Given the dataset structure from an e-commerce platform, give me step by step to analyze it. The columns in the dataset: 1. Visitor ID (Unique identifier for visitors) 2. Visit Time (Timestamp of the visit) 3. Product SKU & Product Name (Product identifiers and descriptions) 4. Transaction Count (Number of transactions for each product) 5. City & Province (Geographical data related to transactions) My goal is to identify top-selling products, detect anomalies, and find the most profitable locations. What preprocessing steps should I take?

**ChatGPT said:**

To analyze your e-commerce dataset effectively and achieve your goals, follow these **step-by-step preprocessing steps**:

**1. Load & Explore the Data**

* Load the dataset using pandas.
* Display the first few rows (df.head()) and check column data types (df.info()).
* Identify missing values (df.isnull().sum()).
* Check for duplicate records (df.duplicated().sum()).

**2. Handle Missing Values**

* If Transaction Count has missing values, fill them with 0 (assuming no transaction occurred).
* If City or Province is missing, consider filling it with "Unknown" or using external data to infer locations.
* Drop rows with missing Visitor ID, Product SKU, or Visit Time if these are critical.

**3. Convert Data Types**

* Convert Visit Time to a datetime format:

python

CopyEdit

df['Visit Time'] = pd.to\_datetime(df['Visit Time'])

* Convert Transaction Count to integer:

python

CopyEdit

df['Transaction Count'] = df['Transaction Count'].astype(int)

**4. Handle Duplicates**

* If the dataset contains duplicate records (same Visitor ID, Visit Time, Product SKU), remove them.

python

CopyEdit

df.drop\_duplicates(inplace=True)

**5. Aggregate Data for Analysis**

* **Top-Selling Products:**  
  Aggregate transactions per product to get total sales:

python

CopyEdit

top\_products = df.groupby(['Product SKU', 'Product Name'])['Transaction Count'].sum().reset\_index()

top\_products = top\_products.sort\_values(by='Transaction Count', ascending=False)

* **Most Profitable Locations:**  
  Group by City and Province:

python

CopyEdit

top\_locations = df.groupby(['City', 'Province'])['Transaction Count'].sum().reset\_index()

top\_locations = top\_locations.sort\_values(by='Transaction Count', ascending=False)

**6. Detect Anomalies in Transactions**

* **Outlier Detection in Transaction Counts**  
  Use IQR (Interquartile Range) to identify extreme sales activity:

python

CopyEdit

Q1 = df['Transaction Count'].quantile(0.25)

Q3 = df['Transaction Count'].quantile(0.75)

IQR = Q3 - Q1

threshold\_low = Q1 - 1.5 \* IQR

threshold\_high = Q3 + 1.5 \* IQR

anomalies = df[(df['Transaction Count'] < threshold\_low) | (df['Transaction Count'] > threshold\_high)]

* **Time-Based Anomalies (Spike in Sales)**  
  Check unusual transaction peaks by grouping data over time:

python

CopyEdit

df.set\_index('Visit Time').resample('D')['Transaction Count'].sum().plot()

**7. Data Enrichment (Optional)**

* Add **hourly, daily, or monthly trends**:

python

CopyEdit

df['Hour'] = df['Visit Time'].dt.hour

df['Day'] = df['Visit Time'].dt.day\_name()

df['Month'] = df['Visit Time'].dt.month

* Check if some locations have higher conversion rates than others.

**8. Final Data Preparation for Visualization**

* Export cleaned data to a CSV:

python

CopyEdit

df.to\_csv("cleaned\_ecommerce\_data.csv", index=False)

* Use Power BI, Tableau, or Python (Matplotlib/Seaborn) to visualize insights.

Would you like help with specific visualizations? 🚀

4o

**You said:**

What preprocessing steps should I apply to clean and prepare this dataset before analysis? How should I handle missing values, duplicate records, and data inconsistencies?

**ChatGPT said:**

To **clean and prepare** your dataset for analysis, follow these **preprocessing steps**:

**1. Load & Inspect Data**

First, load the dataset and inspect its structure:

python

CopyEdit

import pandas as pd

# Load dataset

df = pd.read\_csv("ecommerce\_data.csv")

# Check the first few rows

print(df.head())

# Check for missing values and data types

print(df.info())

# Check for duplicate records

print(f"Duplicate Rows: {df.duplicated().sum()}")

**2. Handle Missing Values**

**Step 2.1: Identify Missing Data**

python

CopyEdit

print(df.isnull().sum())

Check where missing values occur in columns like Transaction Count, City, and Province.

**Step 2.2: Fill or Drop Missing Values**

* **Visitor ID**: If missing, drop these rows (since a missing ID makes tracking impossible).
* **Visit Time**: If missing, drop the row (time is crucial for trend analysis).
* **Product SKU & Product Name**: If missing, drop or try to

4o

**You said:**

How can I determine the top-selling products based on total transaction revenue per day? Should I use Python or SQL? Give me the best practice!

**ChatGPT said:**

To determine the **top-selling products based on total transaction revenue per day**, you can use **either Python (pandas)** or **SQL (aggregations & ranking functions)**. The best approach depends on your dataset size and system setup:

* **Use Python** if you're working with a pandas DataFrame for quick analysis.
* **Use SQL** if your data is stored in a relational database and performance is a concern.

**Best Practice: SQL (Recommended for large datasets)**

SQL is optimized for aggregations and rankings, making it the best choice for large-scale data.  
Here’s how you can calculate daily revenue per product:

sql

CopyEdit

SELECT

DATE(Visit\_Time) AS transaction\_date,

Product\_SKU,

Product\_Name,

SUM(Transaction\_Count) AS total\_transactions

FROM ecommerce\_data

GROUP BY transaction\_date, Product\_SKU, Product\_Name

ORDER BY transaction\_date ASC, total\_transactions DESC;

**Why SQL?**

✅ Faster performance on large datasets  
✅ Easily scalable for production  
✅ Supports indexing & efficient queries

**Best Practice: Python (For Data Exploration)**

If you’re working in pandas and need a quick analysis, use:

python

CopyEdit

import pandas as pd

# Load dataset

df = pd.read\_csv("ecommerce\_data.csv")

# Convert 'Visit Time' to datetime

df['Visit Time'] = pd.to\_datetime(df['Visit Time'])

# Extract date for aggregation

df['Transaction Date'] = df['Visit Time'].dt.date

# Aggregate transaction revenue per product per day

top\_products = df.groupby(['Transaction Date', 'Product SKU', 'Product Name'])['Transaction Count'].sum().reset\_index()

# Sort by date and transaction count

top\_products = top\_products.sort\_values(by=['Transaction Date', 'Transaction Count'], ascending=[True, False])

# Display top-selling products per day

print(top\_products.head())

**Why Python?**

✅ Great for quick exploratory analysis  
✅ Easy to visualize trends with matplotlib or seaborn  
✅ Flexible for further processing

**Which One Should You Use?**

* **If working with a database → SQL (Best for production)**
* **If exploring data in a notebook → Python (Best for prototyping)**

Would you like a visualization of the daily top-selling products? 🚀

4o

**You said:**

How can I detect anomalies in sales trends, such as a sharp increase or decrease in transactions for a product? What is the best practice? Give me step-by-step using statistical methods and machine learning techniques.

2/2

**ChatGPT said:**

To **detect anomalies in sales trends**, follow these **best practices** using both **statistical methods** and **machine learning techniques**.

**1. Understanding Anomalies in Sales Trends**

Anomalies can be:

* **Sudden spikes** in sales (e.g., promotions, flash sales)
* **Sharp drops** in transactions (e.g., supply issues, seasonality)
* **Gradual deviations** from expected patterns

We will use **statistical methods (IQR, Z-score, moving averages)** and **machine learning (Isolation Forest, LSTM for time series)** to detect anomalies.

**2. Step-by-Step Approach**

**A. Data Preparation**

First, load and preprocess the dataset.

python

CopyEdit

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset

df = pd.read\_csv("ecommerce\_data.csv")

# Convert 'Visit Time' to datetime format

df['Visit Time'] = pd.to\_datetime(df['Visit Time'])

# Aggregate sales by date and product

df['Transaction Date'] = df['Visit Time'].dt.date

sales\_data = df.groupby(['Transaction Date', 'Product SKU'])['Transaction Count'].sum().reset\_index()

**B. Detecting Anomalies Using Statistical Methods**

**1. Moving Average & Standard Deviation (Rolling Window)**

We use a rolling mean to identify deviations.

python

CopyEdit

# Compute moving average and standard deviation

sales\_data['Moving\_Avg'] = sales\_data.groupby('Product SKU')['Transaction Count'].transform(lambda x: x.rolling(7, min\_periods=1).mean())

sales\_data['Std\_Dev'] = sales\_data.groupby('Product SKU')['Transaction Count'].transform(lambda x: x.rolling(7, min\_periods=1).std())

# Define an anomaly threshold (e.g., 2 standard deviations from the mean)

sales\_data['Upper\_Bound'] = sales\_data['Moving\_Avg'] + (2 \* sales\_data['Std\_Dev'])

sales\_data['Lower\_Bound'] = sales\_data['Moving\_Avg'] - (2 \* sales\_data['Std\_Dev'])

# Flag anomalies

sales\_data['Anomaly'] = (sales\_data['Transaction Count'] > sales\_data['Upper\_Bound']) | \

(sales\_data['Transaction Count'] < sales\_data['Lower\_Bound'])

# Filter anomalies

anomalies = sales\_data[sales\_data['Anomaly']]

print(anomalies)

🔹 **Best Practice**:

* Use a **7-day rolling window** for short-term trends.
* Adjust threshold (±2 std dev) based on historical trends.

**2. Z-Score Method (Detects Outliers in Sales)**

Z-score identifies values that are significantly different from the mean.

python

CopyEdit

from scipy.stats import zscore

# Compute Z-score for each product's transactions

sales\_data['Z\_Score'] = sales\_data.groupby('Product SKU')['Transaction Count'].transform(lambda x: zscore(x, nan\_policy='omit'))

# Flag anomalies (Z-score > 3 or < -3)

sales\_data['Anomaly\_Z'] = (sales\_data['Z\_Score'] > 3) | (sales\_data['Z\_Score'] < -3)

# Filter anomalies

z\_anomalies = sales\_data[sales\_data['Anomaly\_Z']]

print(z\_anomalies)

🔹 **Best Practice**:

* Use **Z-score threshold of ±3** for extreme outliers.
* Works best for **normally distributed** sales data.

**C. Detecting Anomalies Using Machine Learning**

**3. Isolation Forest (Unsupervised Anomaly Detection)**

This tree-based model isolates anomalies effectively.

python

CopyEdit

from sklearn.ensemble import IsolationForest

# Prepare features

features = sales\_data[['Transaction Count']].copy()

# Train Isolation Forest model

iso\_forest = IsolationForest(n\_estimators=100, contamination=0.05, random\_state=42)

sales\_data['Anomaly\_IF'] = iso\_forest.fit\_predict(features)

# Convert anomaly labels (-1 = anomaly, 1 = normal)

sales\_data['Anomaly\_IF'] = sales\_data['Anomaly\_IF'].apply(lambda x: True if x == -1 else False)

# Filter anomalies

if\_anomalies = sales\_data[sales\_data['Anomaly\_IF']]

print(if\_anomalies)

🔹 **Best Practice**:

* Set contamination=0.05 (5% data considered anomalies).
* Ideal for large datasets with **irregular sales trends**.

**4. LSTM (Deep Learning for Time Series)**

For **complex time-series sales forecasting**, train an **LSTM model** (Long Short-Term Memory).  
This requires:

* Formatting data into sequences
* Training an **LSTM autoencoder** to reconstruct normal sales patterns
* Detecting large deviations

Would you like a full **LSTM implementation** for anomaly detection? 🚀

4o

**You said:**

How can I determine the most profitable locations (city/province) based on total transaction revenue? What aggregation method should I use?

**ChatGPT said:**

To **determine the most profitable locations (city/province) based on total transaction revenue**, you should use **aggregation techniques** like **SUM()** in SQL or **groupby().sum()** in Python.

**1. Understanding Profitability Metrics**

You can define **profitability** based on:

1. **Total transaction revenue** = Sum of all transactions per city/province
2. **Average transaction value** = Total revenue / Number of transactions
3. **Top-performing locations over time** = Tracking trends by date

**2. SQL Approach (Best for Large Databases)**

Using **SQL**, aggregate total revenue per city and province:

sql

CopyEdit

SELECT

Province,

City,

SUM(Transaction\_Count) AS total\_transactions

FROM ecommerce\_data

GROUP BY Province, City

ORDER BY total\_transactions DESC;

🔹 **Best Practice:**

* Index City and Province for faster querying
* Use LIMIT 10 to get the **top 10 most profitable locations**

**3. Python Approach (For Data Exploration)**

Using **pandas**, aggregate and visualize total revenue by city/province:

python

CopyEdit

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset

df = pd.read\_csv("ecommerce\_data.csv")

# Aggregate transaction revenue per city and province

profit\_by\_location = df.groupby(['Province', 'City'])['Transaction Count'].sum().reset\_index()

# Sort by total transactions

profit\_by\_location = profit\_by\_location.sort\_values(by='Transaction Count', ascending=False)

# Display top 10 profitable locations

print(profit\_by\_location.head(10))

# Visualize the top locations

plt.figure(figsize=(12,6))

sns.barplot(data=profit\_by\_location.head(10), x='City', y='Transaction Count', hue='Province', palette='viridis')

plt.xticks(rotation=45)

plt.title("Top 10 Most Profitable Locations")

plt.show()

🔹 **Best Practice:**

* **Use .groupby() to sum transaction counts per city/province**
* **Sort by highest revenue to get the most profitable locations**
* **Visualize with a bar chart** for easy comparison

**4. Choosing the Best Aggregation Method**

**✅ Use SUM(Transaction\_Count)**

* Best for finding total revenue contribution per location
* Shows absolute profitability

**✅ Use AVG(Transaction\_Count)**

* Helps understand locations with **higher spending per transaction**
* Useful for analyzing **customer purchasing power**

Would you like additional insights, such as **trends over time** or **seasonality effects**? 🚀

4o

**You said:**

What data visualization techniques should I use to present insights on top-selling products, anomalies, and profitable locations? Which Python libraries are best suited for this?

**ChatGPT said:**

To present insights effectively, you should use **different visualization techniques** tailored to each objective. Here’s the best approach:

**1. Top-Selling Products**

**✅ Visualization Techniques**

* **Bar Chart**: Best for showing top-selling products.
* **Line Chart**: Shows daily sales trends