```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.graph_objects as go
from plotly.subplots import make_subplots
```

Data loading and exploring

```
# data converted to csv as while loading in excel taking too much
df_es = pd.read_csv('/content/ES.csv')
df_fr = pd.read_csv('/content/FR.csv')
df_uk = pd.read_csv('/content/UK.csv')
df_de = pd.read_csv('/content/DE.csv')
df_us = pd.read_csv('/content/US.csv')

print(df_es.head(2))
print(df_fr.head(2))
print(df_uk.head(2))
print(df_de.head(2))
print(df_us.head(2))
```

```
0                 0.0                     0.0                       0.0
1                 0.0                     0.0                       0.0

   selling fees  fba fees  other transaction fees  other  total
0          1.68       0.0                     0.0    0.0 -12.31
1          0.00       0.0                     0.0   6.46   6.46

[2 rows x 28 columns]
                    date/time  settlement id        type             order id  \
0  Jan 1, 2021 1:24:15 AM PST    13704186161  Adjustment  114-3447667-8401830
1  Jan 1, 2021 2:56:03 AM PST    13704186161  Adjustment  114-6813146-6150605

                      sku                                   description  \
0  PAN-EU-App-PhoRep-1274  FBA Inventory Reimbursement - Customer Return
1            GR-J04V-42V3  FBA Inventory Reimbursement - Customer Return

   quantity marketplace Country       fulfilment  ... promotional rebates tax  \
0       1.0         NaN      US  Standard Orders  ...                        0
1       1.0         NaN      US  Standard Orders  ...                        0

  marketplace withheld tax selling fees fba fees other transaction fees other  \
0                        0          0.0        0                     0.0   0.0
1                        0          0.0        0                     0.0   0.0

   total  other transaction fees.1  other.1  total.1
0    0.0                       0.0    11.09    11.09
1    0.0                       0.0     5.14     5.14

[2 rows x 31 columns]
```

```
print(df_es.shape)
print(df_fr.shape)
print(df_uk.shape)
print(df_de.shape)
print(df_us.shape)
```

```
(266201, 28)
(531454, 28)
(433052, 28)
(781300, 28)
(706797, 31)
```

```
print(df_es.info())
print(df_fr.info())
print(df_uk.info())
print(df_de.info())
print(df_us.info())
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 266201 entries, 0 to 266200
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   date/time                 266201 non-null  object
```

```
 1   settlement id             266201 non-null  int64
 2   type                      266201 non-null  object
 3   order id                  261820 non-null  object
 4   sku                       263626 non-null  object
 5   description               266186 non-null  object
 6   quantity                  263384 non-null  float64
 7   marketplace               258443 non-null  object
 8   Country                   266201 non-null  object
 9   fulfilment                256012 non-null  object
 10  order city                256010 non-null  object
 11  order state               250154 non-null  object
 12  order postal              256008 non-null  object
 13  tax collection model      54 non-null      object
 14  product sales             266201 non-null  float64
 15  product sales tax         266201 non-null  float64
 16  postage credits           266201 non-null  float64
 17  shipping credits tax      266201 non-null  float64
 18  gift wrap credits         266201 non-null  float64
 19  giftwrap credits tax      266201 non-null  float64
 20  promotional rebates       266201 non-null  float64
 21  promotional rebates tax   266201 non-null  float64
 22  marketplace withheld tax  266201 non-null  float64
 23  selling fees              266201 non-null  float64
 24  fba fees                  266201 non-null  float64
 25  other transaction fees    266201 non-null  float64
 26  other                     266201 non-null  float64
 27  total                     266201 non-null  float64
dtypes: float64(15), int64(1), object(12)
memory usage: 56.9+ MB
None
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 531454 entries, 0 to 531453
Data columns (total 28 columns):
 #   Column                    Non-Null Count   Dtype
---  ------                    --------------   -----
 0   date/time                 531454 non-null  object
 1   settlement id             531454 non-null  int64
 2   type                      531454 non-null  object
 3   order id                  526558 non-null  object
 4   sku                       527757 non-null  object
 5   description               531441 non-null  object
 6   quantity                  527318 non-null  float64
 7   marketplace               518301 non-null  object
 8   Country                   531454 non-null  object
 9   fulfilment                514900 non-null  object
 10  order city                514897 non-null  object
 11  order state               70558 non-null   object
 12  order postal              514884 non-null  object
 13  tax collection model      40 non-null      object
 14  product sales             531454 non-null  float64
 15  product sales tax         531454 non-null  float64
 16  postage credits           531454 non-null  float64
```

```
print(df_es.describe())
print(df_fr.describe())
print(df_uk.describe())
print(df_de.describe())
print(df_us.describe())
```

```
       settlement id       quantity  product sales  product sales tax  \
count   2.662010e+05  263384.000000  266201.000000      266201.000000
mean    1.791711e+10       4.931841       7.820112           1.482195
std     1.834035e+09       3.164997       9.600997           1.898482
min     1.465180e+10       0.000000    -141.240000         -29.640000
25%     1.642754e+10       2.000000       6.600000           1.390000
50%     1.793293e+10       5.000000       7.430000           1.560000
75%     1.929530e+10       8.000000      10.730000           2.080000
max     2.134899e+10      10.000000     236.900000          46.800000

       postage credits  shipping credits tax  gift wrap credits  \
count    266201.000000         266201.000000      266201.000000
mean          0.197646              0.036783           0.000359
std           0.864099              0.167286           0.035671
min         -13.580000             -2.520000          -3.300000
25%           0.000000              0.000000           0.000000
50%           0.000000              0.000000           0.000000
75%           0.000000              0.000000           0.000000
max          58.800000              3.350000           3.990000

       giftwrap credits tax  promotional rebates  promotional rebates tax  \
count         266201.000000        266201.000000            266201.000000
mean               0.000072            -0.037739                -0.007024
std                0.007299             0.268605                 0.053622
min               -0.690000            -6.120000                -1.280000
25%                0.000000             0.000000                 0.000000
50%                0.000000             0.000000                 0.000000
75%                0.000000             0.000000                 0.000000
max                0.690000             3.990000                 0.720000

       marketplace withheld tax   selling fees       fba fees  \
count              266201.00000  266201.000000  266201.000000
mean                   -0.00042      -1.460285      -2.597334
std                     0.03500       1.652376       1.643336
min                    -5.46000     -57.600000    -100.000000
25%                     0.00000      -1.950000      -3.020000
50%                     0.00000      -1.390000      -2.510000
75%                     0.00000      -1.230000      -2.260000
max                     1.87000      16.600000      11.040000

       other transaction fees          other          total
count           266201.000000  266201.000000  266201.000000
mean                -0.704205       0.038845       4.769006
std                 18.800518       4.135264      21.367608
min               -524.240000    -614.740000    -614.740000
25%                  0.000000       0.000000       4.300000
50%                  0.000000       0.000000       5.180000
75%                  0.000000       0.000000       7.800000
max                  0.000000     180.000000     212.190000
       settlement id       quantity  product sales  product sales tax  \
count   5.314540e+05  527318.000000  531454.000000      531454.000000
mean    1.804464e+10       9.470039       8.458621           1.610610
std     1.854689e+09       6.064791       8.386757           1.624145
min     1.464804e+10       0.000000    -129.990000         -26.000000
25%     1.651081e+10       4.000000       6.660000           1.330000
50%     1.820392e+10       9.000000       8.320000           1.670000
75%     1.955005e+10      15.000000      10.820000           2.170000
max     2.134465e+10      20.000000     403.650000          55.440000
```

```
print(df_es.isnull().sum())
print(df_fr.isnull().sum())
print(df_uk.isnull().sum())
print(df_de.isnull().sum())
print(df_us.isnull().sum())
```

```
other                          0
total                          0
other transaction fees.1       0
other.1                        0
total.1                        0
dtype: int64
```

```
print(df_es.duplicated().sum())
print(df_fr.duplicated().sum())
print(df_uk.duplicated().sum())
print(df_de.duplicated().sum())
print(df_us.duplicated().sum())
```

```
719
490
2976
4353
5690
```

```
print()
```

Data cleaning and preprocessing

```
def clean_and_preprocess(df, country_code, exchange_rates):
    if country_code in ['ES', 'FR', 'UK']:
        df['date/time'] = pd.to_datetime(df['date/time'], format=
    elif country_code == 'DE':
        df['date/time'] = pd.to_datetime(df['date/time'], format=
    elif country_code == 'US':
        df['date/time'] = pd.to_datetime(df['date/time'], format=

    df = df.dropna(subset=['total'])
    df = df.drop_duplicates()

    df['country'] = country_code
    df['total'] = pd.to_numeric(df['total'], errors='coerce')

    if country_code in exchange_rates:
        df['total_eur'] = df['total'] * exchange_rates[country_co
    else:
        df['total_eur'] = df['total']

    return df


```

```
exchange_rates = {
    'UK': 1.18,
    'US': 0.93,
    'DE': 1.00,
    'FR': 1.00,
    'ES': 1.00
}

df_es_cleaned = clean_and_preprocess(df_es.copy(), 'ES', exchange
df_fr_cleaned = clean_and_preprocess(df_fr.copy(), 'FR', exchange
df_uk_cleaned = clean_and_preprocess(df_uk.copy(), 'UK', exchange
df_de_cleaned = clean_and_preprocess(df_de.copy(), 'DE', exchange
df_us_cleaned = clean_and_preprocess(df_us.copy(), 'US', exchange

print("Done cleaning and prepping all datasets.")

```

```
Done cleaning and prepping all datasets.
```

Combine data

```
dataframes_to_combine = [df_de_cleaned, df_es_cleaned, df_fr_clea
df_combined = pd.concat(dataframes_to_combine, ignore_index=True)

display(df_combined.head())
print(df_combined.shape)
```

	date/time	settlement id	type	order id	sku	description	quantity	marke
0	2020-12-31 23:54:41	14668686782	Refund	305-3488972-4981168	PAN-EU-App-PhoRep-1151 Child2	MMOBIEL Akku kompatibel mit iPhone 4S Li-Ion B...	9.0	ama
1	2021-01-01 00:25:44	14668686782	Adjustment	305-3760164-5117930	PAN-EU-MM-Tool-1017	Versand durch Amazon Erstattung für Lagerbesta...	3.0	
2	2021-01-01 00:26:05	14668686782	Adjustment	302-2179035-0298738	PAN-EU-Sam-PhoRep-1215	Versand durch Amazon Erstattung für Lagerbesta...	7.0	
3	2021-01-01 01:02:31	14668686782	Adjustment	NaN	PAN-EU-App-PhoRep-1008	Versand durch Amazon Erstattung für Lagerbesta...	3.0	
4	2021-01-01 02:38:31	14668686782	Adjustment	NaN	PAN-EU-Sam-PhoRep-1311 Child1	Versand durch Amazon Erstattung für Lagerbesta...	4.0	

5 rows × 33 columns

```
(2697990, 33)
```

Data analysis and reporting

Subtask:

Perform data analysis to determine sales by country and region, analyze trends over time, and investigate refunds.

Reasoning: Perform the data analysis steps requested in the subtask: calculate total sales by country and type, group by month for time trend analysis, and calculate total refund amount by country.

```

def summarize_total(df, group_col, value_col='total_eur', filter_
```

```
    if filter_cond is not None:
        df = df.query(filter_cond)
    summary = df.groupby(group_col)[value_col].sum().reset_index(
    return summary

df_combined['month'] = df_combined['date/time'].dt.to_period('M')

sales_by_country = summarize_total(df_combined, 'country')
sales_by_type = summarize_total(df_combined, 'type')
monthly_sales = summarize_total(df_combined, 'month')
refunds_by_country = summarize_total(df_combined, 'country', filt

print("Total Sales by Country:")
print(sales_by_country)

print("\nTotal Sales by Transaction Type:")
print(sales_by_type)

print("\nMonthly Sales Trends:")
print(monthly_sales)

print("\nTotal Refund Amount by Country:")
print(refunds_by_country)
```

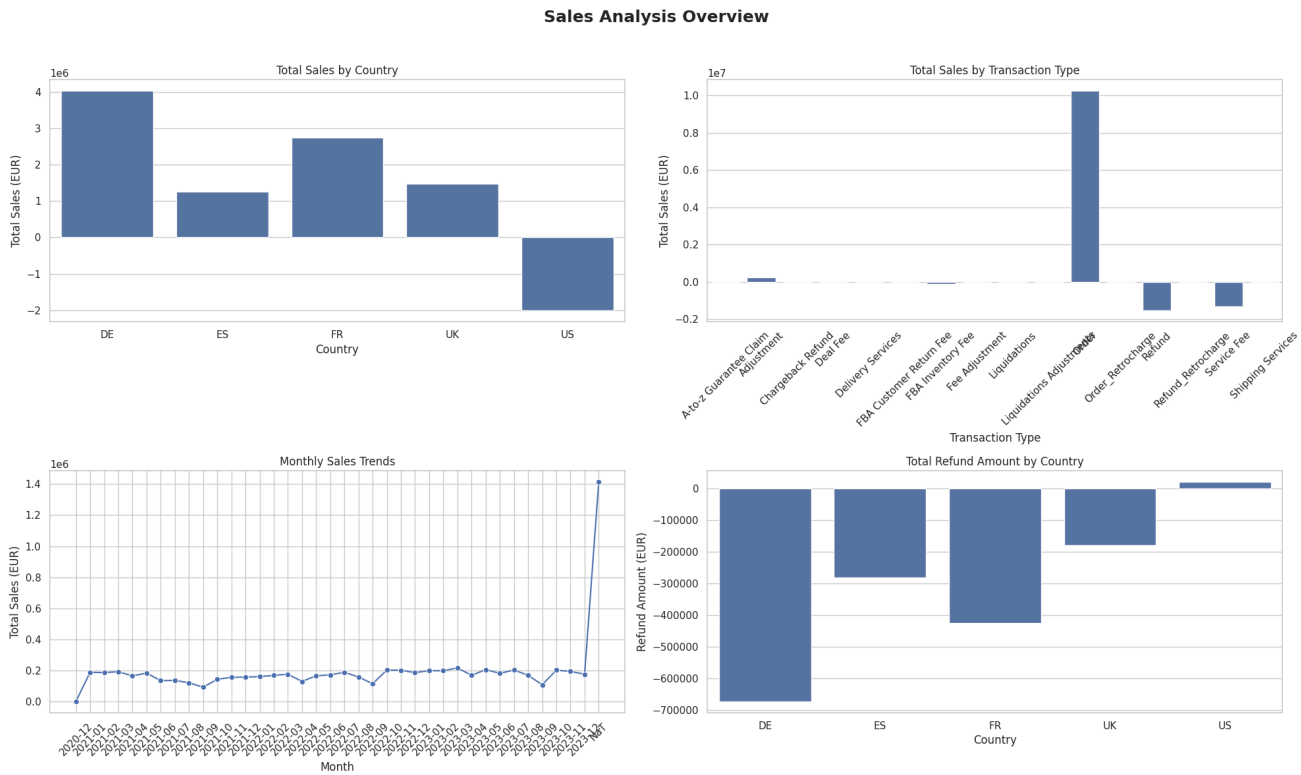
```
7            Fee Adjustment  3.988118e+03
8              Liquidations  1.514827e+03
9   Liquidations Adjustments -7.296180e+01
10                     Order  1.025627e+07
11         Order_Retrocharge  3.542949e+03
12                    Refund -1.537640e+06
13        Refund_Retrocharge -8.101050e+02
14               Service Fee -1.332817e+06
15         Shipping Services -5.990000e+00

Monthly Sales Trends:
```

```
16  2022-04  1.298311e+05
17  2022-05  1.655399e+05
18  2022-06  1.720682e+05
19  2022-07  1.880690e+05
20  2022-08  1.580547e+05
21  2022-09  1.147745e+05
22  2022-10  2.024582e+05
23  2022-11  2.019905e+05
24  2022-12  1.857764e+05
25  2023-01  1.992102e+05
26  2023-02  1.983045e+05
27  2023-03  2.167420e+05
28  2023-04  1.686332e+05
29  2023-05  2.059485e+05
30  2023-06  1.813801e+05
31  2023-07  2.033312e+05
32  2023-08  1.688099e+05
33  2023-09  1.084340e+05
34  2023-10  2.024556e+05
35  2023-11  1.937571e+05
36  2023-12  1.761942e+05
37      NaT  1.414322e+06

Total Refund Amount by Country:
  country    total_eur
0      DE -672532.0900
1      ES -280885.7200
2      FR -425004.8100
3      UK -179472.0292
4      US   20254.2654
```

Visualization

```
sns.set(style='whitegrid')

fig, axs = plt.subplots(2, 2, figsize=(20, 12))
fig.suptitle('Sales Analysis Overview', fontsize=18, fontweight='

# Chart 1: Total Sales by Country
sns.barplot(x='country', y='total_eur', data=sales_by_country, ax
axs[0, 0].set_title('Total Sales by Country')
axs[0, 0].set_xlabel('Country')
axs[0, 0].set_ylabel('Total Sales (EUR)')

# Chart 2: Total Sales by Transaction Type
sns.barplot(x='type', y='total_eur', data=sales_by_type, ax=axs[0
axs[0, 1].set_title('Total Sales by Transaction Type')
axs[0, 1].set_xlabel('Transaction Type')
axs[0, 1].set_ylabel('Total Sales (EUR)')
axs[0, 1].tick_params(axis='x', rotation=45)

# Chart 3: Monthly Sales Trends
```

```
sns.lineplot(x='month', y='total_eur', data=monthly_sales, marker
axs[1, 0].set_title('Monthly Sales Trends')
axs[1, 0].set_xlabel('Month')
axs[1, 0].set_ylabel('Total Sales (EUR)')
axs[1, 0].tick_params(axis='x', rotation=45)

# Chart 4: Total Refund Amount by Country
sns.barplot(x='country', y='total_eur', data=refunds_by_country,
axs[1, 1].set_title('Total Refund Amount by Country')
axs[1, 1].set_xlabel('Country')
axs[1, 1].set_ylabel('Refund Amount (EUR)')

plt.tight_layout(rect=[0, 0, 1, 0.96])
plt.show()

```




```
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=[
        "Total Sales by Country",
        "Total Sales by Transaction Type",
        "Monthly Sales Trends",
        "Total Refund Amount by Country"
```

```
    ]
)

# Chart 1: Total Sales by Country
fig.add_trace(
    go.Bar(
        x=sales_by_country['country'],
        y=sales_by_country['total_eur'],
        name="Sales by Country",
        marker_color='royalblue'
    ),
    row=1, col=1
)

# Chart 2: Total Sales by Transaction Type
fig.add_trace(
    go.Bar(
        x=sales_by_type['type'],
        y=sales_by_type['total_eur'],
        name="Sales by Type",
        marker_color='seagreen'
    ),
    row=1, col=2
)

# Chart 3: Monthly Sales Trends
fig.add_trace(
    go.Scatter(
        x=monthly_sales['month'],
        y=monthly_sales['total_eur'],
        mode='lines+markers',
        name="Monthly Sales",
        line=dict(color='orange', width=3)
    ),
    row=2, col=1
)

# Chart 4: Total Refund Amount by Country
fig.add_trace(
    go.Bar(
        x=refunds_by_country['country'],
        y=refunds_by_country['total_eur'],
        name="Refunds by Country",
        marker_color='crimson'
    ),
```

```
    row=2, col=2
)

# Update layout
fig.update_layout(
    height=800,
    width=1200,
    title_text="Sales Analytics Dashboard"
    showlegend=False,
    template='plotly_white',
    margin=dict(t=80, b=50)
)

fig.show()
```