**Cohort Analysis** In this assignment our job is to perform a cohort analysis on the customers of the company. We will group the customers into cohorts that are groups sharing common charecteristics. #loading the required packages In [22]: import pandas as pd import matplotlib.pyplot as plt import seaborn as sb # Load the datasets In [23]: online sales = pd.read csv('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Online discount coupons = pd.read csv('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Di tax\_amount = pd.read\_excel('C:\\Users\\sujoydutta\\Desktop\\Data analysis\\Projects\\Marketing insights\\Tax\_amount # Merging datasets using 'CustomerID' as the common key In [24]: merged data = pd.merge(online sales, discount coupons, on='Product Category', how='left') merged data = pd.merge(merged data, tax amount, on='Product Category', how='left') merged data.head() Out[24]: CustomerID Transaction\_ID Transaction\_Date Product\_SKU Product\_name Product\_Category Quantity Avg\_Price Delivery\_Charges Cc **Nest Learning** Thermostat 0 17850 16679 20190101 GGOENEBJ079499 Nest-USA 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Thermostat 1 17850 16679 20190101 GGOENEBJ079499 Nest-USA 1 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Thermostat 20190101 GGOENEBJ079499 2 17850 16679 Nest-USA 1 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Thermostat 3 17850 16679 20190101 GGOENEBJ079499 Nest-USA 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Thermostat 4 17850 16679 20190101 GGOENEBJ079499 Nest-USA 153.71 6.5 3rd Gen-USA -Stainle... #examining the merged dataset In [25]: merged data.head() Out[25]: CustomerID Transaction\_ID Transaction\_Date Product\_SKU Product\_name Product\_Category Quantity Avg\_Price Delivery\_Charges Cc **Nest Learning** Thermostat 0 17850 16679 20190101 GGOENEBJ079499 Nest-USA 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Thermostat Nest-USA 1 17850 16679 20190101 GGOENEBJ079499 1 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Thermostat 2 17850 16679 20190101 GGOENEBJ079499 Nest-USA 1 153.71 6.5 3rd Gen-USA -Stainle... **Nest Learning** Inermostat 3 153.71 17850 16679 20190101 GGOENEBJ079499 Nest-USA 6.5 3rd Gen-USA -Stainle... **Nest Learning Thermostat** 17850 16679 20190101 GGOENEBJ079499 Nest-USA 153.71 6.5 3rd Gen-USA -Stainle... #getting information of the dataset In [26]: merged data.info() <class 'pandas.core.frame.DataFrame'> Int64Index: 630688 entries, 0 to 630687 Data columns (total 14 columns): Non-Null Count Dtype # Column CustomerID CustomerID 630688 non-null int64
Transaction\_ID 630688 non-null int64 0 1 Transaction\_Date 630688 non-null int64 Product\_SKU 630688 non-null object
Product\_name 630688 non-null object Product\_Category 630688 non-null object Quantity 630688 non-null int64 Avg Price 630688 non-null floate Avg Price 630688 non-null float64 Delivery\_Charges 630688 non-null float64 Coupon\_Status 630688 non-null object 10 Month b30200 Non Null object 630288 non-null object float6 12 Discount pct 630288 non-null float64 13 GST 630688 non-null float64 dtypes: float64(4), int64(4), object(6) memory usage: 72.2+ MB In [27]: # Calculating Invoice Value for each transaction in the merged dataset merged data['Invoice\_Value'] = ( (merged\_data['Quantity'] \* merged\_data['Avg\_Price']) \* (1 - merged\_data['Discount\_pct']) \* (1 + merged\_data['GST']) + merged\_data['Delivery\_Charges'] ).abs() In [31]: # Extracting relevant columns cohort\_data = merged\_data[['CustomerID', 'Transaction\_Date', 'Invoice\_Value']] cohort\_data.head() CustomerID Transaction\_Date Invoice\_Value Out[31]: 0 17850 20190101 1515.229 1 17850 20190101 3206.039 2 17850 20190101 4896.849 3 17850 20190101 1515.229 4 17850 20190101 3206.039 In [32]: # Convert 'Transaction\_Date' in DataFrame to datetime data type cohort\_data['Transaction\_Date'] = pd.to\_datetime(merged\_data['Transaction\_Date'], format='%Y%m%d') cohort\_data['Transaction\_Date'] C:\Users\sujoydutta\AppData\Local\Temp\ipykernel 6844\4112860847.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row\_indexer,col\_indexer] = value instead See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user guide/indexing.html#ret urning-a-view-versus-a-copy cohort\_data['Transaction\_Date'] = pd.to\_datetime(merged\_data['Transaction\_Date'], format='%Y%m%d') 2019-01-01 Out[32]: 2019-01-01 1 2019-01-01 2 3 2019-01-01 4 2019-01-01 630683 2019-12-31 630684 2019-12-31 2019-12-31 630685 2019-12-31 630686 2019-12-31 630687 Name: Transaction Date, Length: 630688, dtype: datetime64[ns] In [33]: # Calculating the first purchase month for each customer cohort data['FirstPurchaseMonth'] = cohort data.groupby('CustomerID')['Transaction Date'].transform('min').dt.te cohort data['FirstPurchaseMonth'] C:\Users\sujoydutta\AppData\Local\Temp\ipykernel 6844\1830222443.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row indexer, col indexer] = value instead See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user guide/indexing.html#ret urning-a-view-versus-a-copy cohort data['FirstPurchaseMonth'] = cohort data.groupby('CustomerID')['Transaction Date'].transform('min').d t.to period('M') 2019-01 Out[33]: 2019-01 2 2019-01 2019-01 3 2019-01 . . . 2019-12 630683 630684 2019-12 630685 2019-12 630686 2019-12 630687 2019-12 Name: FirstPurchaseMonth, Length: 630688, dtype: period[M] In [34]: # Getting the cohort index based on the number of months since the first purchase cohort data['CohortIndex'] = ( (cohort data['Transaction Date'].dt.to period('M') - cohort data['FirstPurchaseMonth']).apply(lambda x: x.n cohort data['CohortIndex'] C:\Users\sujoydutta\AppData\Local\Temp\ipykernel 6844\747533931.py:2: SettingWithCopyWarning: A value is trying to be set on a copy of a slice from a DataFrame. Try using .loc[row\_indexer,col\_indexer] = value instead See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user guide/indexing.html#ret urning-a-view-versus-a-copy cohort data['CohortIndex'] = ( 1 Out[34]: 1 1 2 1 3 4 630683 630684 630685 1 630686 630687 Name: CohortIndex, Length: 630688, dtype: int64 In [35]: # Group the data by the first purchase month and cohort index cohort\_data = cohort\_data.groupby(['FirstPurchaseMonth', 'CohortIndex']).agg( TotalCustomers=('CustomerID', 'nunique'), TotalRevenue=('Invoice\_Value', 'sum') ).reset index() cohort\_data.round() Out[35]: FirstPurchaseMonth CohortIndex TotalCustomers TotalRevenue 0 2019-01 101071550.0 1 215 9882640.0 2019-01 13 2 2019-01 3 24 11300315.0 28921521.0 2019-01 7263371.0 73 2019-10 2390306.0 2019-10 682336.0 75 2019-11 1 68 53840226.0 2019-11 1102275.0 76 77 2019-12 1 106 61584100.0 78 rows × 4 columns # Creating a pivot table to represent the cohort analysis In [36]: cohorts = cohort\_data.pivot\_table( index=['FirstPurchaseMonth'], columns='CohortIndex', values='TotalCustomers' cohorts Out[36]: CohortIndex 2 3 5 6 8 10 11 12 FirstPurchaseMonth **2019-01** 215.0 34.0 23.0 44.0 35.0 47.0 23.0 28.0 20.0 34.0 13.0 24.0 2019-02 96.0 7.0 9.0 16.0 17.0 22.0 19.0 15.0 12.0 11.0 16.0 NaN 2019-03 18.0 32.0 22.0 19.0 177.0 35.0 25.0 33.0 22.0 15.0 NaN NaN 2019-04 163.0 14.0 24.0 24.0 18.0 15.0 10.0 16.0 12.0 NaN NaN NaN 112.0 10.0 12.0 9.0 13.0 13.0 14.0 8.0 NaN NaN NaN NaN 20.0 2019-06 137.0 22.0 12.0 11.0 14.0 11.0 NaN NaN NaN NaN NaN 2019-07 94.0 13.0 4.0 6.0 11.0 9.0 NaN NaN NaN NaN NaN NaN 2019-08 135.0 14.0 15.0 10.0 8.0 NaN NaN NaN NaN NaN NaN NaN 2019-09 78.0 6.0 3.0 2.0 NaN NaN NaN NaN NaN NaN NaN NaN 2019-10 87.0 6.0 4.0 NaN NaN NaN NaN NaN NaN NaN NaN NaN 2019-11 68.0 7.0 NaN 2019-12 106.0 NaN In [37]: # Calculating the retention rates for each cohort over time cohort size = cohorts.iloc[:, 0] retention = cohorts.divide(cohort size, axis=0) retention CohortIndex 2 3 5 6 7 8 9 10 11 12 Out[37]: FirstPurchaseMonth **2019-01** 1.0 0.060465 0.111628 0.158140 0.106977 0.204651 0.162791 0.218605 0.106977 0.130233 0.093023 0.15814 0.072917 0.093750 0.166667 0.177083 0.229167 0.197917 0.156250 0.125000 0.114583 0.166667 2019-02 1.0 NaN 2019-03 0.101695 0.197740 0.141243 0.180791 0.186441 0.124294 0.124294 0.084746 0.107345 1.0 NaN NaN 2019-04 0.061350 0.098160 0.073620 0.085890 0.147239 0.147239 0.110429 0.092025 1.0 NaN NaN NaN 2019-05 1.0 0.107143 0.080357 0.116071 0.089286 0.116071 0.125000 0.071429 NaN NaN NaN NaN 0.080292 2019-06 1.0 NaN NaN NaN NaN NaN **2019-07** 1.0 0.138298 0.042553 0.063830 0.117021 0.095745 NaN NaN NaN NaN NaN NaN 2019-08 0.103704 0.111111 0.074074 0.059259 1.0 NaN NaN NaN NaN NaN NaN NaN 1.0 0.076923 0.038462 0.025641 2019-09 NaN NaN NaN NaN NaN NaN NaN NaN **2019-10** 1.0 0.068966 0.045977 NaN NaN NaN NaN NaN NaN NaN NaN NaN **2019-11** 1.0 0.102941 NaN **2019-12** 1.0 NaN # Creating a heatmap to visualize cohort retention rates In [38]: plt.figure(figsize=(12, 8)) plt.title('Cohort Analysis - Customer Retention Rate') sb.heatmap(retention, annot=True, fmt='.0%', cmap='YlGnBu', vmin=0, vmax=1, cbar=False) plt.xticks(1 + retention.columns.values, [f'Month {i}' for i in range(1, len(retention.columns) + 1)]) plt.xlabel('Cohort Index') plt.ylabel('Cohort Start Month') plt.show() Cohort Analysis - Customer Retention Rate 2019-01 -100% 6% 16% 16% 11% 11% 20% 16% 22% 13% 9% 11% 2019-02 -100% 17% 17% 18% 23% 16% 12% 11% 2019-03 -100% 10% 20% 14% 18% 19% 12% 12% 8% 11% 2019-04 -100% 10% 15% 15% 11% 6% 7% 2019-05 -100% 11% 8% 12% 9% 12% 12% 7% Cohort Start Month 2019-06 -100% 15% 9% 8% 10% 8% 12% 2019-07 -100% 6% 10% 14% 4% 2019-08 -100% 2019-09 -100% 8% 4% 3% 2019-10 -100% 5% 2019-11 -100% 10% 2019-12 -100% Month 1 Month 2 Month 3 Month 4 Month 5 Month 6 Month 7 Month 8 Month 9 Month 10 Month 11 Month 12 Cohort Index # Remove rows with NaN values from the cohort table In [39]: cohort data = cohort data.dropna() # Reset the index of the DataFrame cohort data.reset index(drop=True, inplace=True) #viewing the cleaned cohort data In [40]: cohort\_data.round() Out[40]: FirstPurchaseMonth CohortIndex TotalCustomers TotalRevenue 0 2019-01 101071550.0 215 2019-01 2 13 9882640.0 1 2 2019-01 3 11300315.0 24 28921521.0 3 2019-01 34 2019-01 4 5 23 7263371.0 **73** 2019-10 2 2390306.0 2019-10 682336.0 **75** 2019-11 68 53840226.0 2019-11 76 7 1102275.0 77 1 2019-12 106 61584100.0  $78 \text{ rows} \times 4 \text{ columns}$ # Cohort metrics table In [41]: cohort\_metrics = cohort\_data.pivot\_table(index=[ 'FirstPurchaseMonth'], columns='CohortIndex', values=['TotalCu cohort\_metrics TotalCustomers ... Out[41]: CohortIndex 1 2 3 5 6 7 8 9 10 ... 3 5 FirstPurchaseMonth **2019-01** 215.0 13.0 34.0 23.0 44.0 35.0 47.0 23.0 28.0 ... 1.130031e+07 2.892152e+07 7.263371e+06 1.303335e+07 24.0 2019-02 96.0 7.0 22.0 12.0 11.0 ... 3.185927e+06 5.464640e+06 5.901013e+06 1.044326e+07 9.0 16.0 17.0 19.0 15.0 **2019-03** 177.0 18.0 19.0 ... 9.076607e+06 7.720443e+06 1.149867e+07 1.752918e+07 35.0 25.0 32.0 33.0 22.0 22.0 15.0 2019-04 163.0 14.0 24.0 24.0 18.0 15.0 10.0 16.0 12.0 NaN ... 4.195253e+06 7.428573e+06 6.280578e+06 7.573408e+06 **2019-05** 112.0 12.0 9.0 13.0 10.0 13.0 14.0 8.0 ... 2.982523e+06 3.534678e+06 4.296835e+06 8.957424e+06 NaN NaN 2019-06 137.0 20.0 22.0 11.0 NaN ... 3.194789e+06 3.781086e+06 2.510709e+06 9.131961e+06 12.0 14.0 11.0 NaN NaN 2019-07 94.0 ... 1.810631e+06 3.654439e+06 5.981508e+06 7.220105e+06 13.0 4.0 6.0 11.0 9.0 NaN NaN NaN NaN 2019-08 135.0 10.0 2.741047e+06 7.144423e+06 4.141043e+06 14.0 15.0 8.0 NaN NaN NaN NaN NaN NaN 2019-09 78.0 6.0 3.0 2.0 NaN NaN NaN NaN NaN NaN ... 6.298675e+05 1.611835e+05 NaN NaN 2019-10 87.0 6.823357e+05 6.0 4.0 NaN 2019-11 NaN 68.0 7.0 NaN **2019-12** 106.0 NaN 12 rows × 24 columns # Making a heatmap for retention rate In [42]: plt.figure(figsize=(20,15)) sb.heatmap(cohort\_metrics['TotalCustomers'], annot=True, fmt='.0%', cmap='YlGnBu') plt.title('Cohort Analysis - Retention Rate') plt.xlabel('Cohort Index') plt.ylabel('Cohort Month') plt.show() Cohort Analysis - Retention Rate 2019-01 21500% 1300% 2400% 3400% 2300% 4700% 2800% 2000% 3400% 4400% 2300% - 200 2019-02 9600% 1900% 1500% 700% 900% 1600% 1700% 2200% 1200% 1100% 1600% - 175 2019-03 17700% 1800% 3500% 2500% 3200% 3300% 2200% 2200% 1500% 1900% 2019-04 1400% 2400% 2400% 1800% 1500% 1000% 1600% 1200% - 150 2019-05 1200% 1300% 1000% 1300% 1400% 800% - 125 2000% 2200% 1200% 1100% 1400% 1100% 201 Cohort Mont 2019-07 9400% 1300% 400% 600% 1100% 900% - 100 2019-08 1400% 800% 1500% 1000% - 75 2019-09 7800% 600% 200% 300% - 50 2019-10 8700% 600% 400% 2019-11 6800% 700% 2019-12 'n 'n 12 In [43]: # Creating a heatmap for average revenue plt.figure(figsize=(20, 10)) sb.heatmap(cohort\_metrics['TotalRevenue'], annot=True, fmt='.0f', cmap='YlGnBu') plt.title('Cohort Analysis - Average Revenue') plt.xlabel('Cohort Index') plt.ylabel('Cohort Month') plt.show() Cohort Analysis - Average Revenue 1e8 2019-12 2019-11 2019-10 2019-09 2019-08 2019-07 2019-06 2019-05 2019-04 2019-03 2019-02 2019-01 101071550 9882640 11300315 28921521 7263371 10111802 11375873 23352836 13033347 21082388 12573827 10053170 2037764 3185927 5464640 5901013 10443259 5531927 11316199 4255940 7150164 11861084 10294541 9076607 7720443 11498671 17529176 12169138 9870306 6128743 9070328 0.8 5936516 4195253 7428573 6280578 7573408 3899688 14047450 4507513 1738670 2982523 3534678 4296835 8957424 9425838 2844875 2642818 3194789 3781086 2510709 9131961 3841362 37531615 3637098 1810631 5981508 7220105 3654439 0.4 48197728 2651899 2741047 7144423 4141043 36771104 483680 629868 161184 2390306 682336 0.2 1102275 'n **Remark:** The first cohort is by far the best performing cohort in terms of retention rate and revenue.