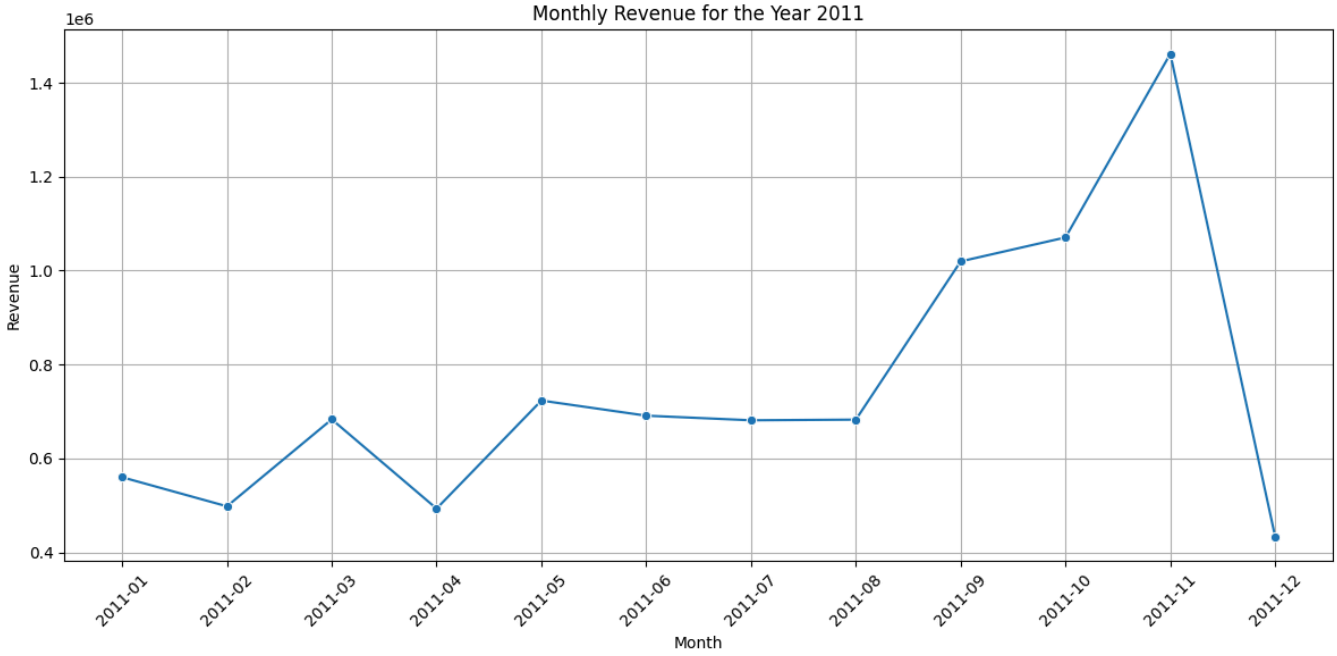
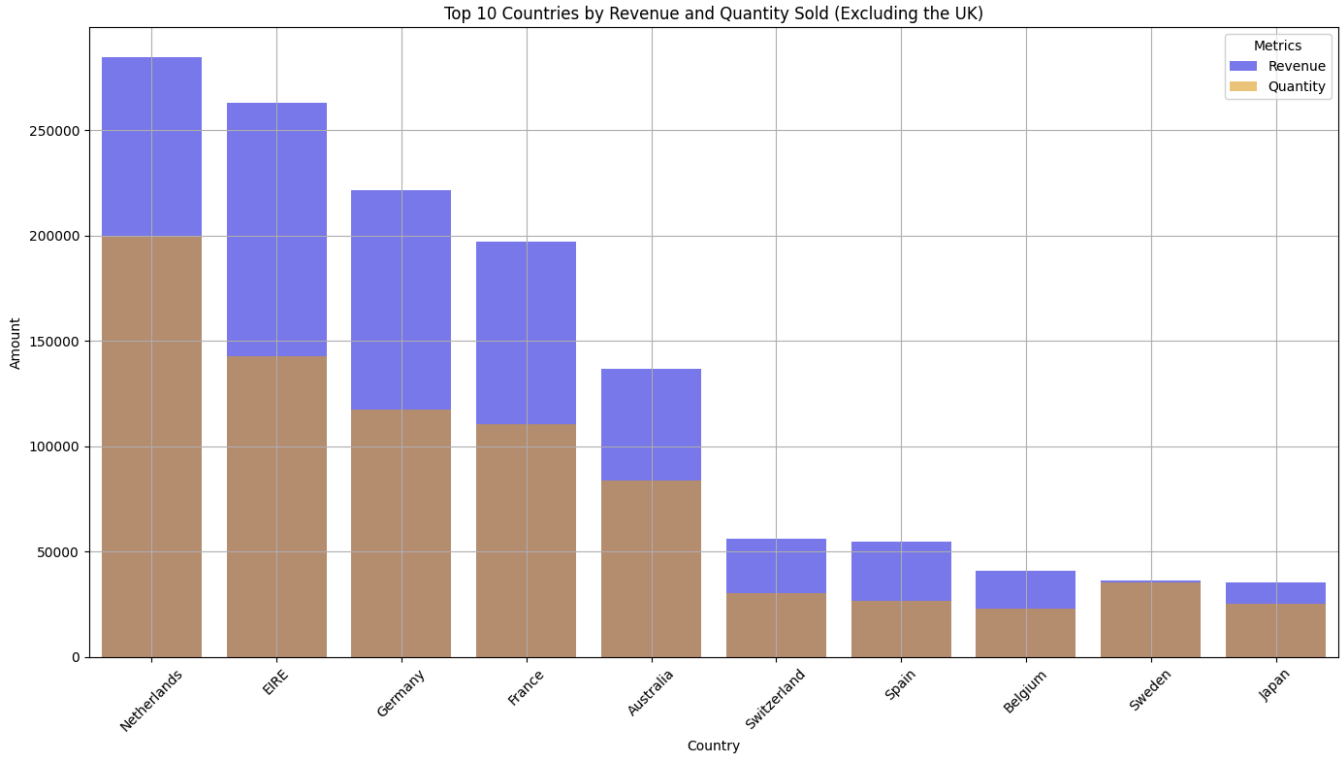
Start coding or generate with AI.

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data from Excel
df = pd.read_excel('/content/Online Retail Data Set.xlsx', engine='openpyxl')

# Convert 'InvoiceDate' to datetime format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], format='%d-%m-%Y %H:%M')

# Filter data for the year 2011 and create a copy
df_2011 = df[df['InvoiceDate'].dt.year == 2011].copy()

# Calculate revenue for each transaction
df_2011['Revenue'] = df_2011['Quantity'] * df_2011['UnitPrice']

# Extract month from 'InvoiceDate'
df_2011['Month'] = df_2011['InvoiceDate'].dt.to_period('M')

# Check data types and for any non-numeric values
print(df_2011.dtypes)
print(df_2011[['Revenue', 'Month']].head())

# Ensure 'Revenue' is numeric
df_2011['Revenue'] = pd.to_numeric(df_2011['Revenue'], errors='coerce')

# Aggregate revenue by month
monthly_revenue = df_2011.groupby('Month')['Revenue'].sum().reset_index()

# Ensure 'Month' is treated as a string or datetime
monthly_revenue['Month'] = monthly_revenue['Month'].astype(str)

# Plot the data
plt.figure(figsize=(12, 6))
sns.lineplot(data=monthly_revenue, x='Month', y='Revenue', marker='o')
plt.title('Monthly Revenue for the Year 2011')
plt.xlabel('Month')
plt.ylabel('Revenue')
plt.xticks(rotation=45)
plt.grid(True)
plt.tight_layout()

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

```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
Revenue               float64
Month                period[M]
dtype: object
       Revenue    Month
42481     19.5  2011-01
42482     10.5  2011-01
42483     10.5  2011-01
42484     10.5  2011-01
42485     12.5  2011-01
```

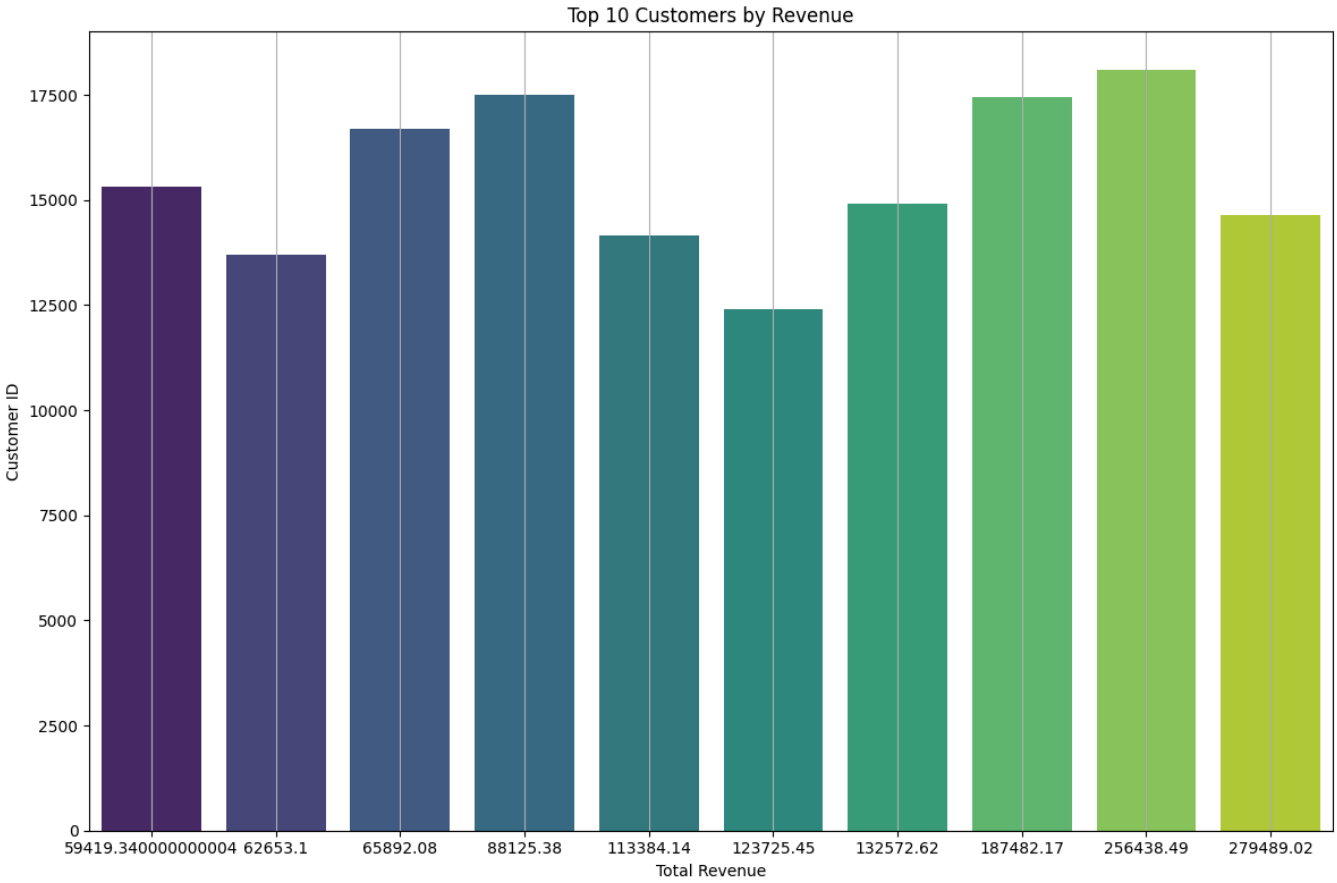


```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data from Excel
df = pd.read_excel('/content/Online Retail Data Set.xlsx', engine='openpyxl')

# Convert 'InvoiceDate' to datetime format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], format='%d-%m-%Y %H:%M')

# Calculate revenue for each transaction
df['Revenue'] = df['Quantity'] * df['UnitPrice']

# Filter out the United Kingdom
df_filtered = df[df['Country'] != 'United Kingdom']

# Aggregate revenue and quantity by country
country_summary = df_filtered.groupby('Country').agg({
    'Revenue': 'sum',
    'Quantity': 'sum'
}).reset_index()

# Sort by revenue and select top 10 countries
top_countries = country_summary.sort_values(by='Revenue', ascending=False).head(10)

# Plot the data
plt.figure(figsize=(14, 8))

# Create a bar plot for revenue
sns.barplot(data=top_countries, x='Country', y='Revenue', color='blue', label='Revenue', alpha=0.6)

# Create a secondary bar plot for quantity
sns.barplot(data=top_countries, x='Country', y='Quantity', color='orange', label='Quantity', alpha=0.6)

# Adding labels and title
plt.title('Top 10 Countries by Revenue and Quantity Sold (Excluding the UK)')
plt.xlabel('Country')
plt.ylabel('Amount')
plt.xticks(rotation=45)
plt.legend(title='Metrics')
plt.grid(True)
plt.tight_layout()

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




```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the data from Excel
df = pd.read_excel('/content/Online Retail Data Set.xlsx', engine='openpyxl')

# Convert 'InvoiceDate' to datetime format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], format='%d-%m-%Y %H:%M')

# Calculate revenue for each transaction
df['Revenue'] = df['Quantity'] * df['UnitPrice']

# Aggregate revenue by customer
customer_summary = df.groupby('CustomerID').agg({
    'Revenue': 'sum'
}).reset_index()

# Sort by revenue and select top 10 customers
top_customers = customer_summary.sort_values(by='Revenue', ascending=False).head(10)

# Plot the data
plt.figure(figsize=(12, 8))

# Create a horizontal bar chart
sns.barplot(data=top_customers, y='CustomerID', x='Revenue', palette='viridis')

plt.title('Top 10 Customers by Revenue')
plt.xlabel('Total Revenue')
plt.ylabel('Customer ID')
plt.grid(axis='x')
plt.tight_layout()

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

```
<ipython-input-13-7bec2a35cac8>:26: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `x` variable to `hue` and set `le

  sns.barplot(data=top_customers, y='CustomerID', x='Revenue', palette='viridis')
```




```
import pandas as pd
import folium
from folium.plugins import HeatMap

# Load the data from Excel
df = pd.read_excel('/content/Online Retail Data Set.xlsx', engine='openpyxl')

# Convert 'InvoiceDate' to datetime format
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'], format='%d-%m-%Y %H:%M')

# Calculate revenue for each transaction
df['Revenue'] = df['Quantity'] * df['UnitPrice']

# Filter out the United Kingdom
df_filtered = df[df['Country'] != 'United Kingdom']

# Aggregate quantity sold by country
country_demand = df_filtered.groupby('Country').agg({
    'Quantity': 'sum'
}).reset_index()

# Create a base map
map_center = [20, 0]  # Center the map roughly in the middle of the world
world_map = folium.Map(location=map_center, zoom_start=2)

# Add a heatmap layer
heat_data = [[row['Latitude'], row['Longitude'], row['Quantity']] for index, row in country_demand.iterrows()]
HeatMap(heat_data).add_to(world_map)

# Save map to HTML file
world_map.save('/content/country_demand_map.html')

# If running in a Jupyter Notebook, you can display the map inline
world_map
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3790         try:
-> 3791             return self._engine.get_loc(casted_key)
   3792         except KeyError as err:

index.pyx in pandas._libs.index.IndexEngine.get_loc()

index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas/_libs/hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 'Latitude'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
                                          4 frames
/usr/local/lib/python3.10/dist-packages/pandas/core/indexes/base.py in get_loc(self, key)
   3796             ):
   3797                 raise InvalidIndexError(key)
-> 3798             raise KeyError(key) from err
   3799         except TypeError:
   3800             # If we have a listlike key, _check_indexing_error will raise

KeyError: 'Latitude'
```

Next steps: Explain error

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.