Importing Libraries

In [1]:

```
import pandas as pd
import numpy as np
import chart_studio.plotly as py
import cufflinks as cf
import seaborn as sns
import plotly.express as px

%matplotlib inline

from plotly.offline import download_plotlyjs, init_notebook_mode, plot, iplot

init_notebook_mode(connected=True)
cf.go_offline()
```

Loading Data

The data if collected from kaggle.com. It shows the sales report for an online retail store.

In this section, the data is loaded.

In [51]:

```
df = pd.read_csv('./data.csv')
print(f'The number of rows and columns in the data shown as (row_count, column_count): {d
f.shape} /n')
print("Showing the first 5 rows of the data set.")
df.head()
```

```
The number of rows and columns in the data shown as (row_count, column_count): (541909, 8
) /n
Showing the first 5 rows of the data set.
```

Out[51]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	12/1/10 8:26	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	12/1/10 8:26	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	12/1/10 8:26	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	12/1/10 8:26	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	12/1/10 8:26	3.39	17850.0	United Kingdom

In [52]:

```
print("List of columns and the data types \n")
print(df.dtypes)
```

```
List of columns and the data types

InvoiceNo       object
StockCode       object
Description     object
```

```
Quantity         int64
InvoiceDate     object
UnitPrice      float64
CustomerID     float64
Country         object
dtype: object
```

In [53]:

```
print("Checking for null values, shows the number of null values in each column \n")
print(df.isnull().sum())
```

```
Checking for null values, shows the number of null values in each column

InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     135080
Country             0
dtype: int64
```

In [54]:

```
print('Counting unique values in each column \n')
print(df.nunique())
```

```
Counting unique values in each column

InvoiceNo      25900
StockCode       4070
Description     4223
Quantity         722
InvoiceDate    23260
UnitPrice       1630
CustomerID      4372
Country           38
dtype: int64
```

Statistical Description

In [55]:

```
df.describe(include='all')
```

Out[55]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
count	541909	541909	540455	541909.000000	541909	541909.000000	406829.000000	541909
unique	25900	4070	4223	NaN	23260	NaN	NaN	38
top	573585	85123A	WHITE HANGING HEART T-LIGHT HOLDER	NaN	10/31/11 14:41	NaN	NaN	United Kingdom
freq	1114	2313	2369	NaN	1114	NaN	NaN	495478
mean	NaN	NaN	NaN	9.552250	NaN	4.611114	15287.690570	NaN
std	NaN	NaN	NaN	218.081158	NaN	96.759853	1713.600303	NaN
min	NaN	NaN	NaN	-80995.000000	NaN	-11062.060000	12346.000000	NaN
25%	NaN	NaN	NaN	1.000000	NaN	1.250000	13953.000000	NaN
50%	NaN	NaN	NaN	3.000000	NaN	2.080000	15152.000000	NaN
75%	NaN	NaN	NaN	10.000000	NaN	4.130000	16791.000000	NaN
max	NaN	NaN	NaN	80995.000000	NaN	38970.000000	18287.000000	NaN

Cleaning and modifying data set

Replacing Null values with a more meaningful value for further analysis.

In [56]:

```
df['Description'].replace({np.nan: 'No Description for Item'}, inplace=True)
df['CustomerID'].replace({np.nan: 'No ID'}, inplace=True)
```

Modifying data for by adding necessary columns and changing data types

In [57]:

```
df['TotalSales'] = df['Quantity']*df['UnitPrice']
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])



df['day'] = df['InvoiceDate'].dt.day
df['month'] = df['InvoiceDate'].dt.month
df['year'] = df['InvoiceDate'].dt.year
df['hour'] = df['InvoiceDate'].dt.hour
df['minute'] = df['InvoiceDate'].dt.minute
```

In [58]:

```
print("Checking for null values, shows the number of null values in each column \n")
print(df.isnull().sum())
```

```
Checking for null values, shows the number of null values in each column

InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
TotalSales     0
day            0
month          0
year           0
hour           0
minute         0
dtype: int64
```

In [59]:

```
print("List of columns and the data types \n")
print(df.dtypes)
```

```
List of columns and the data types

InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID             object
Country                object
TotalSales            float64
day                     int64
month                   int64
year                    int64
hour                    int64
```

```
minute                  int64
dtype: object
```

In [60]:

```
print("Showing the first 5 rows of the data set.")
df.head()
```

```
Showing the first 5 rows of the data set.
```

Out[60]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalSales	day	month	year
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30	1	12	2010
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	1	12	2010
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00	1	12	2010
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	1	12	2010
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	1	12	2010

Exploring the data using visualization

From the count in the previous section, there are 25,900 unique invoice, which means there are 25,900 sales made. For the next section of analysis, the data set will be grouped by invoices number and the sale of items in the invoice will be summed to give total invoice value. The visualization will show the distribution of the sales amount by invoice.

In [61]:

```
df_sales_by_invoice = df.groupby(by=['InvoiceNo'], as_index=False).sum()[['InvoiceNo', '
TotalSales']]
df_sales_by_invoice.head()
```

```
/var/folders/t8/pxlpls913sx9206q6nkm_q480000gn/T/ipykernel_2200/3899845691.py:1: FutureWa
rning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future vers
ion, numeric_only will default to False. Either specify numeric_only or select only colum
ns which should be valid for the function.
```

Out[61]:

	InvoiceNo	TotalSales
0	536365	139.12
1	536366	22.20
2	536367	278.73

	InvoiceNo	TotalSales
3	536368	70.05
4	536369	17.85

In this section, the sales is grouped based on the date the sales occurred and plotted.

In [62]:

```
df_sales_by_date = df.groupby(by=['InvoiceDate'], as_index=False).sum()[['InvoiceDate',
'TotalSales']]
df_sales_by_date.head()
```

```
/var/folders/t8/pxlpls913sx9206q6nkm_q480000gn/T/ipykernel_2200/2048500629.py:1: FutureWa
rning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future vers
ion, numeric_only will default to False. Either specify numeric_only or select only colum
ns which should be valid for the function.
```

Out[62]:

	InvoiceDate	TotalSales
0	2010-12-01 08:26:00	139.12
1	2010-12-01 08:28:00	22.20
2	2010-12-01 08:34:00	348.78
3	2010-12-01 08:35:00	17.85
4	2010-12-01 08:45:00	855.86

In [70]:

```
import plotly.express as px
# df = px.data.tips()
sales_distribution_fig = px.box(df_sales_by_invoice, y='TotalSales', points='all', heigh
t=1000,)

sales_distribution_fig.update_layout(title='Sales amount distribution',
                           xaxis = dict(
                               showline=True,
                               showgrid =False,
                               showticklabels=True,
                               linecolor='rgb(204, 204, 204)',
                               linewidth =2,
                               ),
                           yaxis = dict(
                               showline=False,
                               zeroline=True,
                               showgrid =False,
                               ),
                           autosize=True,
                           margin = dict(autoexpand=True, l=100, r=20, t=110),
                           showlegend=True,
                           plot_bgcolor = 'white'

                           )
```

In [81]:

```
import plotly.graph_objects as go

# px.line(df_sales_by_date, x='InvoiceDate', y='TotalSales', labels={'x': 'Date', 'y': 'S
ales'})

sale_date_fig = go.Figure()

trace_01 = go.Scatter(x=df_sales_by_date.InvoiceDate, y=df_sales_by_date.TotalSales, mod
e='lines', name='Sales')

sale_date_fig.add_trace(trace_01)




sale_date_fig.update_layout(title='Sales over Dec 2010 to Dec of 2011',
                            xaxis = dict(
                                showline=True,
                                showgrid =False,
                                showticklabels=True,
                                linecolor='rgb(204, 204, 204)',
                                linewidth =2,
                                ),
                            yaxis = dict(
                                showline=False,
                                zeroline=True,
                                showgrid =False,
                                ),
                            autosize=True,
                            margin = dict(autoexpand=True, l=100, r=20, t=110),
                            showlegend=True,
                            plot_bgcolor = 'white'

                            )
```

In this section, the sales are grouped on a monthly bases and a bar chart is used to show sales on each month.

In [22]:

```
df_sales_by_month = df.groupby(by=['year', 'month'], as_index=False).sum()[['year', 'mon
th', 'TotalSales']]
df_sales_by_month['Month'] = [str(int(df_sales_by_month.iloc[i]['month'])) + '-' +  str(
int(df_sales_by_month.iloc[i]['year'])) for i in range(df_sales_by_month.shape[0])]

df_sales_by_month.head()
```

```
/var/folders/t8/pxlpls913sx9206q6nkm_q480000gn/T/ipykernel_2200/1375340714.py:1: FutureWa
rning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future vers
ion, numeric_only will default to False. Either specify numeric_only or select only colum
ns which should be valid for the function.
```

Out[22]:

	year	month	TotalSales	Month

	year	month	TotalSales	Month
0	2010	12	748957.020	12-2010
1	2011	1	560000.260	1-2011
2	2011	2	498062.650	2-2011
3	2011	3	683267.080	3-2011
4	2011	4	493207.121	4-2011

In [29]:

```
sales_monthly_fig = px.bar(df_sales_by_month, y='TotalSales', x='Month', text='TotalSale
s')


sales_monthly_fig.update_layout(title='Sales over Dec 2010 to Dec of 2011',
                                xaxis = dict(
                                    showline=True,
                                    showgrid =False,
                                    showticklabels=True,
                                    linecolor='rgb(204, 204, 204)',
                                    linewidth =2,
                                    ),
                                yaxis = dict(
                                    showline=False,
                                    zeroline=True,
                                    showgrid =False,
                                    ),
                                autosize=True,
                                margin = dict(autoexpand=True, l=100, r=20, t=90),
                                showlegend=True,
                                plot_bgcolor = 'white'

                                )
sales_monthly_fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
```

In this section, sales will be grouped by country bases and a pie chart is used to show the contribution of sales from each country to the total sales

In [31]:

```
df_sales_by_country = df.groupby(by=['Country'], as_index=False).sum()[['Country','Total
Sales']]
print(df_sales_by_country.shape)
df_sales_by_country.head()
```

```
(38, 2)
```

```
/var/folders/t8/pxlpls913sx9206q6nkm_q480000gn/T/ipykernel_2200/3955509401.py:1: FutureWa
rning:

The default value of numeric_only in DataFrameGroupBy.sum is deprecated. In a future vers
ion, numeric_only will default to False. Either specify numeric_only or select only colum
ns which should be valid for the function.
```

Out[31]:

	Country	TotalSales
0	Australia	137077.27
1	Austria	10154.32
2	Bahrain	548.40
3	Belgium	40910.96
4	Brazil	1143.60

In [47]:

```
# sales_by_country_fig = px.pie(df_sales_by_country, values='TotalSales', names='Country'
, title='Sales by Country', color_discrete_sequence=px.colors.sequential.RdBu)
sales_by_country_fig = px.bar(df_sales_by_country, y='TotalSales', x='Country', text='To
talSales', color='Country')


sales_by_country_fig.update_layout(
                            uniformtext_minsize = 8,
                            title='Sales by Country',
                            xaxis = dict(
                                showline=True,
                                showgrid =False,
                                showticklabels=True,
                                linecolor='rgb(204, 204, 204)',
                                linewidth =2,
                                ),
                            yaxis = dict(
                                showline=False,
                                zeroline=True,
                                showgrid =False,
                                ),
                            autosize=True,
                            margin = dict(autoexpand=True, l=100, r=30, t=50),
                            showlegend=True,
                            plot_bgcolor = 'white',
                            height = 500
                            )
sales_by_country_fig.update_traces(texttemplate='%{text:.2s}', textposition='outside')
```