Step 1: Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import random
from datetime import datetime, timedelta
```

Step 2: Creating Dataset Creation

This Dataset represent e-commerce transactions from an online retail store. It includes the following fields:

1. CustomerID.
2. OrderID.
3. Product.
4. Quantity.
5. UnitPrice.
6. PurchaseDate.
7. Country.

```
num_records = 1000  # Adjust size as needed

products = ["Laptop", "Smartphone", "Headphones", "Smartwatch",
"Tablet", "Gaming Console", "Keyboard", "Mouse", "Monitor", "Printer"]

start_date = datetime(2022, 1, 1)
end_date = datetime(2022, 12, 31)

countries = ["USA", "Canada", "UK", "Germany", "France", "Australia",
"Japan", "India"]

df = pd.DataFrame({
    "CustomerID": np.random.randint(1000, 2000, num_records),
    "OrderID": np.arange(5000, 5000 + num_records),
    "Product": [random.choice(products) for _ in range(num_records)],
    "Quantity": np.random.randint(1, 11, num_records),
    "UnitPrice": np.random.uniform(5, 500, num_records).round(2),
    "PurchaseDate": [start_date + timedelta(days=random.randint(0,
364)) for _ in range(num_records)],
    "Country": [random.choice(countries) for _ in range(num_records)]
})

# random: Python's random module used for generating random choices
# np.random.randint: Generates random integers for CustomerID and
```

```
Quantity
# np.arange: Generates a range of OrderID values
# timedelta: Used to add random days to the start_date for
PurchaseDate
# for _ in range: List comprehension to generate random choices for
Product and Country
# random.choice: Randomly selects an item from products and countries
lists
# random.randint: Generates random integers for days to add to
start_date
# np.random.uniform: Generates random float values for UnitPrice
# .round: Rounds the UnitPrice to 2 decimal places
# .head: Displays the first 5 rows of the DataFrame

df.head()

   CustomerID  OrderID         Product  Quantity  UnitPrice
PurchaseDate  \
0        1463     5000  Gaming Console         7     141.04   2022-02-
12
1        1418     5001        Keyboard         4     110.79   2022-12-
12
2        1056     5002      Headphones         5     312.16   2022-10-
19
3        1582     5003        Keyboard         5      76.14   2022-02-
20
4        1058     5004          Laptop         5     443.32   2022-02-
24

     Country
0  Australia
1         UK
2     Canada
3      India
4      India
```

1. Data Cleaning and Preparation

```
# 1. Check for missing values
print(df.isnull().sum())

# if there are missing values
df.fillna({"Product": "Unknown", "Quantity": 1, "UnitPrice":0},
inplace=True)

CustomerID      0
OrderID         0
Product         0
```

```
Quantity        0
UnitPrice       0
PurchaseDate    0
Country         0
dtype: int64
```

```
# 2. Check for duplicates
print(df.duplicated().sum())

# if there are duplicates
df.drop_duplicates(inplace=True)
```

```
0
```

```
# Convert PurchaseDate to datetime
df["PurchaseDate"] = pd.to_datetime(df["PurchaseDate"])

# Summary of dataset verification
print(df.info())
print(df.describe())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 7 columns):
 #   Column        Non-Null Count  Dtype         
---  ------        --------------  -----         
 0   CustomerID    1000 non-null   int32         
 1   OrderID       1000 non-null   int64         
 2   Product       1000 non-null   object        
 3   Quantity      1000 non-null   int32         
 4   UnitPrice     1000 non-null   float64       
 5   PurchaseDate  1000 non-null   datetime64[ns]
 6   Country       1000 non-null   object        
dtypes: datetime64[ns](1), float64(1), int32(2), int64(1), object(2)
memory usage: 47.0+ KB
None
        CustomerID      OrderID     Quantity   UnitPrice  \
count  1000.000000  1000.000000  1000.000000  1000.00000   
mean   1487.932000  5499.500000     5.543000   255.76674   
min    1001.000000  5000.000000     1.000000     5.39000   
25%    1231.500000  5249.750000     3.000000   132.22750   
50%    1484.500000  5499.500000     6.000000   253.53000   
75%    1743.000000  5749.250000     8.000000   384.84500   
max    1999.000000  5999.000000    10.000000   499.65000   
std     294.477484   288.819436     2.898716   144.55164   

                        PurchaseDate  
count                           1000  
mean   2022-07-04 12:47:31.199999744  
min              2022-01-01 00:00:00  
25%              2022-04-05 00:00:00  
```

```
50%              2022-07-08 00:00:00
75%              2022-10-03 06:00:00
max              2022-12-31 00:00:00
std                              NaN
```

2. Filtering Data

```
# 2.1 Customer and Order Filters:
# 1. Filter all transactions made by customers from a specific country
(e.g., 'USA').
print((df[df["Country"] == "USA"]).head(10))

    CustomerID  OrderID     Product  Quantity  UnitPrice PurchaseDate
Country
8         1304     5008  Headphones         3     384.83   2022-03-04
USA
17        1286     5017       Mouse         4     487.76   2022-08-06
USA
18        1634     5018     Printer         2      26.79   2022-01-08
USA
26        1876     5026      Laptop        10     147.46   2022-07-02
USA
27        1001     5027  Headphones         3     282.72   2022-09-13
USA
32        1374     5032      Laptop        10     117.94   2022-01-20
USA
38        1117     5038  Smartwatch        10     315.70   2022-02-04
USA
40        1697     5040      Tablet         3     173.91   2022-07-14
USA
41        1648     5041     Printer         3     143.76   2022-08-08
USA
57        1253     5057    Keyboard         3      35.53   2022-10-04
USA

# 2. Extract orders where the total spend (Quantity * UnitPrice)
exceeds $500.
df["Total"] = df["Quantity"] * df["UnitPrice"]
print(df[df["Total"] > 500].head(5))

# Note: This could be done using MySQL right??

   CustomerID  OrderID         Product  Quantity  UnitPrice
PurchaseDate  \
0        1463     5000  Gaming Console         7     141.04   2022-02-
12
2        1056     5002      Headphones         5     312.16   2022-10-
19
```

```
4        1058     5004          Laptop        5      443.32   2022-02-
24
5        1033     5005        Keyboard        8      245.41   2022-12-
03
6        1905     5006          Laptop        8      298.40   2022-03-
30

     Country    Total
0  Australia   987.28
2     Canada  1560.80
4      India  2216.60
5  Australia  1963.28
6      India  2387.20

# 3. Identify customers who purchased more than 3 different products.
customer_count = df.groupby("CustomerID")["Product"].nunique()
print(customer_count[customer_count > 3].head(10))

# nunique(): Return series Number of unique values within each group

CustomerID
1253    4
1257    6
1373    4
1392    4
1408    5
1514    4
1579    4
1582    5
1937    4
1949    4
Name: Product, dtype: int64

# 2.2 Time-Based Filters:
# 1. Filter transactions that occurred in July 2022.
print(df[(df["PurchaseDate"].dt.month == 7) &
(df["PurchaseDate"].dt.year == 2022)].head())

    CustomerID  OrderID         Product  Quantity  UnitPrice
PurchaseDate  \
10        1475     5010         Printer         9     286.76   2022-
07-22
16        1642     5016          Tablet         8     463.10   2022-
07-26
20        1958     5020  Gaming Console         4      49.33   2022-
07-13
26        1876     5026          Laptop        10     147.46   2022-
07-02
40        1697     5040          Tablet         3     173.91   2022-
07-14
```

```
      Country    Total
10  Australia  2580.84
16  Australia  3704.80
20     Canada   197.32
26        USA  1474.60
40        USA   521.73

# 2. Extract orders placed during weekends.
print(df[df["PurchaseDate"].dt.dayofweek.isin([5,6])].head())

# isin(): Return a boolean Series showing whether each element in the
Series is exactly contained in the passed sequence of values.
# dt.dayofweek: The day of the week with Monday=0, Sunday=6
# dt: Accessor object for datetime like properties of the Series
values.

    CustomerID  OrderID         Product  Quantity  UnitPrice
PurchaseDate  \
0         1463     5000  Gaming Console         7     141.04   2022-
02-12
3         1582     5003        Keyboard         5      76.14   2022-
02-20
5         1033     5005        Keyboard         8     245.41   2022-
12-03
11        1022     5011         Monitor         8     136.17   2022-
01-09
17        1286     5017           Mouse         4     487.76   2022-
08-06

      Country    Total
0   Australia   987.28
3       India   380.70
5   Australia  1963.28
11  Australia  1089.36
17        USA  1951.04

# 3. Identify transactions during specific sales events, like Black
Friday or Cyber Monday.
black_friday = df[(df["PurchaseDate"].dt.month == 11) &
(df["PurchaseDate"].dt.weekday == 4) &
                        (df["PurchaseDate"].dt.day >= 23) &
(df["PurchaseDate"].dt.day <= 29)]
print(black_friday)


     CustomerID  OrderID     Product  Quantity  UnitPrice PurchaseDate
\
167        1892     5167    Keyboard         4     397.68   2022-11-25

729        1374     5729  Smartphone         2     335.67   2022-11-25
```

```
    Country    Total
167   India  1590.72
729  France   671.34
```

3. Sorting Data

```
# 1. Sort transactions by:
# - Total spend in descending order.
print(df.sort_values("Total", ascending=False).head())

     CustomerID  OrderID         Product  Quantity  UnitPrice
PurchaseDate  \
877        1017     5877      Smartwatch        10     497.35   2022-
12-01
779        1455     5779        Keyboard        10     494.19   2022-
10-11
711        1548     5711         Monitor        10     491.39   2022-
03-30
207        1583     5207         Printer        10     487.46   2022-
07-08
484        1656     5484  Gaming Console        10     487.40   2022-
03-01

       Country   Total
877  Australia  4973.5
779     France  4941.9
711    Germany  4913.9
207  Australia  4874.6
484  Australia  4874.0

# - Purchase date in ascending order.
print(df.sort_values(by="PurchaseDate", ascending=True).head())

# sort_values(): Sort by the values along either axis.

     CustomerID  OrderID     Product  Quantity  UnitPrice PurchaseDate
\
715        1462     5715  Smartphone         9     288.79   2022-01-01

871        1447     5871     Printer         6     123.01   2022-01-01

872        1153     5872      Laptop         2     113.52   2022-01-01

557        1304     5557    Keyboard         6     444.09   2022-01-02

499        1053     5499  Smartwatch         3     420.69   2022-01-03
```

```
       Country    Total
715        USA  2599.11
871         UK   738.06
872     Canada   227.04
557        USA  2664.54
499  Australia  1262.07

# - Product name alphabetically.
print(df.sort_values("Product").head())

     CustomerID  OrderID         Product  Quantity  UnitPrice
PurchaseDate  \
0          1463     5000  Gaming Console         7     141.04   2022-
02-12
236        1389     5236  Gaming Console         8     202.92   2022-
04-29
247        1074     5247  Gaming Console        10     197.79   2022-
06-21
254        1026     5254  Gaming Console         3     124.08   2022-
10-11
804        1318     5804  Gaming Console         1     246.47   2022-
01-16

       Country    Total
0    Australia   987.28
236     Canada  1623.36
247        USA  1977.90
254    Germany   372.24
804    Germany   246.47
```

Will Continue in MySQL

```
df.to_csv("ecommerce_data.csv", index=False)
```

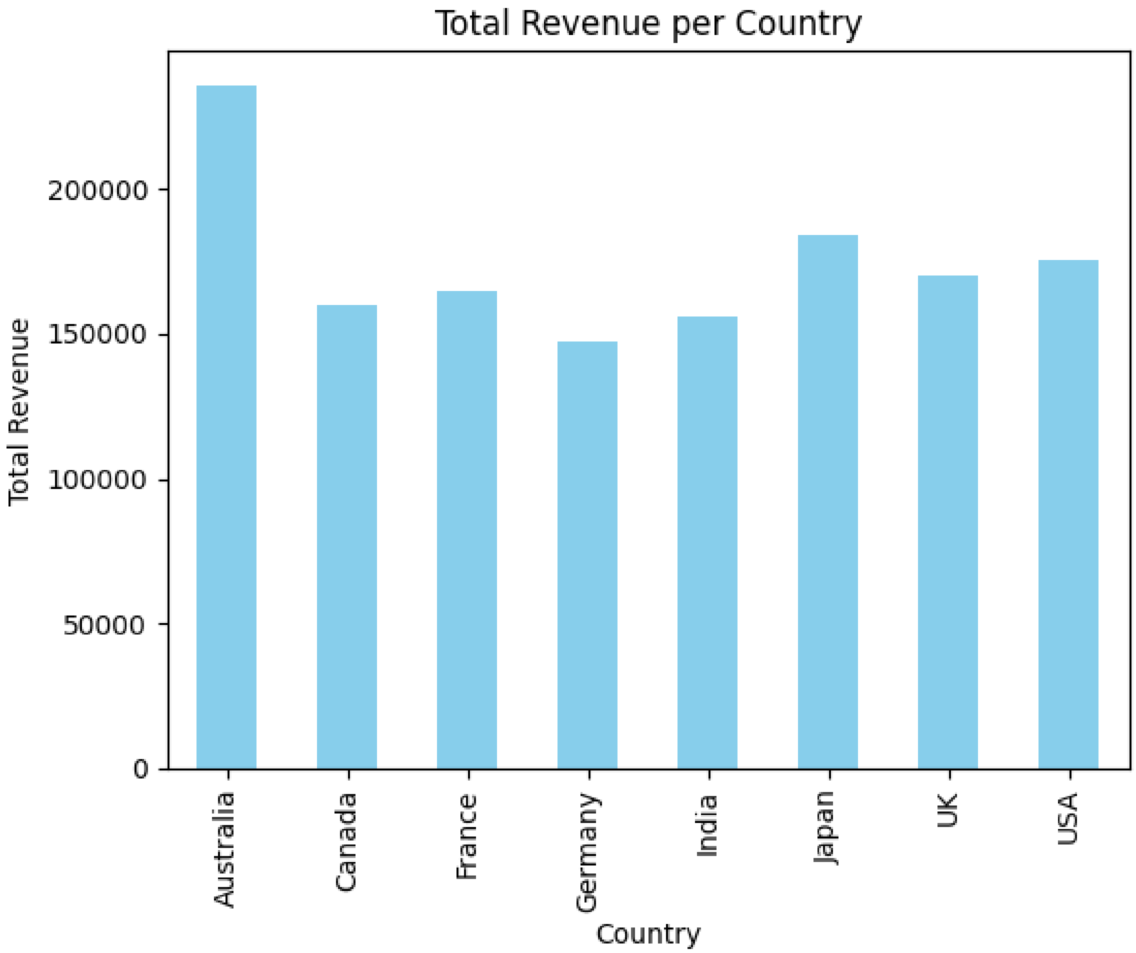
Going Back to Visualization

```
data = pd.read_csv('ecommerce_data_from_SQL.csv', delimiter=';')
data

     CustomerID  OrderID     Product  Quantity  UnitPrice  \
0          1197     5000       Mouse         2     331.04
1          1599     5001  Smartwatch         7     237.26
2          1584     5002  Smartphone         5      36.82
3          1779     5003     Monitor         4      13.17
4          1660     5004     Monitor         8     224.16
..          ...      ...         ...       ...        ...
```

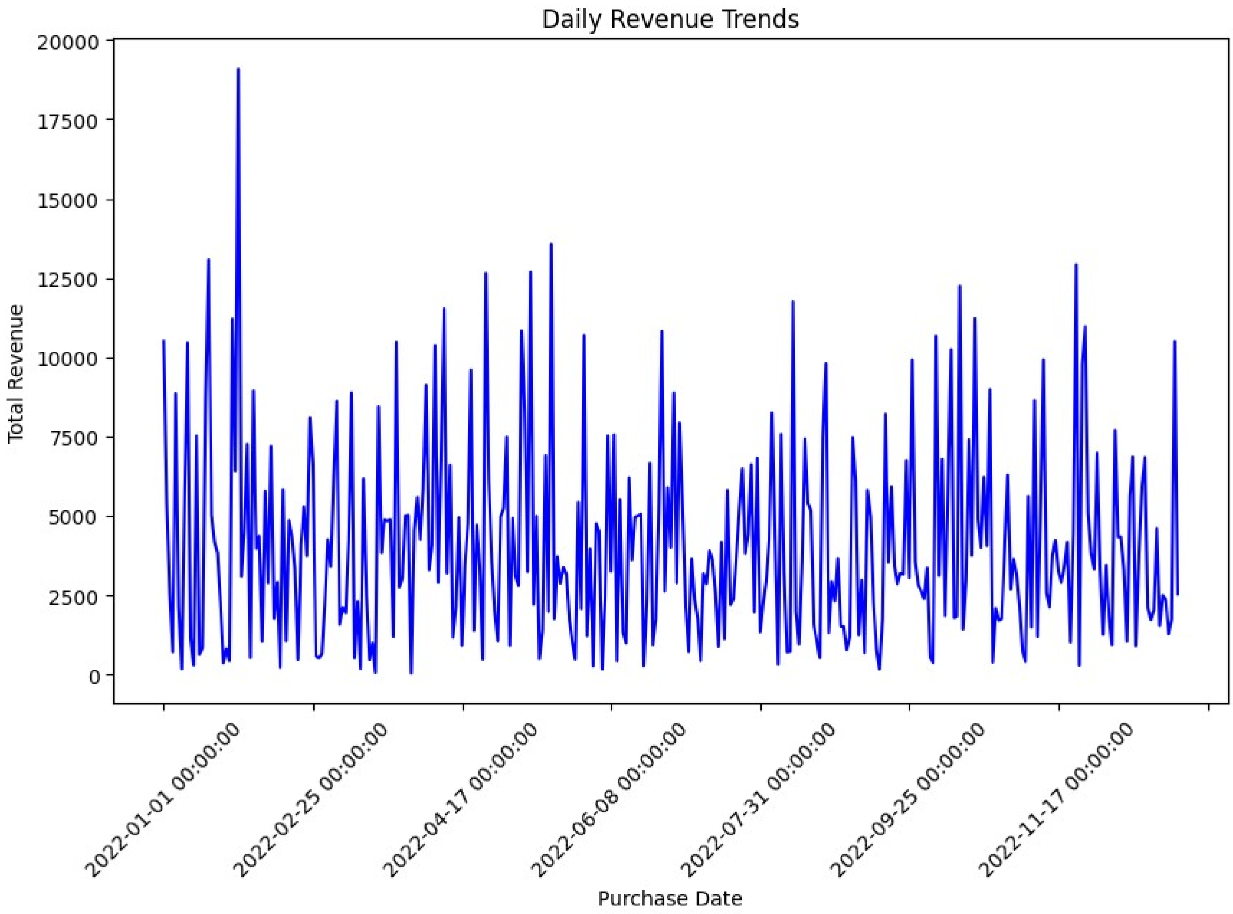
```
995        1907      5995  Smartwatch         7     215.17
996        1081      5996      Laptop         2      45.93
997        1903      5997      Laptop         3     300.15
998        1506      5998     Printer         6     268.08
999        1582      5999    Keyboard         7     471.05

            PurchaseDate  Country  TotalSpend
0    2022-10-18 00:00:00    India      662.08
1    2022-11-07 00:00:00  Germany     1660.82
2    2022-02-20 00:00:00    Japan      184.10
3    2022-03-19 00:00:00    Japan       52.68
4    2022-04-11 00:00:00   France     1793.28
..                   ...      ...         ...
995  2022-11-28 00:00:00    Japan     1506.19
996  2022-05-27 00:00:00    Japan       91.86
997  2022-11-08 00:00:00    Japan      900.45
998  2022-02-14 00:00:00      USA     1608.48
999  2022-11-24 00:00:00      USA     3297.35

[1000 rows x 8 columns]
```

6. Visualization(using matplotlib) :Extra grad

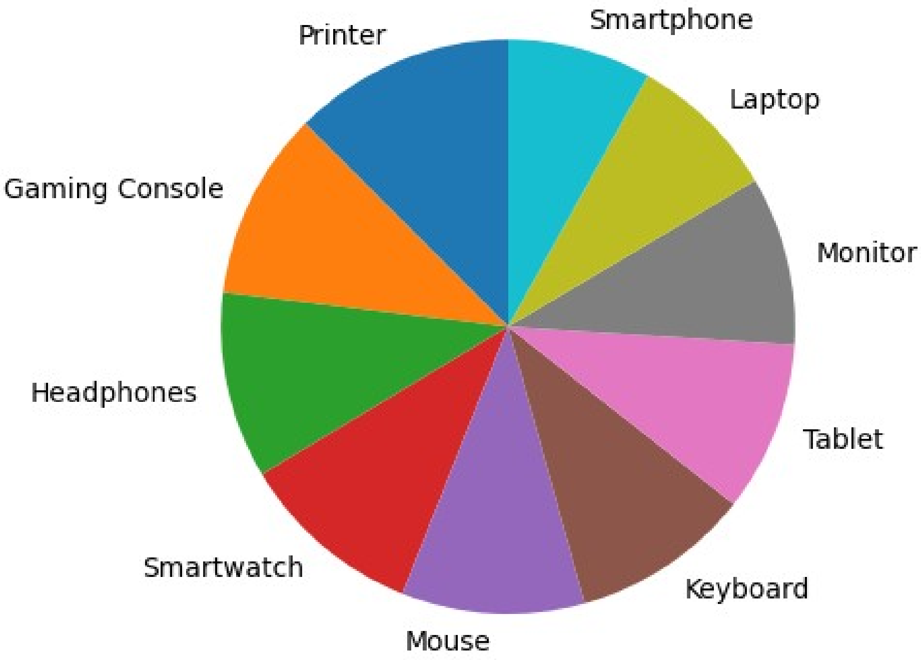
```python
# 1. Plot the total revenue per country using a bar chart.
revenue_country = data.groupby("Country")["TotalSpend"].sum()
# Plot the total revenue per country using a bar chart
revenue_country.plot(kind="bar", color="Blue")

# Add title and labels to the plot
plt.title("Total Revenue per Country")
plt.ylabel("Total Revenue")
plt.xlabel("Country")

# Display the plot
plt.show()
```



```
# 2. Visualize daily revenue trends with a line chart.
daily_revenue = data.groupby("PurchaseDate")["TotalSpend"].sum()
plt.figure(figsize=(10, 6))  # Set the figure size
daily_revenue.plot(kind="line", color="blue")
plt.title("Daily Revenue Trends")
plt.ylabel("Total Revenue")
plt.xlabel("Purchase Date")

plt.xticks(rotation=45)

plt.show()
```



```python
# 3. Use a pie chart to show the distribution of products sold.
Product_distribution = data["Product"].value_counts()
plt.pie(Product_distribution, labels = Product_distribution.index,
startangle = 90)
plt.show()
```


Printer
Smartphone
Laptop
Gaming Console
Monitor
Headphones
Tablet
Smartwatch
Keyboard
Mouse