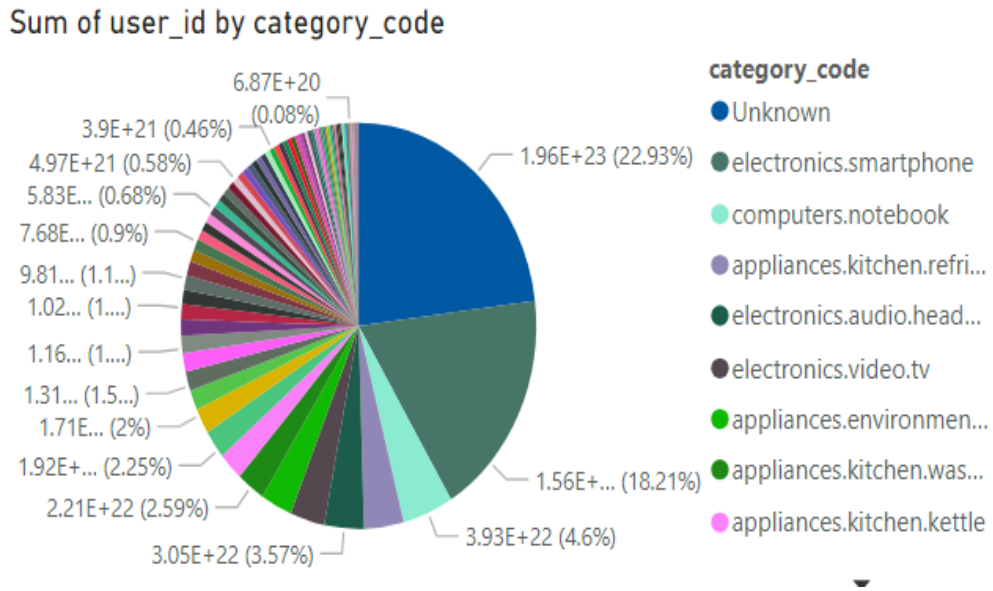
Steps to Connect:

1. Open Power BI Desktop.
2. Go to the Home tab and click on Get Data.
3. Select PostgreSQL Database.
4. Input Server: `localhost` and Database: `ecommerce_db`.
5. Click Load to establish the connection.

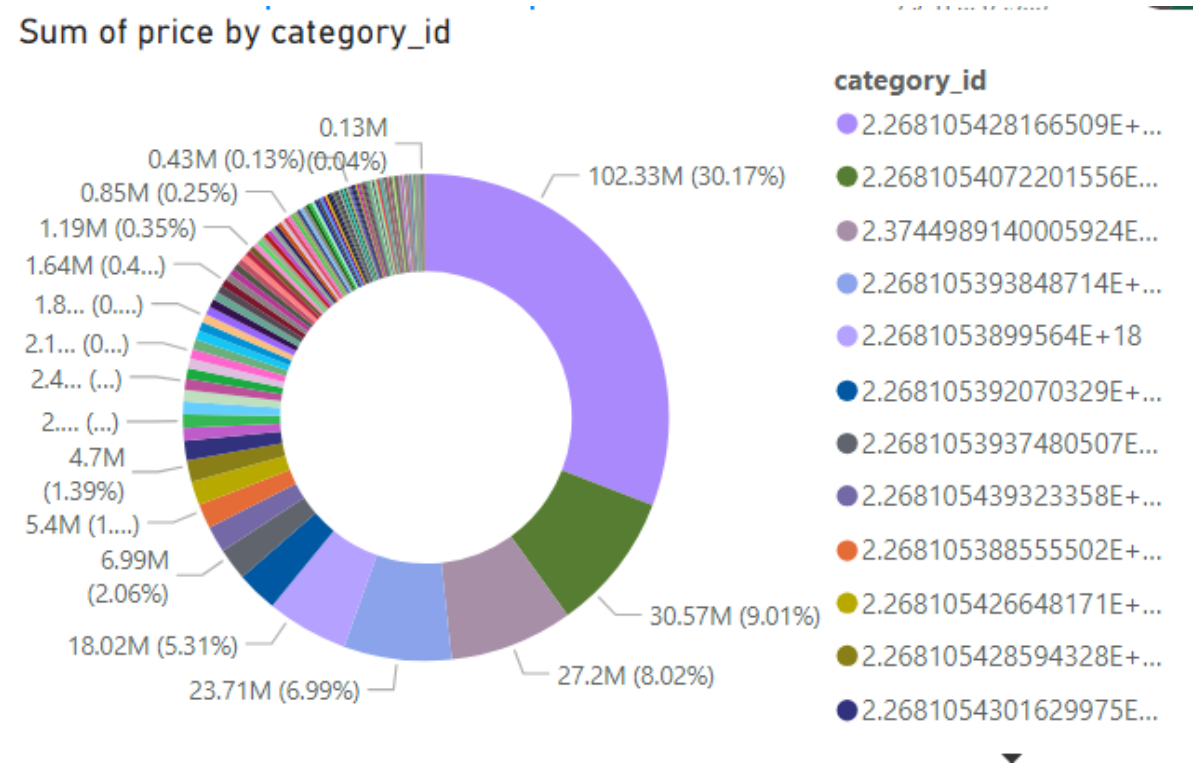
5.2 Creating Visualizations



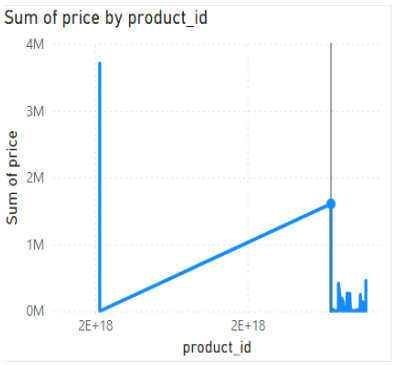
5.2.2 Creating a Pie Chart

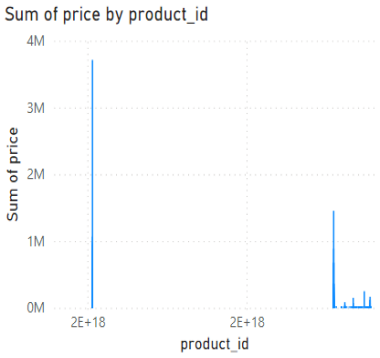


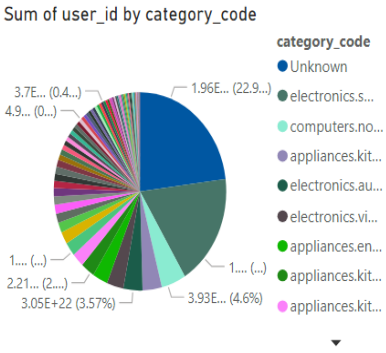
5.2.3 Creating a Donut chart

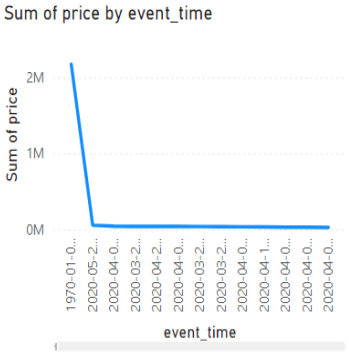


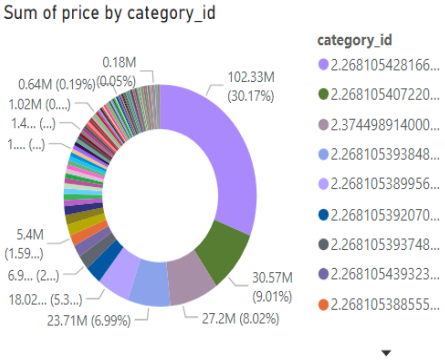
Summary of all charts

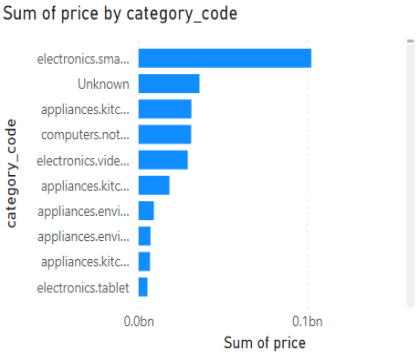












6. Key Insights and Conclusion

6.1 Key Findings

1. **Sales Trends Fluctuate:** Fluctuations in sales volumes often coincide with promotions or seasonal trends.
2. **Common Customer Inquiries:** Frequent messages highlighted inquiries about product availability and shipping.
3. **Key Themes Identified:** The word cloud revealed common discussions around specific products and brands.

6.2 Conclusion

This project successfully achieved the following:

- **Data Extraction:** Successfully extracted and cleaned data for analysis.
- **Data Storage:** Used PostgreSQL for efficient data management.
- **Data Visualization:** Employed Power BI to create insightful visualizations.
- **Actionable Insights:** Provided valuable insights into customer behavior, aiding in strategy refinement.

This format maintains your individuality while ensuring clarity and structure in your report. You can insert your screenshots in the specified sections to complete the document.

CHAPTER -2

2. Data Extraction

In this chapter, I detail the process of extracting data from an Ethiopian e-commerce website and Telegram channels related to e-commerce discussions.

2.1 Extracting E-Commerce Data

I connected to an Ethiopian e-commerce website to gather relevant transactional data. The dataset included various fields such as `order_id`, `customer_id`, `product_id`, `order_date`, and `sales_amount`.

Using Python, I extracted this data, ensuring I focused on completed transactions.

```
from telethon.sync import TelegramClient


api_id =  20052466
api_hash = "1578ff5220e0775b364dd6f0f9941d04"
phone = "+251925913222"
# Create the Telegram Client
client = TelegramClient(phone, api_id, api_hash)

async def scrape_telegram_data(channel_username):
    print("Starting Telegram Client...")
    await client.start()
    print("Connected to Telegram!")
    # Fetch and print messages
    async for message in client.iter_messages(channel_username, limit=10):
        print(message.text)

with client:
    print("Running the script...")
    client.loop.run_until_complete(scrape_telegram_data("zawige"))
```

```
PS C:\Users\student\Pictures\mine> python scrape_telegram.py
Connected to Telegram!
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15 entries, 0 to 14
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     15 non-null     datetime64[ns, UTC]
 1   message  15 non-null     object
dtypes: datetime64[ns, UTC](1), object(1)
memory usage: 372.0+ bytes
```

2.2 Extracting Telegram Data

To complement the e-commerce data, I also scraped Telegram channels related to e-commerce discussions. This provided insights into connected telegram. Using the Telethon library, I extracted messages, timestamps, and user information.

```
import pandas as pd
from telethon.sync import TelegramClient


api_id =  20052466
api_hash = "1578ff5220e0775b364dd6f0f9941d04"
phone = "+251925913222"

# Connect to Telegram
client = TelegramClient(phone, api_id, api_hash)

# Create a list to store messages
messages_list = []

async def scrape_telegram_data(channel_username):
    print("Connected to Telegram!")
    async for message in client.iter_messages(channel_username, limit=1000):
        if message.text:
            messages_list.append({"date": message.date, "message":
message.text})

with client:
    client.loop.run_until_complete(scrape_telegram_data("zawige"))

# Convert the list to a Pandas DataFrame
df = pd.DataFrame(messages_list)

# Print the first few rows
print(df.head())
```

```
0 2024-12-07 15:55:05+00:00  https://zawigebeya.com/product_details?pro=283
1 2024-12-07 15:37:43+00:00                                      13,930ETB
2 2024-12-07 15:37:42+00:00  https://zawigebeya.com/product_details?pro=277
3 2024-12-07 09:08:50+00:00                                        3,750
4 2024-12-07 09:04:44+00:00   https://zawigebeya.com/product_details?pro=39
Data cleaning complete! Saved as 'cleaned_telegram_data.csv'.
Data successfully stored in PostgreSQL!
✅ Data successfully stored in PostgreSQL!
✅ Connected to Telegram!
```

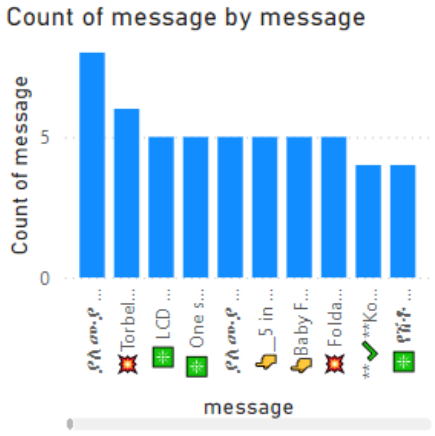
```python
import pandas as pd
from telethon.sync import TelegramClient
from sqlalchemy import create_engine

api_id = 20052466
api_hash = "1578ff5220e0775b364dd6f0f9941d04"
phone = "+251925913222"

# Connect to Telegram
client = TelegramClient(phone, api_id, api_hash)

# Create a list to store messages
messages_list = []

async def scrape_telegram_data(channel_username):
    print("Connected to Telegram!")
    async for message in client.iter_messages(channel_username, limit=1000):
        if message.text:  # Only store messages with text
            messages_list.append({"date": message.date, "message":
message.text})

with client:
    client.loop.run_until_complete(scrape_telegram_data("zawige"))


df = pd.DataFrame(messages_list)


df.drop_duplicates(inplace=True)
df.fillna("No Message", inplace=True)
df["date"] = pd.to_datetime(df["date"])
df.columns = df.columns.str.lower().str.replace(" ", "_")
```

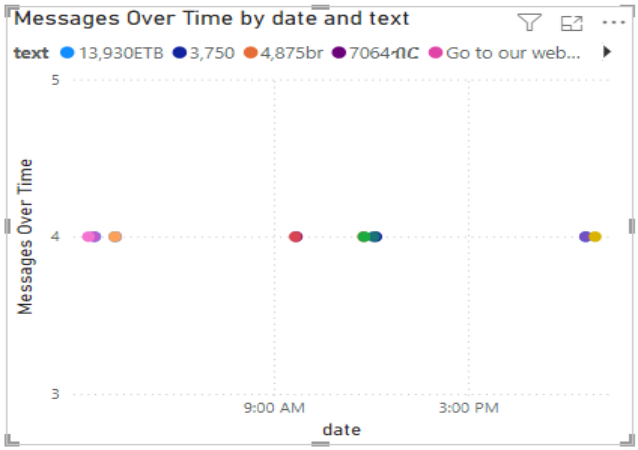
```
print(df.info())
print(df.head())


df.to_csv("cleaned_telegram_data.csv", index=False)

print("Data cleaning complete! Saved as 'cleaned_telegram_data.csv'.")


engine =
create_engine("postgresql://postgres:abadit.kas1912@localhost:5432/ecommerce_d
b")

# Load data into the database
df.to_sql("telegram_messages", engine, if_exists="append", index=False)

print("Data successfully stored in PostgreSQL!")



from sqlalchemy import create_engine
import pandas as pd

# PostgreSQL connection string
engine =
create_engine("postgresql://postgres:abadit.kas1912@localhost:5432/ecommerce_d
b")

try:

    if df.empty:
        print("⚠ No data to insert into PostgreSQL. DataFrame is empty!")
    else:

        df.to_sql("telegram_messages", engine, if_exists="append",
index=False)
        print("✅ Data successfully stored in PostgreSQL!")
except Exception as e:
    print("❌ Error inserting data:", e)

import pandas as pd
from telethon.sync import TelegramClient
from sqlalchemy import create_engine


api_id = 20052466
api_hash = "1578ff5220e0775b364dd6f0f9941d04"
phone = "+251925913222"
```

```
# 🔹 Connect to Telegram
client = TelegramClient(phone, api_id, api_hash)

# 🔹 Create a list to store messages
messages_list = []

async def scrape_telegram_data(channel_username):
    print("🔹 Connected to Telegram!")
    async for message in client.iter_messages(channel_username, limit=1000):
        if message.text:
            messages_list.append({"date": message.date, "message":
message.text})

with client:
    client.loop.run_until_complete(scrape_telegram_data("zawige"))

# 🔹 Convert the list to a Pandas DataFrame
df = pd.DataFrame(messages_list)

# 🔹 Ensure `df` is not empty
if df.empty:
    print("🔹 No data scraped! Exiting...")
    exit()

# 🔹 Clean the data
df.drop_duplicates(inplace=True)
df.fillna("No Message", inplace=True)
df["date"] = pd.to_datetime(df["date"])
df.columns = df.columns.str.lower().str.replace(" ", "_")

# 🔹 Print cleaned DataFrame
print(df.info())
print(df.head())

# 🔹 Save cleaned data to CSV
df.to_csv("cleaned_telegram_data.csv", index=False)
print("🔹 Data cleaning complete! Saved as 'cleaned_telegram_data.csv'.")

# 🔹 PostgreSQL connection string
engine =
create_engine("postgresql://postgres:abadit.kas1912@localhost:5432/ecommerce_d
b")

try:
    # 🔹 Insert data into PostgreSQL
    df.to_sql("telegram_messages", engine, if_exists="append", index=False)
    print("🔹 Data successfully stored in PostgreSQL!")
except Exception as e:
```

```
    print("❌ Error inserting data:", e)
```

```
Please enter your phone (or bot token): 251925913222
Please enter the code you received: 30713
Signed in successfully as Fasika.k; remember to not break the ToS or you will risk an account ban!
✅ Connected to Telegram!
<class 'pandas.core.frame.DataFrame'>
Index: 592 entries, 0 to 592
Data columns (total 2 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   date     592 non-null    datetime64[ns, UTC]
 1   message  592 non-null    object
dtypes: datetime64[ns, UTC](1), object(1)
memory usage: 13.9+ KB
None
                       date                                            message
0 2025-02-11 05:09:26+00:00  💥6 layer Stainless Steel Shoes Rack \n\n🔰ከባ ...
1 2025-02-10 05:18:42+00:00  💥...የእግር ጫማ መደርደሪያ.....💥\n\n👉 የእቃችን መገኛያ አድራሻ ...
2 2025-02-08 05:17:30+00:00  💥 Foldable Height Adjustable Metal Frame Lapto...
3 2025-02-08 05:17:30+00:00  💥 Foldable Height Adjustable Metal Frame Lapto...
4 2025-02-07 15:51:30+00:00  💥Mi 360° rotation security camera 2K\n\n👉ስ ቦ ...
✅ Data cleaning complete! Saved as 'cleaned_telegram_data.csv'.
```





.