**Assignment 5**

Aim:

K-Means Clustering is a popular unsupervised machine learning technique used for clustering or grouping similar data points based on their characteristics or features. The primary aim of K-Means Clustering is to partition a given dataset into a predefined number of clusters, where each cluster represents a group of data points that are similar to each other and dissimilar to data points in other clusters.

Theory:

K-Means Clustering is based on the concept of centroid-based clustering. The algorithm starts by randomly selecting k number of initial centroids from the dataset. Then, it assigns each data point to the closest centroid based on their similarity. After that, it recalculates the centroids of each cluster by taking the mean of all the data points in that cluster. The algorithm repeats the process of assigning data points to the nearest centroid and updating the centroid until the convergence criterion is met, which is usually a set number of iterations or until the centroids no longer move significantly.

Case Study:

In the current scenario, we have considered online retail dataset. Online retail is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail. The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

We will be using the online retail transactional dataset to build a RFM clustering and choose the best set of customers which the company should target.

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** missingno **as** msno

df = pd.read\_csv("onlineretail.csv",delimiter=',', encoding = "ISO- 8859-1")

df.head()

InvoiceNo StockCode Description Quantity \

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 |

InvoiceDate UnitPrice CustomerID Country 0 01-12-2010 8.26 2.55 17850.0 United Kingdom

1 01-12-2010 8.26 3.39 17850.0 United Kingdom

2 01-12-2010 8.26 2.75 17850.0 United Kingdom

3 01-12-2010 8.26 3.39 17850.0 United Kingdom

4 01-12-2010 8.26 3.39 17850.0 United Kingdom

df.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 541909 entries, 0 to 541908 Data columns (total 8 columns):

# Column Non-Null Count Dtype

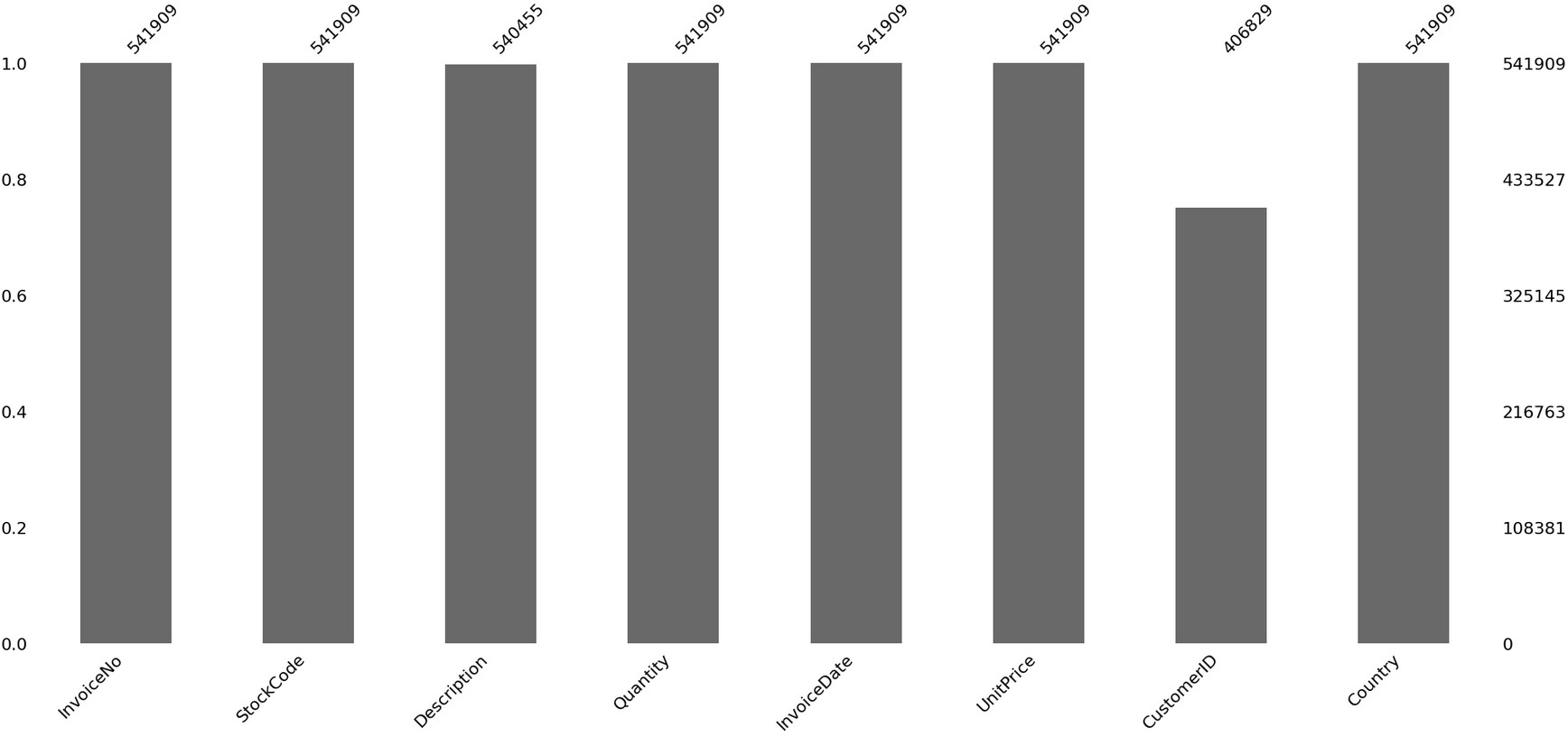
1. InvoiceNo 541909 non-null object
2. StockCode 541909 non-null object
3. Description 540455 non-null object
4. Quantity 541909 non-null int64
5. InvoiceDate 541909 non-null object
6. UnitPrice 541909 non-null float64
7. CustomerID 406829 non-null float64
8. Country 541909 non-null object dtypes: float64(2), int64(1), object(5) memory usage: 33.1+ MB

df.describe()

|  |  |  |  |
| --- | --- | --- | --- |
|  | Quantity | UnitPrice | CustomerID |
| count | 541909.000000 | 541909.000000 | 406829.000000 |
| mean | 9.552250 | 4.611114 | 15287.690570 |
| std | 218.081158 | 96.759853 | 1713.600303 |
| min | -80995.000000 | -11062.060000 | 12346.000000 |
| 25% | 1.000000 | 1.250000 | 13953.000000 |
| 50% | 3.000000 | 2.080000 | 15152.000000 |
| 75% | 10.000000 | 4.130000 | 16791.000000 |
| max | 80995.000000 | 38970.000000 | 18287.000000 |

msno.bar(df)

<Axes: >



|  |  |
| --- | --- |
| df.count() |  |
| InvoiceNo | 541909 |
| StockCode | 541909 |
| Description | 540455 |
| Quantity | 541909 |
| InvoiceDate | 541909 |
| UnitPrice | 541909 |
| CustomerID | 406829 |
| Country | 541909 |
| dtype: int64 |  |

df[df['CustomerID'].isnull()].count()

|  |  |
| --- | --- |
| InvoiceNo | 135080 |
| StockCode | 135080 |
| Description | 133626 |
| Quantity | 135080 |
| InvoiceDate | 135080 |
| UnitPrice | 135080 |
| CustomerID | 0 |

Country 135080

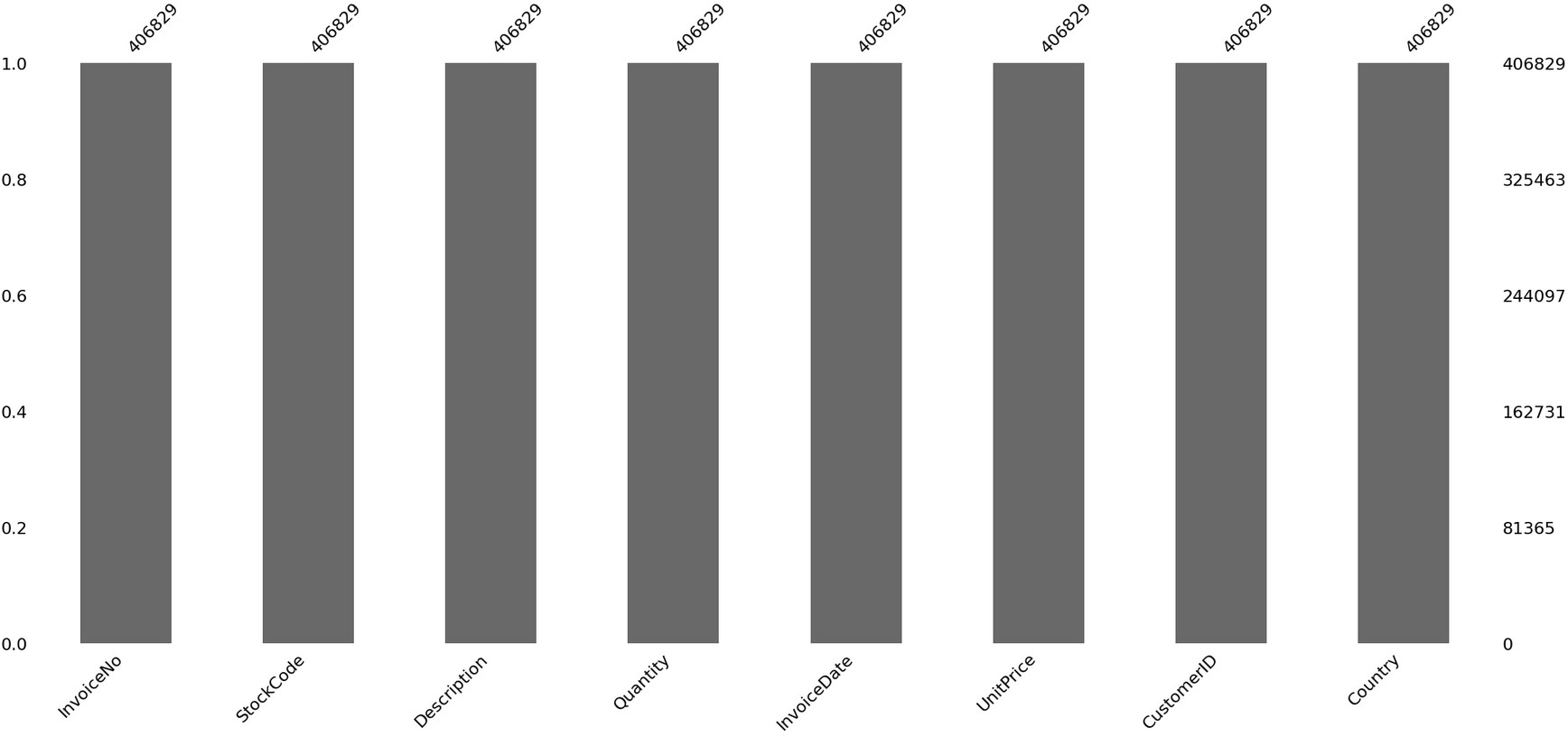
dtype: int64

a = 100 - ((541909-135000)/541909 \* 100)

print("Percentge of values missing:- ",a) Percentge of values missing:- 24.911931708091203 df.dropna(inplace=True)

msno.bar(df)

<Axes: >



df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'], format='%d-%m-%Y

%H.%M')

df['Total Amount Spent']= df['Quantity'] \* df['UnitPrice']

total\_amount = df['Total Amount Spent'].groupby(df['CustomerID']).sum()

total\_amount = pd.DataFrame(total\_amount).reset\_index() total\_amount.head()

CustomerID Total Amount Spent 0 12346.0 0.00

1 12347.0 4310.00

2 12348.0 1797.24

3 12349.0 1757.55

4 12350.0 334.40

transactions = df['InvoiceNo'].groupby(df['CustomerID']).count() transaction = pd.DataFrame(transactions).reset\_index() transaction.head()

CustomerID InvoiceNo 0 12346.0 2

1 12347.0 182

2 12348.0 31

3 12349.0 73

4 12350.0 17

final = df['InvoiceDate'].max() df['Last\_transact'] = final - df['InvoiceDate']

LT = df.groupby(df['CustomerID']).min()['Last\_transact'] LT = pd.DataFrame(LT).reset\_index()

LT.head()

CustomerID Last\_transact 0 12346.0 325 days 02:33:00

1 12347.0 1 days 20:58:00

2 12348.0 74 days 23:37:00

3 12349.0 18 days 02:59:00

4 12350.0 309 days 20:49:00

df\_new = pd.merge(total\_amount, transaction, how='inner', on='CustomerID')

df\_new = pd.merge(df\_new, LT, how='inner', on='CustomerID') df\_new.head()

CustomerID Total Amount Spent InvoiceNo Last\_transact 0 12346.0 0.00 2 325 days 02:33:00

1 12347.0 4310.00 182 1 days 20:58:00

2 12348.0 1797.24 31 74 days 23:37:00

3 12349.0 1757.55 73 18 days 02:59:00

4 12350.0 334.40 17 309 days 20:49:00

df\_new['Last\_transact'] = df\_new['Last\_transact'].dt.days df\_new.head()

CustomerID Total Amount Spent InvoiceNo Last\_transact 0 12346.0 0.00 2 325

1 12347.0 4310.00 182 1

2 12348.0 1797.24 31 74

3 12349.0 1757.55 73 18

4 12350.0 334.40 17 309

**from** sklearn.cluster **import** KMeans kmeans= KMeans(n\_clusters=2)

kmeans.fit(df\_new[['Total Amount Spent', 'InvoiceNo', 'Last\_transact']])

pred = kmeans.predict(df\_new[['Total Amount Spent', 'InvoiceNo', 'Last\_transact']])

C:\Users\Lenovo\AppData\Local\Programs\Python\Python311\Lib\site- packages\sklearn\cluster\\_kmeans.py:870: FutureWarning: The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

warnings.warn( kmeans.cluster\_centers\_

array([[1.65070406e+03, 9.06685754e+01, 9.11630783e+01], [1.82181982e+05, 1.82833333e+03, 6.66666667e+00]])

kmeans.labels\_

array([0, 0, 0, ..., 0, 0, 0])

pred = pd.DataFrame(pred, columns=['pred']) df\_new = df\_new.join(pred)

fig, ax =plt.subplots(nrows= 1, ncols = 3, figsize= (14,6)) ty=sns.stripplot(x='pred', y='Total Amount Spent', data=df\_new, s=8, ax = ax[0], palette='magma\_r')

sns.despine(left=True)

ty.set\_title('Clusters based on different Amounts') ty.set\_ylabel('Total Spent') ty.set\_xlabel('Clusters')

tt=sns.boxplot(x='pred', y='InvoiceNo', data=df\_new, ax = ax[1], palette='coolwarm\_r')

tt.set\_title('Clusters based on Number of Transactions') tt.set\_ylabel('Total Transactions') tt.set\_xlabel('Clusters')

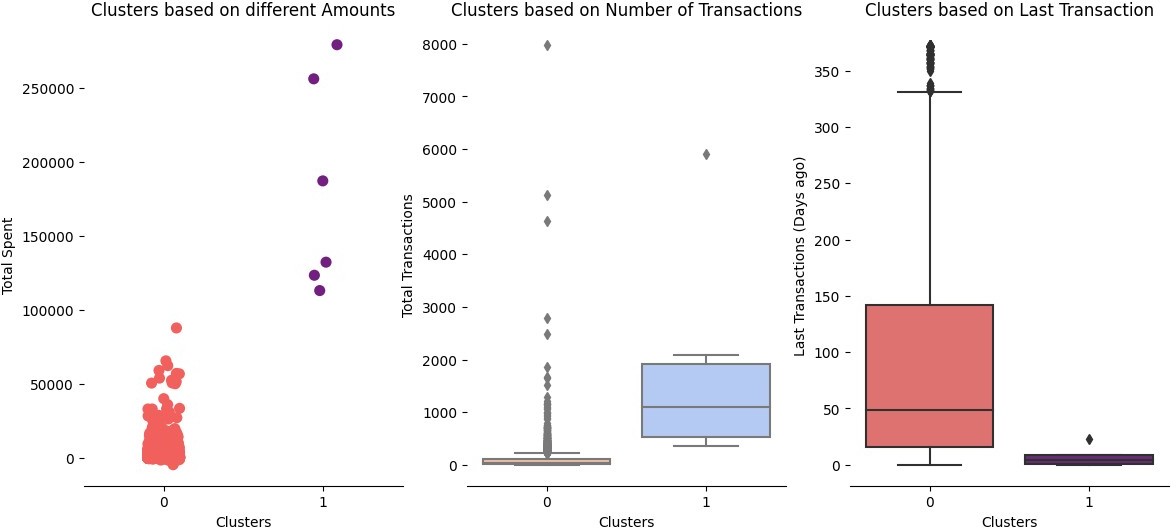
tr=sns.boxplot(x='pred', y='Last\_transact', data=df\_new, ax = ax[2], palette='magma\_r')

tr.set\_title('Clusters based on Last Transaction') tr.set\_ylabel('Last Transactions (Days ago)') tr.set\_xlabel('Clusters')

C:\Users\Lenovo\AppData\Local\Temp\ipykernel\_12464\3755150140.py:2: FutureWarning: Passing `palette` without assigning `hue` is deprecated.

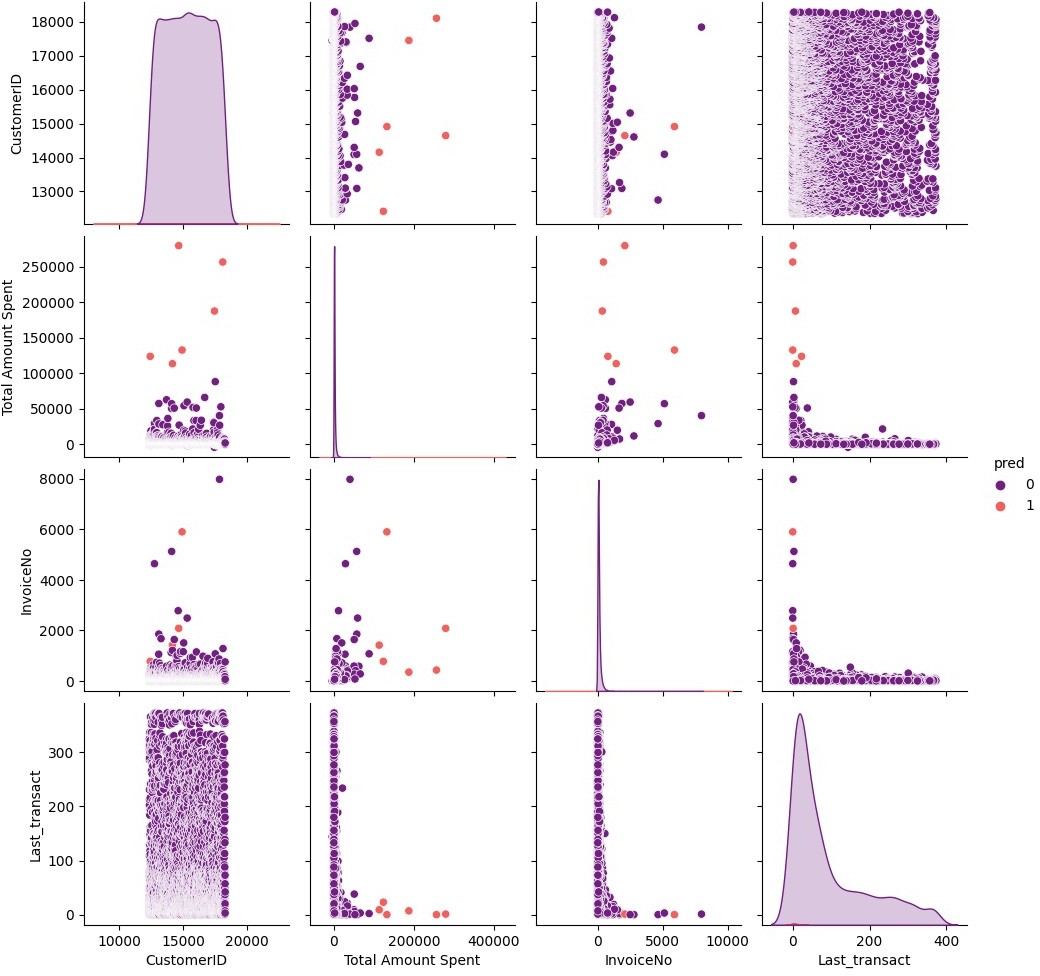
ty=sns.stripplot(x='pred', y='Total Amount Spent', data=df\_new, s=8, ax = ax[0], palette='magma\_r')

Text(0.5, 0, 'Clusters')



sns.pairplot(hue='pred', data=df\_new, diag\_kind='kde', palette='magma')

<seaborn.axisgrid.PairGrid at 0x19aabdd0210>



kmeans.inertia\_ 100276002377.65202

error\_rate = []

**for** clusters **in** range(1,16):

kmeans = KMeans(n\_clusters = clusters) kmeans.fit(df\_new) kmeans.predict(df\_new) error\_rate.append(kmeans.inertia\_)

error\_rate

Cluster Error 0 1 3.085422e+11

1 2 1.132431e+11

2 3 6.237784e+10

3 4 4.337622e+10

4 5 3.095299e+10

5 6 2.195090e+10

6 7 1.603851e+10

7 8 1.281771e+10

8 9 1.034822e+10

9 10 8.741114e+09

10 11 7.656500e+09

11 12 6.596477e+09

12 13 5.794264e+09

13 14 5.169446e+09

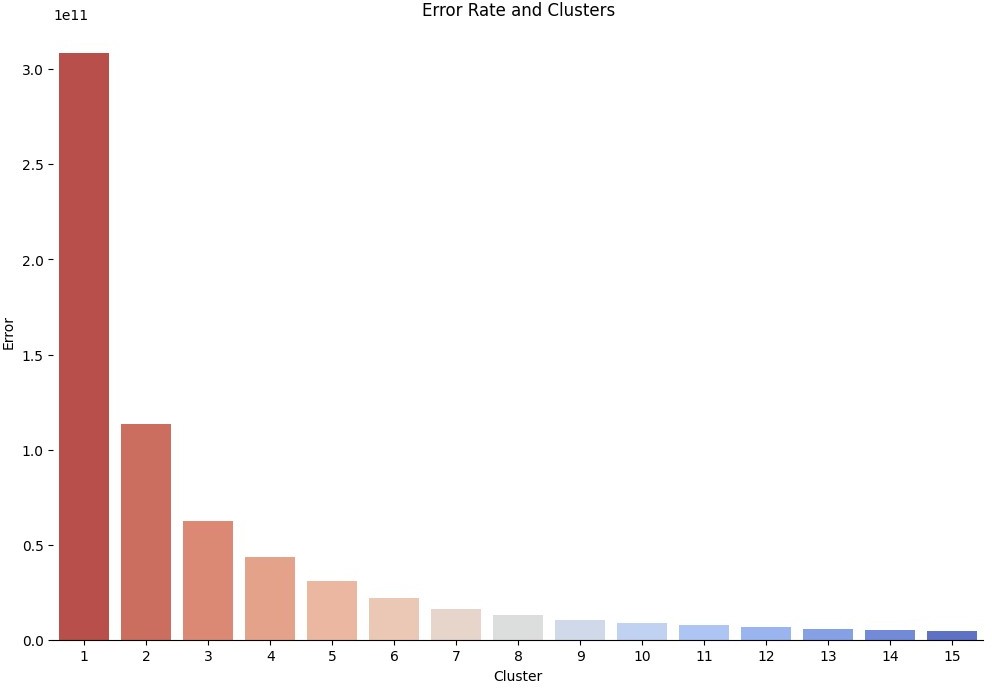
14 15 4.779799e+09

plt.figure(figsize=(12,8))

p = sns.barplot(x='Cluster', y= 'Error', data= error\_rate, palette='coolwarm\_r')

sns.despine(left=True) p.set\_title('Error Rate and Clusters')

Text(0.5, 1.0, 'Error Rate and Clusters')



country\_wise = df.groupby('Country').sum()

C:\Users\Lenovo\AppData\Local\Temp\ipykernel\_12464\2950913764.py:1: FutureWarning: The default value of numeric\_only in DataFrameGroupBy.sum is deprecated. In a future version, numeric\_only will default to False. Either specify numeric\_only or select only columns which should be valid for the function.

country\_wise = df.groupby('Country').sum()

country\_codes = pd.read\_csv('wikipedia-iso-country-codes.csv', names=['Country', 'two', 'three', 'numeric', 'ISO'])

country\_codes.head()

Country two three

numeric \

1. English short name lower case Alpha-2 code Alpha-3 code Numeric code
2. Afghanistan AF AFG

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 004 |  | | | | | | | |
| 2 |  | Åland | Islands |  | AX |  | ALA |  |
| 248 |  |  |  |  |  |  |  |  |
| 3 |  |  | Albania |  | AL |  | ALB |  |
| 008 |  |  |  |  |  |  |  |  |
| 4 |  |  | Algeria |  | DZ |  | DZA |  |
| 012 |  |  |  |  |  |  |  |  |

ISO

0 ISO 3166-2

1 ISO 3166-2:AF

2 ISO 3166-2:AX

3 ISO 3166-2:AL

4 ISO 3166-2:DZ

country\_wise = pd.merge(country\_codes,country\_wise, on='Country') country\_wise.head()

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Country two three | numeric | ISO | Quantity | UnitPrice | \ |
| 0 Australia AU AUS | 036 | ISO 3166-2:AU | 83653 | 4054.75 |  |
| 1 Austria AT AUT | 040 | ISO 3166-2:AT | 4827 | 1701.52 |  |
| 2 Bahrain BH BHR | 048 | ISO 3166-2:BH | 260 | 78.95 |  |
| 3 Belgium BE BEL | 056 | ISO 3166-2:BE | 23152 | 7540.13 |  |
| 4 Brazil BR BRA | 076 | ISO 3166-2:BR | 356 | 142.60 |  |

|  |  |  |
| --- | --- | --- |
|  | CustomerID | Total Amount Spent |
| 0 | 15693002.0 | 137077.27 |
| 1 | 5021102.0 | 10154.32 |
| 2 | 210027.0 | 548.40 |
| 3 | 25718288.0 | 40910.96 |
| 4 | 408608.0 | 1143.60 |

**from** plotly **import** version

**import** cufflinks **as** cf

**from** plotly.offline **import** download\_plotlyjs, init\_notebook\_mode, plot, iplot

init\_notebook\_mode(connected=True) cf.go\_offline()

**import** plotly.graph\_objs **as** go

data = dict(type='choropleth',colorscale='GnBu', locations = country\_wise['three'], locationmode = 'ISO-3', z= country\_wise['Total Amount Spent'], text = country\_wise['Country'], colorbar={'title':'Revenue'}, marker = dict(line=dict(width=0))) layout = dict(title = 'European Countries According to Revenue!', geo

= dict(scope='europe',showlakes=False, projection = {'type': 'winkel tripel'}))

Choromaps2 = go.Figure(data=[data], layout=layout) iplot(Choromaps2)



data = dict(type='choropleth',colorscale='rainbow', locations = country\_wise['three'], locationmode = 'ISO-3', z= country\_wise['Total Amount Spent'], text = country\_wise['Country'], colorbar={'title':'Revenue'}, marker = dict(line=dict(width=0)))

layout = dict(title = 'All Countries According to Revenue!', geo = dict(scope='world',showlakes=False, projection = {'type': 'winkel tripel'}))

Choromaps2 = go.Figure(data=[data], layout=layout) iplot(Choromaps2)

