You are Welcome to My Projects

환영합니다

I am learnig my best to find a new way
If any Mistake Please Suggest me.Mail me (mailto:krishdb38@gmail.comSubject=ThankYou%20VeryMuch)

Krishna 크리스나

My Git Hub Repository

(https://www.github.com/krishdb38/portfolio)

(https://www.github.com/krishdb38/portfolio)
(https://www.github.com/krishdb38/portfolio)

In [5]:
```
%reset -f
```

1.Import Necessary Library

In [6]:
```
import pandas as pd
import numpy as np
import os

import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

In [9]:
```
df=pd.read_csv("ebay.csv") #파일 읽는다 (Loading the file as a data frame )
```

Out[9]:
```
Index(['Category', 'City', 'Country', 'Customer Name', 'Discount',
       'Number of Records', 'Order Date', 'Order ID', 'Postal Code',
       'Manufacturer', 'Product Name', 'Profit', 'Quantity', 'Region', 'Sales',
       'Segment', 'Ship Date', 'Ship Mode', 'State', 'Sub-Category'],
      dtype='object')
```

In [10]:
```
df.size #size of data frame
```

Out[10]:
```
199880
```

In [11]:
```
df.shape  #size 2004 rows and 20 Columns
```

Out[11]:
```
(9994, 20)
```

In [12]:
```
df.columns #list the details of Colun
```

Out[12]:
```
Index(['Category', 'City', 'Country', 'Customer Name', 'Discount',
       'Number of Records', 'Order Date', 'Order ID', 'Postal Code',
       'Manufacturer', 'Product Name', 'Profit', 'Quantity', 'Region', 'Sales',
       'Segment', 'Ship Date', 'Ship Mode', 'State', 'Sub-Category'],
      dtype='object')
```

In [15]:
```
df["Category"].head()     #Verifying data from one Column
```

Out[15]:
```
0         Furniture
1         Furniture
2   Office Supplies
3         Furniture
4   Office Supplies
Name: Category, dtype: object
```

In [3]:
```
df.info()  #All Information about data and data type
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9994 entries, 0 to 9993
Data columns (total 20 columns):
Category             9994 non-null object
City                 9994 non-null object
Country              9994 non-null object
Customer Name        9994 non-null object
Discount             9994 non-null float64
Number of Records    9994 non-null int64
Order Date           9994 non-null object
Order ID             9994 non-null object
Postal Code          9983 non-null float64
Manufacturer         9994 non-null object
Product Name         9994 non-null object
Profit               9994 non-null float64
Quantity             9994 non-null int64
Region               9994 non-null object
Sales                9994 non-null float64
Segment              9994 non-null object
Ship Date            9994 non-null object
Ship Mode            9994 non-null object
State                9994 non-null object
Sub-Category         9994 non-null object
dtypes: float64(4), int64(2), object(14)
memory usage: 1.5+ MB
```

(https://www.github.com/krishdb38/portfolio)

(https://www.github.com/krishdb38/portfolio)
(https://www.github.com/krishdb38/portfolio)

(https://www.github.com/krishdb38/portfolio)

In the above we can see object 14 categories in object float type is 4 int type 2 (https://www.github.com/krishdb38/portfolio)

In [ ]:
```
def row_print(col_name):
    return df[col_name][0:10]
def data_describe(col_name):
    return df[col_name].describe()
```

In [ ]:
```
#df["Sales"].describe()      #This descrive the Sales in one line
data_describe("Sales")       #Same Thing Using Function
```

In [ ]:
```
#df["City"]    #You Can check Value in City Column
```

Col describe data type object must be changed to Suitable Data Format

In [4]:
```
df.isnull().sum() #Checking which Column has nan (Non Availiable Value)
```

Out[4]:
```
Category             0
City                 0
Country              0
Customer Name        0
Discount             0
Number of Records    0
Order Date           0
Order ID             0
Postal Code         11
Manufacturer         0
Product Name         0
Profit               0
Quantity             0
Region               0
Sales                0
Segment              0
Ship Date            0
Ship Mode            0
State                0
Sub-Category         0
dtype: int64
```

In the Postal Code Column there are 11 nan value

In [23]:
```
df["Order Date"].describe() #Checking the data type of "Order Date"
```

Out[23]:
```
count                    9994
unique                   1236
top       2017-09-05 00:00:00
freq                       38
first     2015-01-03 00:00:00
last      2018-12-30 00:00:00
Name: Order Date, dtype: object
```

Convert to Date format

In [21]:
```
#row_print("Order Date")
df["Order Date"]=pd.to_datetime(df["Order Date"])#Pandas to_datetime() method helps to#
#convert string Date time into Python Date time object.
```

In [24]:
```
df["Order Date"].describe()
#df["Order Date"].head()
```

Out[24]:
```
count                    9994
unique                   1236
top       2017-09-05 00:00:00
freq                       38
first     2015-01-03 00:00:00
last      2018-12-30 00:00:00
Name: Order Date, dtype: object
```

Check every Column and COnverting to best format

In [25]:
```
#df["Ship Date"]  #chek the date format

df["Ship Date"].describe()
```

Out[25]:
```
count                    9994
unique                   1334
top       2016-12-16 00:00:00
freq                       35
first     2015-01-07 00:00:00
last      2019-01-05 00:00:00
Name: Ship Date, dtype: object
```

In [20]:
```
df["Ship Date"].head(3) #chek the date format Before Changing format
```

Out[20]:
```
0   2017-11-11
1   2017-11-11
2   2017-06-16
Name: Ship Date, dtype: datetime64[ns]
```

Ship Date is in object(string) format to we need to convert in to Date Format

In [26]:
```
#df["Ship Date"]
#change date format into String Format
df["Ship Date"]=pd.to_datetime(df["Ship Date"])
df["Ship Date"].head(3)
```

Out[26]:
```
0   2017-11-11
1   2017-11-11
2   2017-06-16
Name: Ship Date, dtype: datetime64[ns]
```

In [18]:
```
df["Ship Date"].describe()
```

Out[18]:
```
count                    9994
unique                   1334
top       2016-12-16 00:00:00
freq                       35
first     2015-01-07 00:00:00
last      2019-01-05 00:00:00
Name: Ship Date, dtype: object
```

We Converted all Date(object) format to datetype format

Now we check other column

In [ ]:
```

```

In [29]:
```
#df["Ship Mode"].head(10)
df["Ship Mode"].describe()
#List unique values in the df['name'] column
```

Out[29]:
```
count             9994
unique               4
top       Standard Class
freq              5968
Name: Ship Mode, dtype: object
```

In [30]:
```
df["Ship Mode"].unique()
```

Out[30]:
```
array(['Second Class', 'Standard Class', 'First Class', 'Same Day'],
      dtype=object)
```

In [33]:
```
df.City.unique() #All the Unique values in City Column
```

Out[33]:
```
array(['Henderson', 'Los Angeles', 'Fort Lauderdale', 'Concord',
       'Seattle', 'Fort Worth', 'Madison', 'West Jordan', 'San Francisco',
       'Fremont', 'Philadelphia', 'Orem', 'Houston', 'Richardson',
       'Naperville', 'Melbourne', 'Eagan', 'Westland', 'Dover',
       'New Albany', 'New York City', 'Troy', 'Chicago', 'Gilbert',
       'Springfield', 'Jackson', 'Memphis', 'Decatur', 'Durham',
       'Columbia', 'Rochester', 'Minneapolis', 'Portland', 'Saint Paul',
       'Aurora', 'Charlotte', 'Orland Park', 'Urbandale', 'Columbus',
       'Bristol', 'Wilmington', 'Bloomington', 'Phoenix', 'Roseville',
       'Independence', 'Pasadena', 'Newark', 'Franklin', 'Scottsdale',
       'San Jose', 'Edmond', 'Carlsbad', 'San Antonio', 'Monroe',
       'Fairfield', 'Grand Prairie', 'Redlands', 'Hamilton', 'Westfield',
       'Akron', 'Denver', 'Dallas', 'Whittier', 'Saginaw', 'Medina',
       'Dublin', 'Detroit', 'Tampa', 'Santa Clara', 'Lakeville',
       'San Diego', 'Brentwood', 'Chapel Hill', 'Morristown',
       'Cincinnati', 'Inglewood', 'Tamarac', 'Colorado Springs',
       'Belleville', 'Taylor', 'Lakewood', 'Arlington', 'Arvada',
       'Hackensack', 'Saint Petersburg', 'Long Beach', 'Hesperia',
       'Murfreesboro', 'Layton', 'Austin', 'Lowell', 'Manchester',
       'Harlingen', 'Tucson', 'Quincy', 'Pembroke Pines', 'Des Moines',
       'Peoria', 'Las Vegas', 'Warwick', 'Miami', 'Huntington Beach',
       'Richmond', 'Louisville', 'Lawrence', 'Canton', 'New Rochelle',
       'Gastonia', 'Jacksonville', 'Auburn', 'Norman', 'Park Ridge',
       'Amarillo', 'Lindenhurst', 'Huntsville', 'Fayetteville',
       'Costa Mesa', 'Parker', 'Atlanta', 'Gladstone', 'Great Falls',
       'Lakeland', 'Montgomery', 'Mesa', 'Green Bay', 'Anaheim',
       'Marysville', 'Salem', 'Laredo', 'Grove City', 'Dearborn',
       'Warner Robins', 'Vallejo', 'Mission Viejo', 'Rochester Hills',
       'Plainfield', 'Sierra Vista', 'Vancouver', 'Cleveland', 'Tyler',
       'Burlington', 'Waynesboro', 'Chester', 'Cary', 'Palm Coast',
       'Mount Vernon', 'Hialeah', 'Oceanside', 'Evanston', 'Trenton',
       'Cottage Grove', 'Bossier City', 'Lancaster', 'Asheville',
       'Lake Elsinore', 'Omaha', 'Edmonds', 'Santa Ana', 'Milwaukee',
       'Florence', 'Lorain', 'Linden', 'Salinas', 'New Brunswick',
       'Garland', 'Norwich', 'Alexandria', 'Toledo', 'Farmington',
       'Riverside', 'Torrance', 'Round Rock', 'Boca Raton',
       'Virginia Beach', 'Oklahoma City', 'Kirkwood', 'La Porte',
       'Lansing', 'El Paso', 'Mansfield', 'Des Plaines', 'Freeport',
       'Perth Amboy', 'Watertown', 'Waterbury', 'Andover', 'Clifton',
       'Clinton', 'Baltimore', 'Everett', 'Buffalo', 'Parma', 'Bethlehem',
       'Lafayette', 'Mobile', 'Murray', 'Coral Springs', 'Knoxville',
       'Bakersfield', 'Oakland', 'Draper', 'Lake Forest', 'Pocatello',
       'Hillsboro', 'Greeley', 'Longmont', 'Encinitas', 'Mount Pleasant',
       'Lawton', 'Greenwood', 'Saint Charles', 'Skokie', 'Lubbock',
       'Middletown', 'Marion', 'Wheeling', 'Hampton', 'Boynton Beach',
       'Apopka', 'Pomona', 'Glendale', 'Tulsa', 'San Angelo',
       'League City', 'Carrollton', 'Frisco', 'Beaumont', 'Paterson',
       'Cranston', 'Woonsocket', 'Vineland', 'Reading', 'Lake Charles',
       'Sandy Springs', 'Harrisonburg', 'Tallahassee', 'Raleigh',
       'Fresno', 'Olympia', 'North Las Vegas', 'Longview', 'Bellingham',
       'Sacramento', 'Thornton', 'Fort Collins', 'Littleton',
       'Dearborn Heights', 'Kenosha', 'Midland', 'Eau Claire', 'Wichita',
       'Mishawaka', 'Texas City', 'Carol Stream', 'Allen',
       'College Station', 'Plano', 'Highland Park', 'Pharr',
       'Brownsville', 'Waco', 'Buffalo Grove', 'Rockford', 'Champaign',
       'Yonkers', 'Rockville', 'Washington', 'Marlborough', 'Hempstead',
       'Allentown', 'Passaic', 'Southaven', 'Conway', 'Jonesboro',
```

```
'Athens', 'Georgetown', 'Chesapeake', 'Hattiesburg', 'Macon',
'Little Rock', 'Greenville', 'Chattanooga', 'Nashville', 'Deltona',
'Hollywood', 'Miramar', 'Wilson', 'Hendersonville', 'Greensboro',
'Daytona Beach', 'Westminster', 'Bellevue', 'Spokane',
'Moreno Valley', 'Kent', 'Citrus Heights', 'Oxnard', 'Las Cruces',
'Provo', 'Morgan Hill', 'San Bernardino', 'Redmond', 'Modesto',
'Bullhead City', 'Broomfield', 'Rio Rancho', 'Gresham', 'Tempe',
'Apple Valley', 'Ontario', 'Pasco', 'Garden City', 'Indianapolis',
'Saint Louis', 'Frankfort', 'The Colony', 'Edinburg', 'Abilene',
'Providence', 'East Orange', 'Meriden', 'Suffolk',
'Charlottesville', 'Woodstock', 'Goldsboro', 'Delray Beach',
'Pensacola', 'Redondo Beach', 'Santa Maria', 'San Mateo',
'Commerce City', 'Temecula', 'Portage', 'Bolingbrook', 'Milford',
'Orange', 'Utica', 'Kenner', 'Rock Hill', 'Homestead',
'Plantation', 'Chula Vista', 'Pueblo', 'South Bend', 'Elkhart',
'Noblesville', 'Holland', 'Grand Island', 'Fargo', 'La Crosse',
'Aberdeen', 'Keller', 'Irving', 'Romeoville', 'Hagerstown',
'Nashua', 'Bowling Green', 'Cuyahoga Falls', 'Pine Bluff',
'North Charleston', 'Jupiter', 'Port Orange', 'Johnson City',
'North Miami', 'Port Saint Lucie', 'Clarksville', 'Chico',
'Albuquerque', 'Stockton', 'Rancho Cucamonga', 'Pleasant Grove',
'Escondido', 'Sparks', 'Vacaville', 'Avondale', 'Salt Lake City',
'Broken Arrow', 'Sterling Heights', 'Royal Oak', 'Wausau',
'Norfolk', 'Waterloo', 'Coon Rapids', 'Grand Rapids', 'Waukesha',
'New Castle', 'Sioux Falls', 'Mcallen', 'Coppell', 'Port Arthur',
'Cedar Hill', 'Oswego', 'Arlington Heights', 'Rome', 'Leominster',
'Malden', 'Roswell', 'Newport News', 'Smyrna', 'Owensboro',
'Marietta', 'Rogers', 'Gulfport', 'Orlando', 'Margate',
'Thomasville', 'West Palm Beach', 'Lebanon', 'Bartlett',
'Ormond Beach', 'Hickory', 'Logan', 'Visalia', 'San Luis Obispo',
'San Clemente', 'Camarillo', 'Murrieta', 'Davis', 'Santa Barbara',
'Lodi', 'Renton', 'Lewiston', 'Medford', 'Chandler', 'Missoula',
'Loveland', 'Englewood', 'Tigard', 'Clovis', 'Cedar Rapids',
'Pearland', 'Niagara Falls', 'Boise', 'Thousand Oaks', 'Glenview',
'Moorhead', 'Dubuque', 'Appleton', 'New Bedford', 'Jamestown',
'Hot Springs', 'Kissimmee', 'Santa Fe', 'Woodland', 'Redding',
'Rapid City', 'Odessa', 'Tinley Park', 'Mason', 'Sunnyvale',
'West Allis', 'San Marcos', 'Corpus Christi', 'Covington',
'Eugene', 'Oak Park', 'Lincoln Park', 'Saint Cloud', 'Bryan',
'Mesquite', 'Revere', 'Bangor', 'Laurel', 'Gaithersburg', 'York',
'East Point', 'Texarkana', 'Coral Gables', 'Pompano Beach',
'Twin Falls', 'Caldwell', 'El Cajon', 'Overland Park',
'Haltom City', 'Bedford', 'Conroe', 'Deer Park', 'Altoona',
'Hoover', 'Antioch', 'Helena', 'Billings', 'Iowa City',
'Ann Arbor', 'Atlantic City', 'Elyria', 'Summerville',
'Laguna Niguel', 'Yuma', 'Maple Grove', 'Woodbury', 'Baytown',
'Grapevine', 'Missouri City', 'Bridgeton', 'Beverly', 'Reno',
'Coachella', 'Yucaipa', 'Meridian', 'Redwood City', 'Palatine',
'La Quinta', 'Montebello', 'Lehi', 'Pico Rivera', 'Muskogee',
'Mentor', 'Manteca', 'Danville', 'Olathe', 'Jefferson City',
'Saint Peters', 'Superior', 'Normal', 'Elmhurst', 'Cambridge',
'Holyoke', 'Danbury', 'Tuscaloosa', 'Sanford', 'Burbank',
'San Gabriel', 'Cheyenne', 'Shelton', 'Bayonne', 'Sheboygan',
'Springdale', 'Bozeman', 'La Mesa', 'Manhattan'], dtype=object)
```

In [ ]:

We Can see Unique value in State Column

In [45]:
```
df.State.unique()
unique_state=df.State.unique()
unique_state
```

Out[45]:
```
array(['Kentucky', 'California', 'Florida', 'North Carolina',
       'Washington', 'Texas', 'Wisconsin', 'Utah', 'Nebraska',
       'Pennsylvania', 'Illinois', 'Minnesota', 'Michigan', 'Delaware',
       'Indiana', 'New York', 'Arizona', 'Virginia', 'Tennessee',
       'Alabama', 'South Carolina', 'Oregon', 'Colorado', 'Iowa', 'Ohio',
       'Missouri', 'Oklahoma', 'New Mexico', 'Louisiana', 'Connecticut',
       'New Jersey', 'Massachusetts', 'Georgia', 'Nevada', 'Rhode Island',
       'Mississippi', 'Arkansas', 'Montana', 'New Hampshire', 'Maryland',
       'Idaho', 'West Virginia', 'Vermont', 'Kansas',
       'District of Columbia', 'North Dakota', 'South Dakota', 'Maine',
       'Wyoming'], dtype=object)
```

In [46]:
```
df.State.describe()
```

Out[46]:
```
count           9994
unique            49
top       California
freq            2001
Name: State, dtype: object
```

In [56]:
```
df.Country.unique()
```

Out[56]:
```
array(['United States'], dtype=object)
```

In teh Country Column Ther is one Country Country so delete Country Coulumn for fast Processing

In [62]:
```
df=df.drop("Country", axis=1)
```

In [63]:
```
df.columns
```

Out[63]:
```
Index(['Category', 'City', 'Customer Name', 'Discount', 'Number of Records',
       'Order Date', 'Order ID', 'Postal Code', 'Manufacturer', 'Product Name',
       'Profit', 'Quantity', 'Region', 'Sales', 'Segment', 'Ship Date',
       'Ship Mode', 'State', 'Sub-Category'],
      dtype='object')
```

In the above we can see Country is deleted

In [64]:
```
df.Region.describe()
#row_print("Region")   #By Using Function
```

Out[64]:
```
count     9994
unique       4
top       West
freq      3203
Name: Region, dtype: object
```

In [66]:
```
df.Region.unique() # Unique values in Region
```

Out[66]:
```
array(['South', 'West', 'Central', 'East'], dtype=object)
```

Check For Segment COlumn

In [67]:
```
df.Segment.describe()
```

Out[67]:
```
count         9994
unique           3
top       Consumer
freq          5191
Name: Segment, dtype: object
```

In the Segment Coumn unique variables are 3 Lets Check

In [68]:
```
df.Segment.unique()
```

Out[68]:
```
array(['Consumer', 'Corporate', 'Home Office'], dtype=object)
```

We Can find check 3 main Segment

In [69]:
```
df["Ship Mode"].describe()
```

Out[69]:
```
count              9994
unique                4
top       Standard Class
freq               5968
Name: Ship Mode, dtype: object
```

In [70]:
```
#in the above we can see 4 Unique Value
df["Ship Mode"].unique()
```

Out[70]:
```
array(['Second Class', 'Standard Class', 'First Class', 'Same Day'],
      dtype=object)
```

In [71]:
```
df["State"].describe()
```

Out[71]:
```
count           9994
unique            49
top       California
freq            2001
Name: State, dtype: object
```

In [73]:
```
#df["State"].unique()
```

In [74]:
```
df["State"].describe()
```

Out[74]:
```
count           9994
unique            49
top       California
freq            2001
Name: State, dtype: object
```

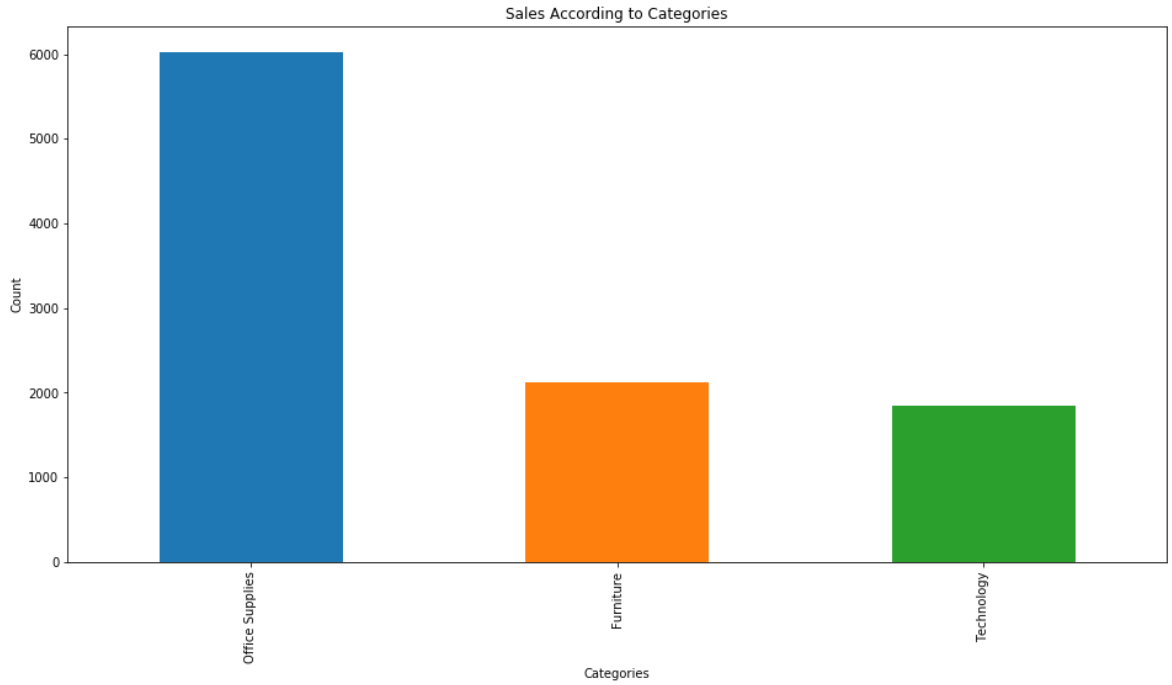
Part 2 Plotting In to graf Using Pandas

In [75]:
```
df.columns
```

Out[75]:
```
Index(['Category', 'City', 'Customer Name', 'Discount', 'Number of Records',
       'Order Date', 'Order ID', 'Postal Code', 'Manufacturer', 'Product Name',
       'Profit', 'Quantity', 'Region', 'Sales', 'Segment', 'Ship Date',
       'Ship Mode', 'State', 'Sub-Category'],
      dtype='object')
```

Histogram Table According to Categories

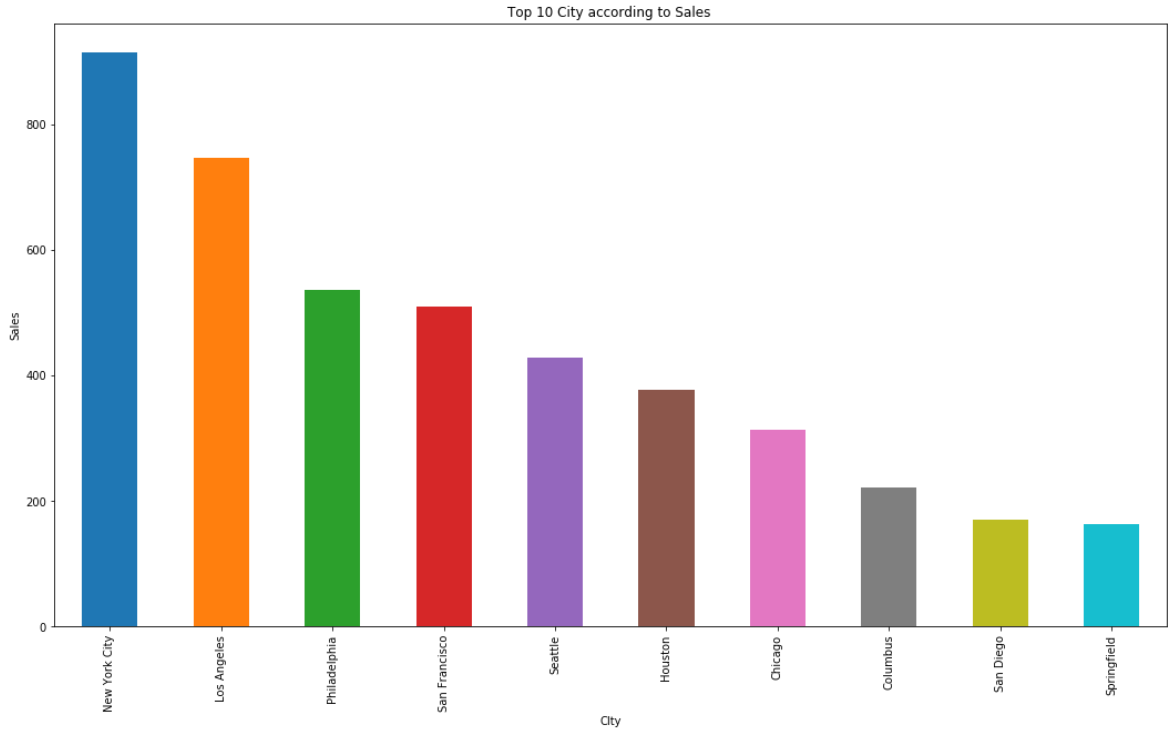
In [107]:
```
plt.figure(figsize=(16,8))
df["Category"].value_counts().plot.bar()
plt.title("Sales According to Categories")
plt.ylabel("Count")
plt.xlabel("Categories")
plt.show()
```



According to City Histogram

In [108]:
```
plt.figure(figsize=(18,10))

top_city=df.groupby("City")["Sales"].count().sort_values(ascending=False)
top_city=top_city[:10]
top_city.plot(kind="bar") #plot in Bar ,Other options are pie
plt.title("Top 10 City according to Sales ")
plt.ylabel("Sales")
plt.xlabel("City")
plt.show()
```



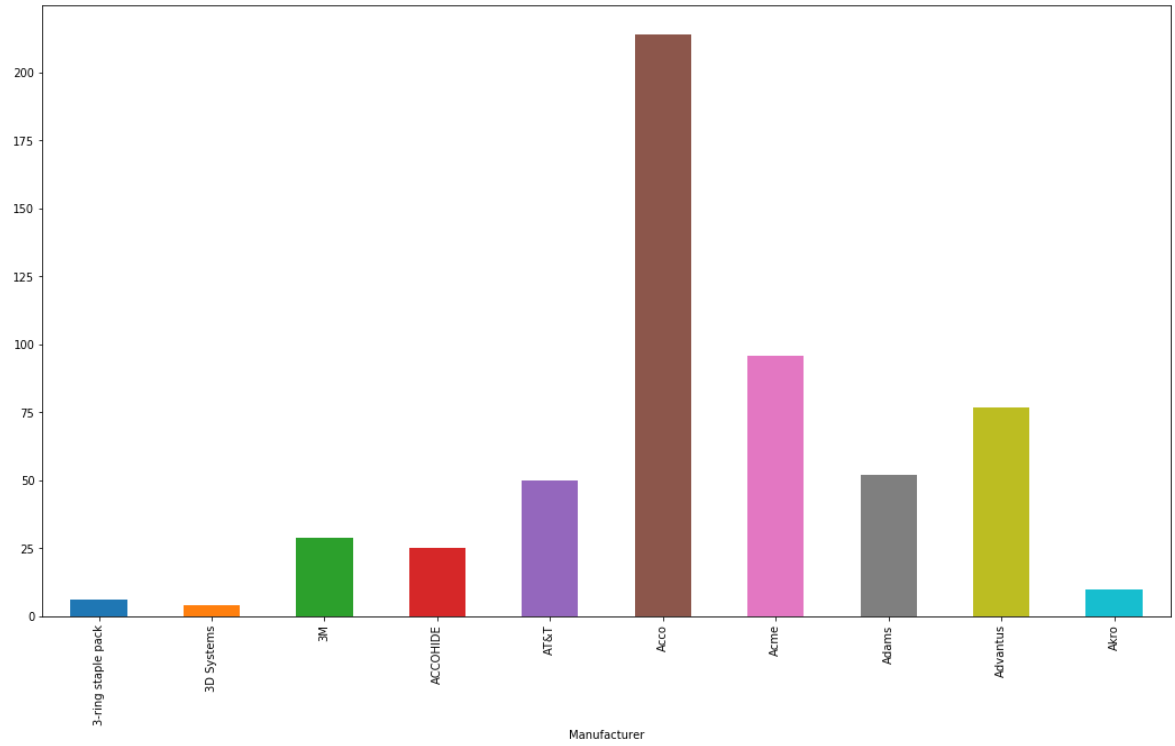
In [100]:
```
#print(top_city)
```

In [79]:
```
df.head(1)
```

Out[79]:

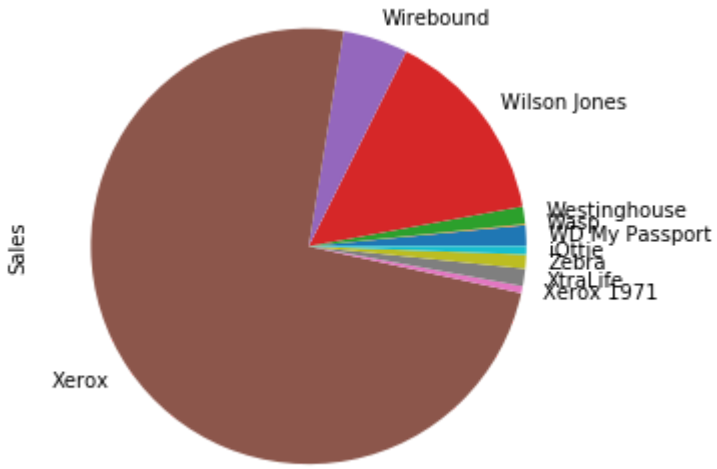
	Category	City	Customer Name	Discount	Number of Records	Order Date	Order ID	Postal Code	Manufacturer
0	Furniture	Henderson	Claire Gute	0.0	1	2017-11-08	CA-2017-152156	42420.0	Bush

In [116]:
```
plt.figure(figsize=(18,10))
manu=df.groupby("Manufacturer")["Sales"].count()
manu_top10=manu[:10] #Top 10
manu_last10=manu[-10:] #Last 10
#top_man=df.groupby("manu").count().plot.bar()
manu_top10.plot(kind="bar")
plt.show()
```



In [ ]:
```
#In the above Table Manufacturer Acco is first in Position and 3D system is in10 Position
```

In [122]:
```
plt.figure(figsize=(10,5))
manu_last10.plot(kind="pie")
plt.
plt.show()
```

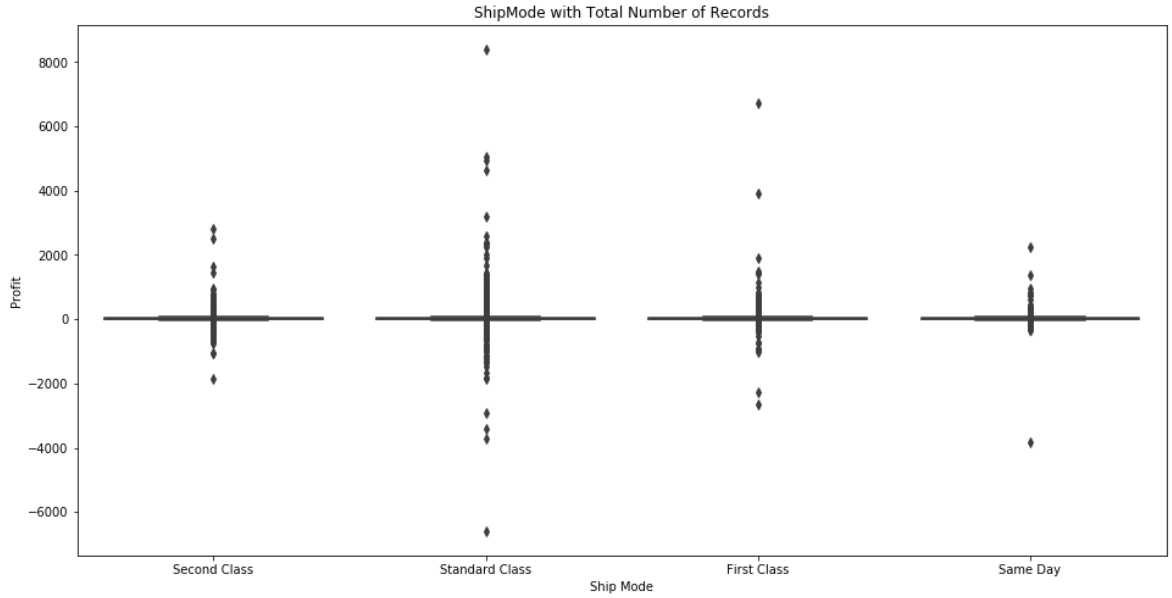


In [124]:
```
df.head(1)
```

Out[124]:

	Category	City	Country	Customer Name	Discount	Number of Records	Order Date	Order ID	Postal Code	M
0	Furniture	Henderson	United States	Claire Gute	0.0	1	11/8/2017	CA-2017-152156	42420.0	

In [132]:
```
plt.figure(figsize=(16,8))
sns.boxplot("Ship Mode","Profit",data=df)
plt.title("ShipMode with Total Number of Records")
plt.show()
```



In box plot we can see Standard Class has been sent more in sales

In [ ]:

In [ ]:

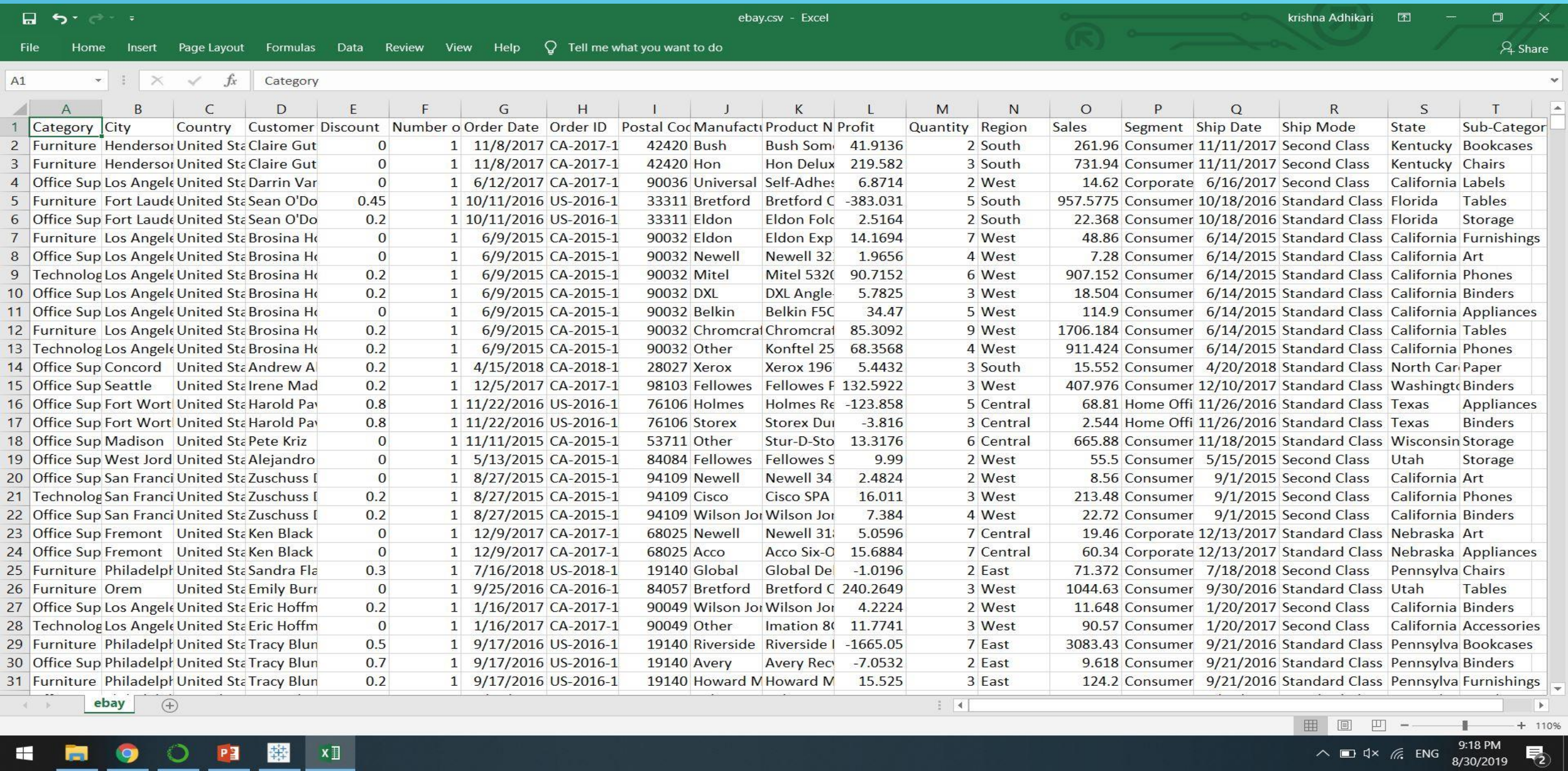
My Portfolio

크리스나

FILE CREATED ON: 8/30/2019 9:08:57 PM

TABLEAU SOFTWARE 태불로 사용 했음

SOURCE:EBAY.CSV DATA SOURCE=HTTPS://GITHUB.COM/KRISHDB38/PORTFOLIO.GIT



ebay.csv - Excel

krishna Adhikari

File Home Insert Page Layout Formulas Data Review View Help Tell me what you want to do Share

A1 Category

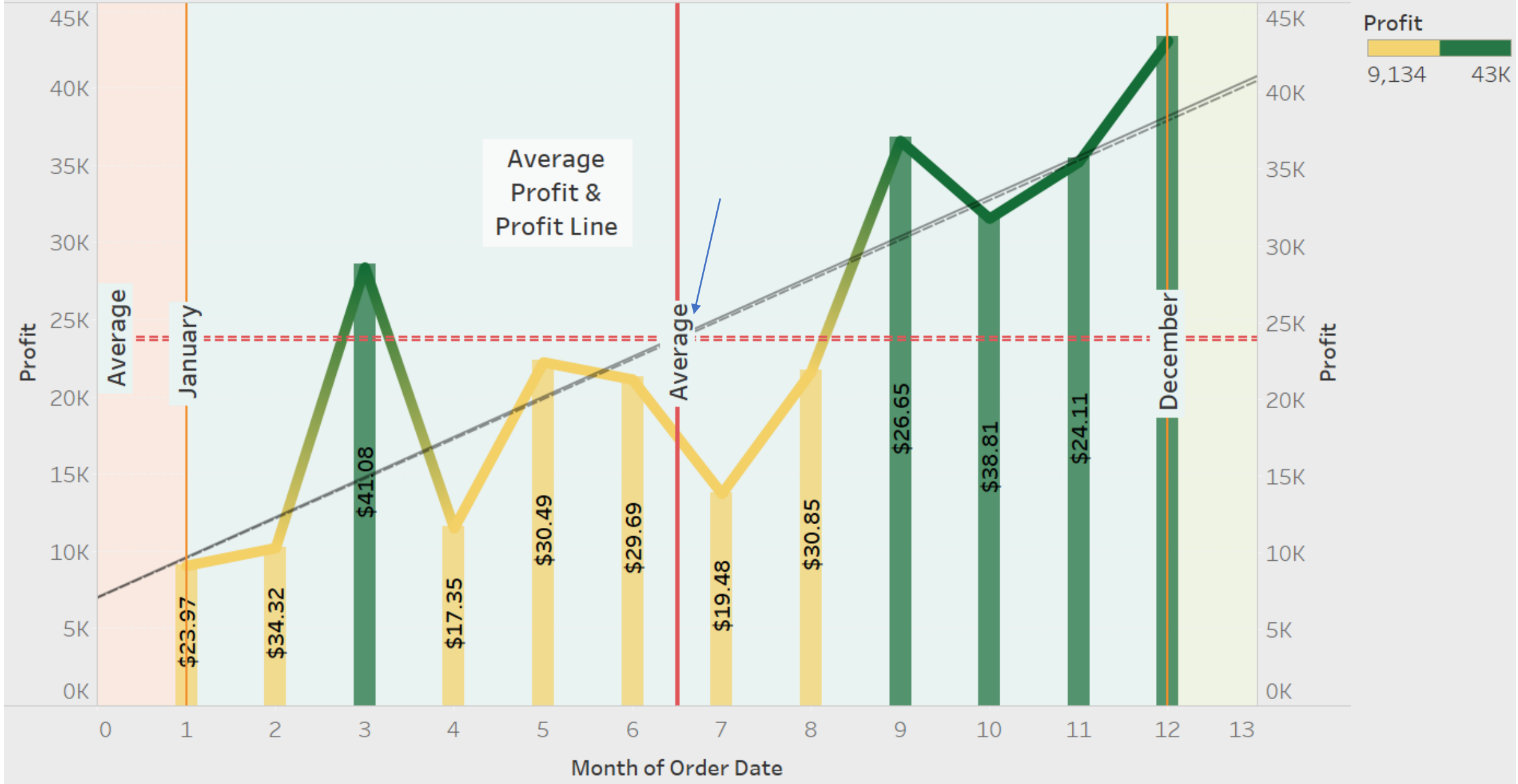
	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Category	City	Country	Customer	Discount	Number o	Order Date	Order ID	Postal Coc	Manufacti	Product N	Profit	Quantity	Region	Sales	Segment	Ship Date	Ship Mode	State	Sub-Categor
2	Furniture	Henderso	United Sta	Claire Gut	0	1	11/8/2017	CA-2017-1	42420	Bush	Bush Som	41.9136	2	South	261.96	Consumer	11/11/2017	Second Class	Kentucky	Bookcases
3	Furniture	Henderso	United Sta	Claire Gut	0	1	11/8/2017	CA-2017-1	42420	Hon	Hon Delux	219.582	3	South	731.94	Consumer	11/11/2017	Second Class	Kentucky	Chairs
4	Office Sup	Los Angele	United Sta	Darrin Var	0	1	6/12/2017	CA-2017-1	90036	Universal	Self-Adhes	6.8714	2	West	14.62	Corporate	6/16/2017	Second Class	California	Labels
5	Furniture	Fort Laude	United Sta	Sean O'Do	0.45	1	10/11/2016	US-2016-1	33311	Bretford	Bretford C	-383.031	5	South	957.5775	Consumer	10/18/2016	Standard Class	Florida	Tables
6	Office Sup	Fort Laude	United Sta	Sean O'Do	0.2	1	10/11/2016	US-2016-1	33311	Eldon	Eldon Folc	2.5164	2	South	22.368	Consumer	10/18/2016	Standard Class	Florida	Storage
7	Furniture	Los Angele	United Sta	Brosina Ho	0	1	6/9/2015	CA-2015-1	90032	Eldon	Eldon Exp	14.1694	7	West	48.86	Consumer	6/14/2015	Standard Class	California	Furnishings
8	Office Sup	Los Angele	United Sta	Brosina Ho	0	1	6/9/2015	CA-2015-1	90032	Newell	Newell 32	1.9656	4	West	7.28	Consumer	6/14/2015	Standard Class	California	Art
9	Technolog	Los Angele	United Sta	Brosina Ho	0.2	1	6/9/2015	CA-2015-1	90032	Mitel	Mitel 532(	90.7152	6	West	907.152	Consumer	6/14/2015	Standard Class	California	Phones
10	Office Sup	Los Angele	United Sta	Brosina Ho	0.2	1	6/9/2015	CA-2015-1	90032	DXL	DXL Angle	5.7825	3	West	18.504	Consumer	6/14/2015	Standard Class	California	Binders
11	Office Sup	Los Angele	United Sta	Brosina Ho	0	1	6/9/2015	CA-2015-1	90032	Belkin	Belkin F5C	34.47	5	West	114.9	Consumer	6/14/2015	Standard Class	California	Appliances
12	Furniture	Los Angele	United Sta	Brosina Ho	0.2	1	6/9/2015	CA-2015-1	90032	Chromcrat	Chromcrat	85.3092	9	West	1706.184	Consumer	6/14/2015	Standard Class	California	Tables
13	Technolog	Los Angele	United Sta	Brosina Ho	0.2	1	6/9/2015	CA-2015-1	90032	Other	Konftel 25	68.3568	4	West	911.424	Consumer	6/14/2015	Standard Class	California	Phones
14	Office Sup	Concord	United Sta	Andrew Al	0.2	1	4/15/2018	CA-2018-1	28027	Xerox	Xerox 196	5.4432	3	South	15.552	Consumer	4/20/2018	Standard Class	North Car	Paper
15	Office Sup	Seattle	United Sta	Irene Mad	0.2	1	12/5/2017	CA-2017-1	98103	Fellowes	Fellowes F	132.5922	3	West	407.976	Consumer	12/10/2017	Standard Class	Washingt	Binders
16	Office Sup	Fort Wort	United Sta	Harold Pa	0.8	1	11/22/2016	US-2016-1	76106	Holmes	Holmes Re	-123.858	5	Central	68.81	Home Offi	11/26/2016	Standard Class	Texas	Appliances
17	Office Sup	Fort Wort	United Sta	Harold Pa	0.8	1	11/22/2016	US-2016-1	76106	Storex	Storex Dur	-3.816	3	Central	2.544	Home Offi	11/26/2016	Standard Class	Texas	Binders
18	Office Sup	Madison	United Sta	Pete Kriz	0	1	11/11/2015	CA-2015-1	53711	Other	Stur-D-Sto	13.3176	6	Central	665.88	Consumer	11/18/2015	Standard Class	Wisconsin	Storage
19	Office Sup	West Jord	United Sta	Alejandro	0	1	5/13/2015	CA-2015-1	84084	Fellowes	Fellowes S	9.99	2	West	55.5	Consumer	5/15/2015	Second Class	Utah	Storage
20	Office Sup	San Franci	United Sta	Zuschuss I	0	1	8/27/2015	CA-2015-1	94109	Newell	Newell 34	2.4824	2	West	8.56	Consumer	9/1/2015	Second Class	California	Art
21	Technolog	San Franci	United Sta	Zuschuss I	0.2	1	8/27/2015	CA-2015-1	94109	Cisco	Cisco SPA	16.011	3	West	213.48	Consumer	9/1/2015	Second Class	California	Phones
22	Office Sup	San Franci	United Sta	Zuschuss I	0.2	1	8/27/2015	CA-2015-1	94109	Wilson Jor	Wilson Jor	7.384	4	West	22.72	Consumer	9/1/2015	Second Class	California	Binders
23	Office Sup	Fremont	United Sta	Ken Black	0	1	12/9/2017	CA-2017-1	68025	Newell	Newell 31	5.0596	7	Central	19.46	Corporate	12/13/2017	Standard Class	Nebraska	Art
24	Office Sup	Fremont	United Sta	Ken Black	0	1	12/9/2017	CA-2017-1	68025	Acco	Acco Six-O	15.6884	7	Central	60.34	Corporate	12/13/2017	Standard Class	Nebraska	Appliances
25	Furniture	Philadelph	United Sta	Sandra Fla	0.3	1	7/16/2018	US-2018-1	19140	Global	Global De	-1.0196	2	East	71.372	Consumer	7/18/2018	Second Class	Pennsylva	Chairs
26	Furniture	Orem	United Sta	Emily Burr	0	1	9/25/2016	CA-2016-1	84057	Bretford	Bretford C	240.2649	3	West	1044.63	Consumer	9/30/2016	Standard Class	Utah	Tables
27	Office Sup	Los Angele	United Sta	Eric Hoffm	0.2	1	1/16/2017	CA-2017-1	90049	Wilson Jor	Wilson Jor	4.2224	2	West	11.648	Consumer	1/20/2017	Second Class	California	Binders
28	Technolog	Los Angele	United Sta	Eric Hoffm	0	1	1/16/2017	CA-2017-1	90049	Other	Imation 8(	11.7741	3	West	90.57	Consumer	1/20/2017	Second Class	California	Accessories
29	Furniture	Philadelph	United Sta	Tracy Blun	0.5	1	9/17/2016	US-2016-1	19140	Riverside	Riverside I	-1665.05	7	East	3083.43	Consumer	9/21/2016	Standard Class	Pennsylva	Bookcases
30	Office Sup	Philadelph	United Sta	Tracy Blun	0.7	1	9/17/2016	US-2016-1	19140	Avery	Avery Rec	-7.0532	2	East	9.618	Consumer	9/21/2016	Standard Class	Pennsylva	Binders
31	Furniture	Philadelph	United Sta	Tracy Blun	0.2	1	9/17/2016	US-2016-1	19140	Howard M	Howard M	15.525	3	East	124.2	Consumer	9/21/2016	Standard Class	Pennsylva	Furnishings

ebay

110%

ENG 9:18 PM 8/30/2019

Profit With Month



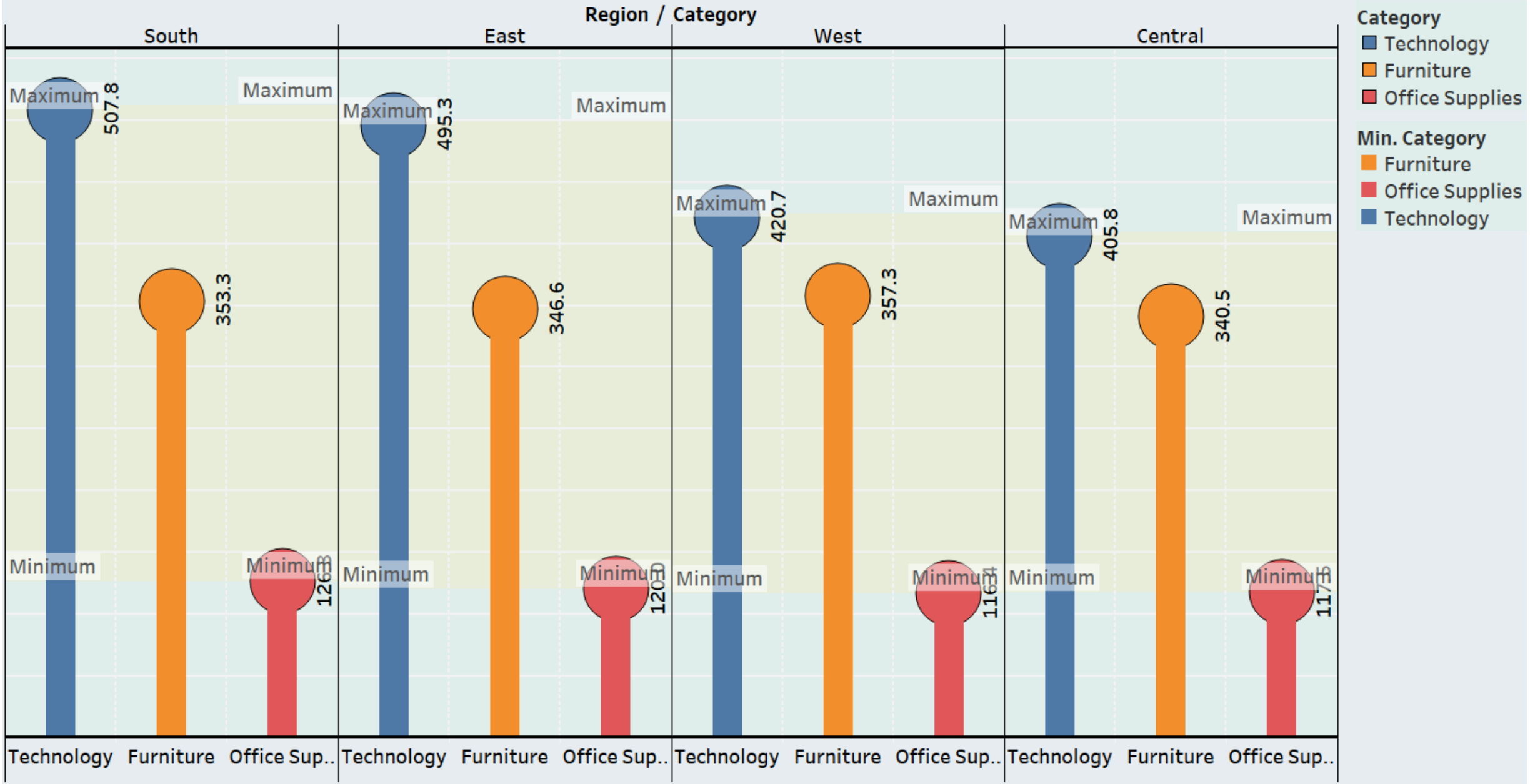
The trends of sum of Profit and sum of Profit for Order Date Month. Color shows sum of Profit. For pane Sum of Profit: The marks are labeled by average of Profit.

Month and Sales Relations
매출 과 이익 관계 월별

- 녹 색 높은 이익 골드 색은 낮은 이익((Green>Gold)

- 12 월 높은 이익 ,1 월 제일 낮음 (1,2,4,5,6,7,8 월 광고,할인 여러 행사 추천)

What do find Out in 5 Sec?? (Sales, Region,Category Field)

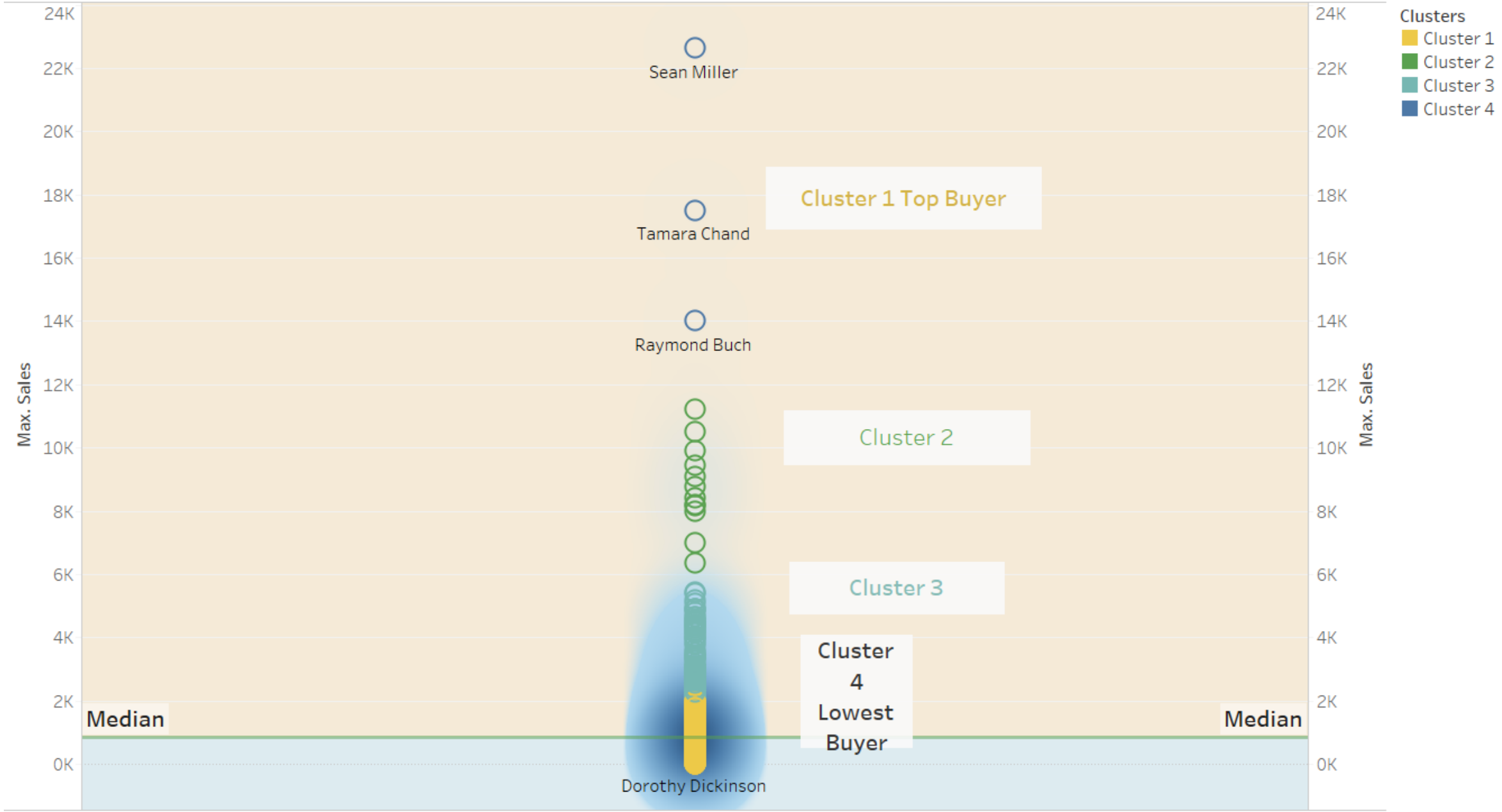


Average of Sales and average of Sales for each Category broken down by Region. For pane Average of Sales: Color shows details about Category. The marks are labeled by average of Sales. For pane Average of Sales (2): Color shows details about minimum of Category.

지역 (Region) ,매출(Sales),Category (캐토개리) 의 관계

- 높은 매출 치역 남쪽 이다.
- 많이 팔렸던 캐토고리 는 테크노로조 이다.
- 오피스 섶라이 모두 지역에서 낮고 거위 피쓰다다
- 매내저가 West 과 Central 쪽에 Technology 팔아야되는 방법 잦아야 된다

Customer Cluster



Maximum of Sales and maximum of Sales. The marks are labeled by Customer Name. Details are shown for Customer Name. For pane Maximum of Sales: Details are shown for Customer Name and Clusters. For pane Maximum of Sales (2): Color shows details about Clusters.

고객 관리 (CUSTOMER MANAGEMENT)

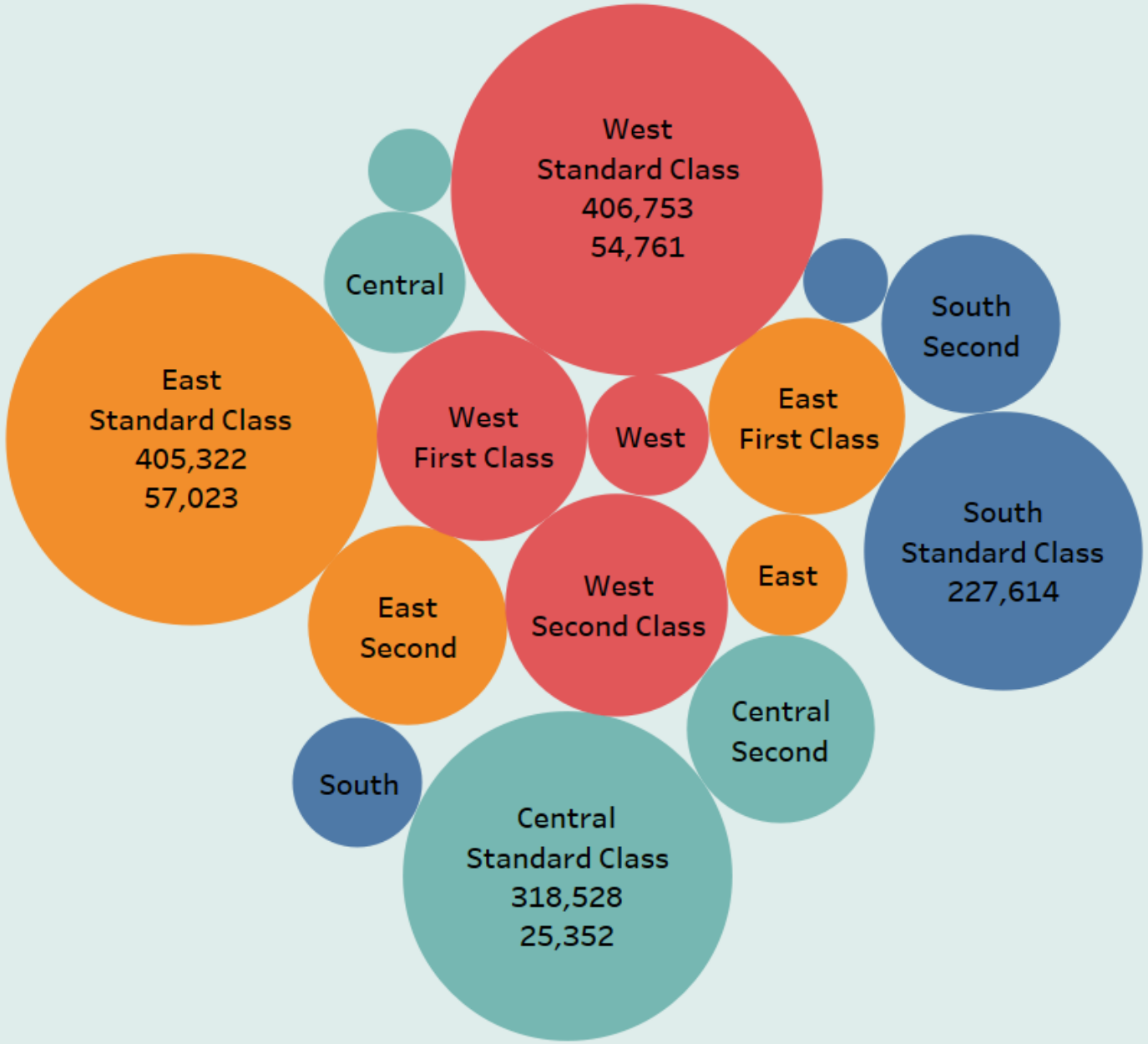
고객 4 Group 에 나눴다

Cluster 1's Customers are highly Valued Customers and Manager should take care of them

녹색 선 은 모두 고객의 Median Line 이다.

탑 구매자의 정보를 알수 있다

Mail

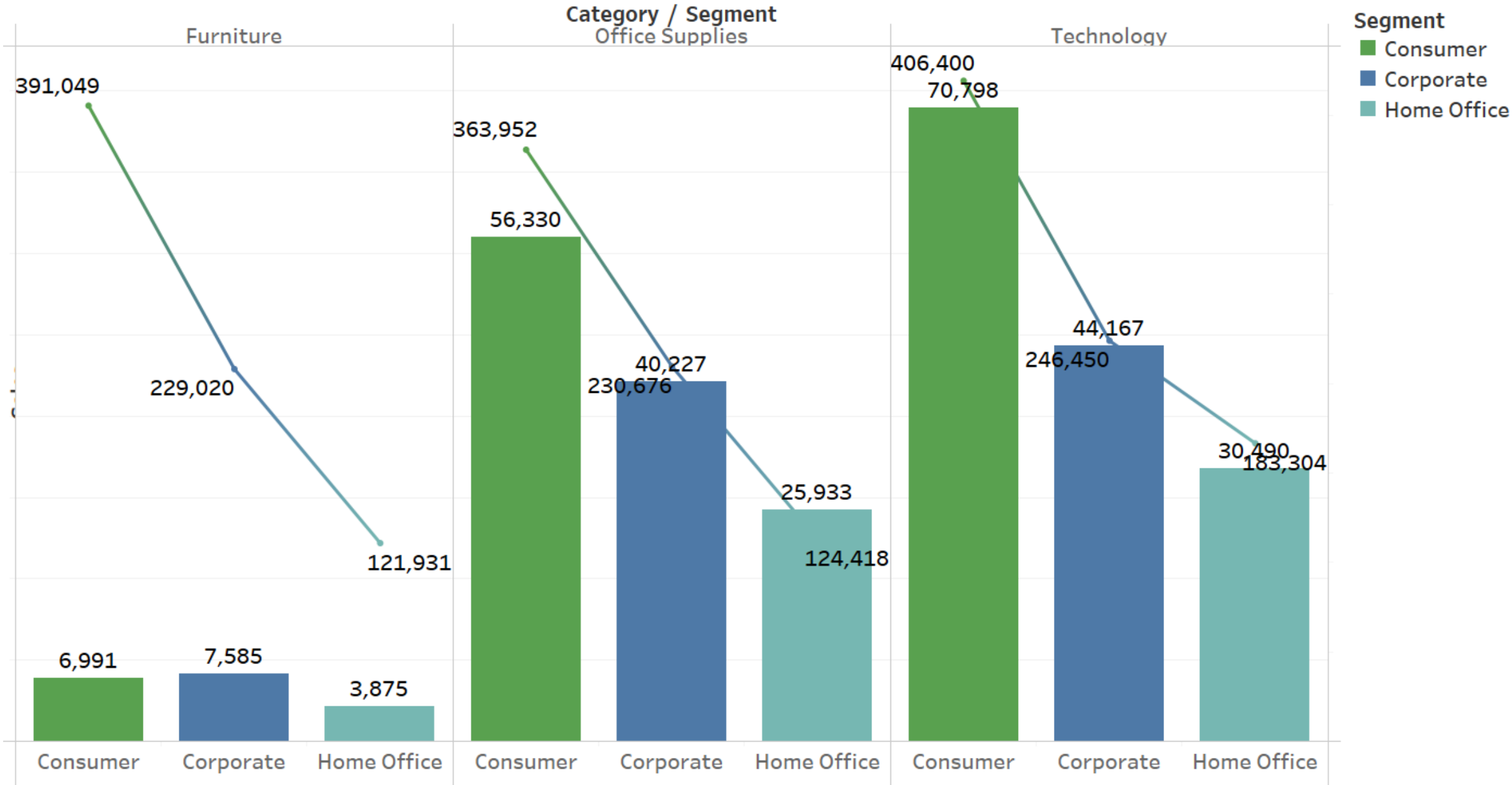


Region, Ship Mode, sum of Sales and sum of Profit. Color shows details about Region. Size shows sum of Sales. The marks are labeled by Region, Ship Mode, sum of Sales and sum of Profit.

택배 분석 (Types of Mails Analysis)

- **Standard Class Mails were sent is on Top Position**
- **모두 지역에 Standard Class 택배(배송) 방법 이용 했다**

Category,segment,profit

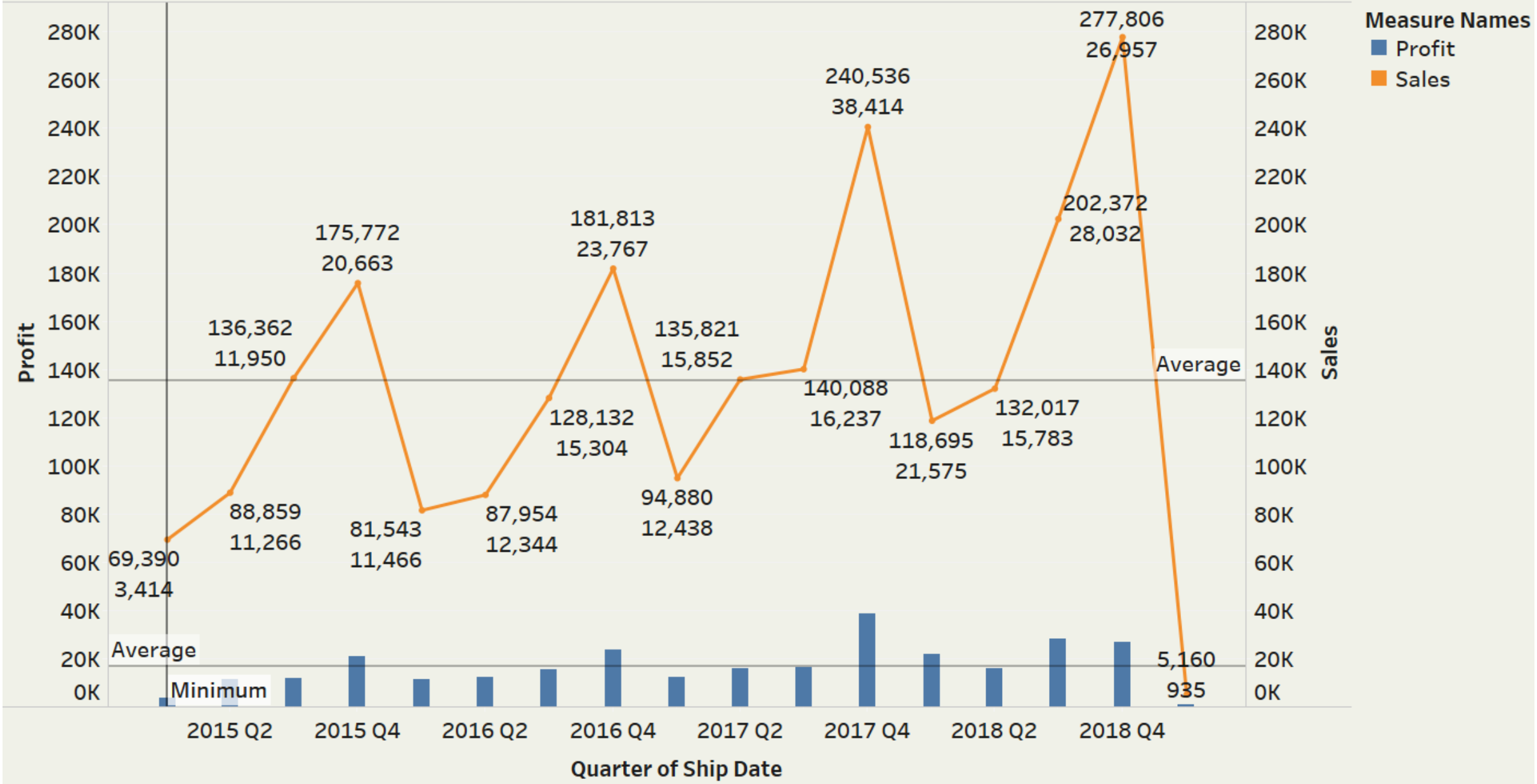


The trends of sum of Sales and sum of Profit for Segment broken down by Category. Color shows details about Segment. For pane Sum of Sales: The marks are labeled by sum of Sales. For pane Sum of Profit: The marks are labeled by sum of Profit.

Category,Segment,Sales and Profit

- 라인 은 매출 이다 그래프는 이익 이다(Line is Sale and Graph is for Profit)
- Bigger Amount is for Sale and small Amount is for Profit
- Furniture 가 Corporate(법인) 가 많이 구매 했지만 이익 너무 작다.
- Technology Items 가 마진 Ratio 가 제일 높다.

Sheet 7



The trends of Profit and Sales for Ship Date Quarter. Color shows details about Profit and Sales. For pane Sum of Sales: The marks are labeled by Sales and Profit.

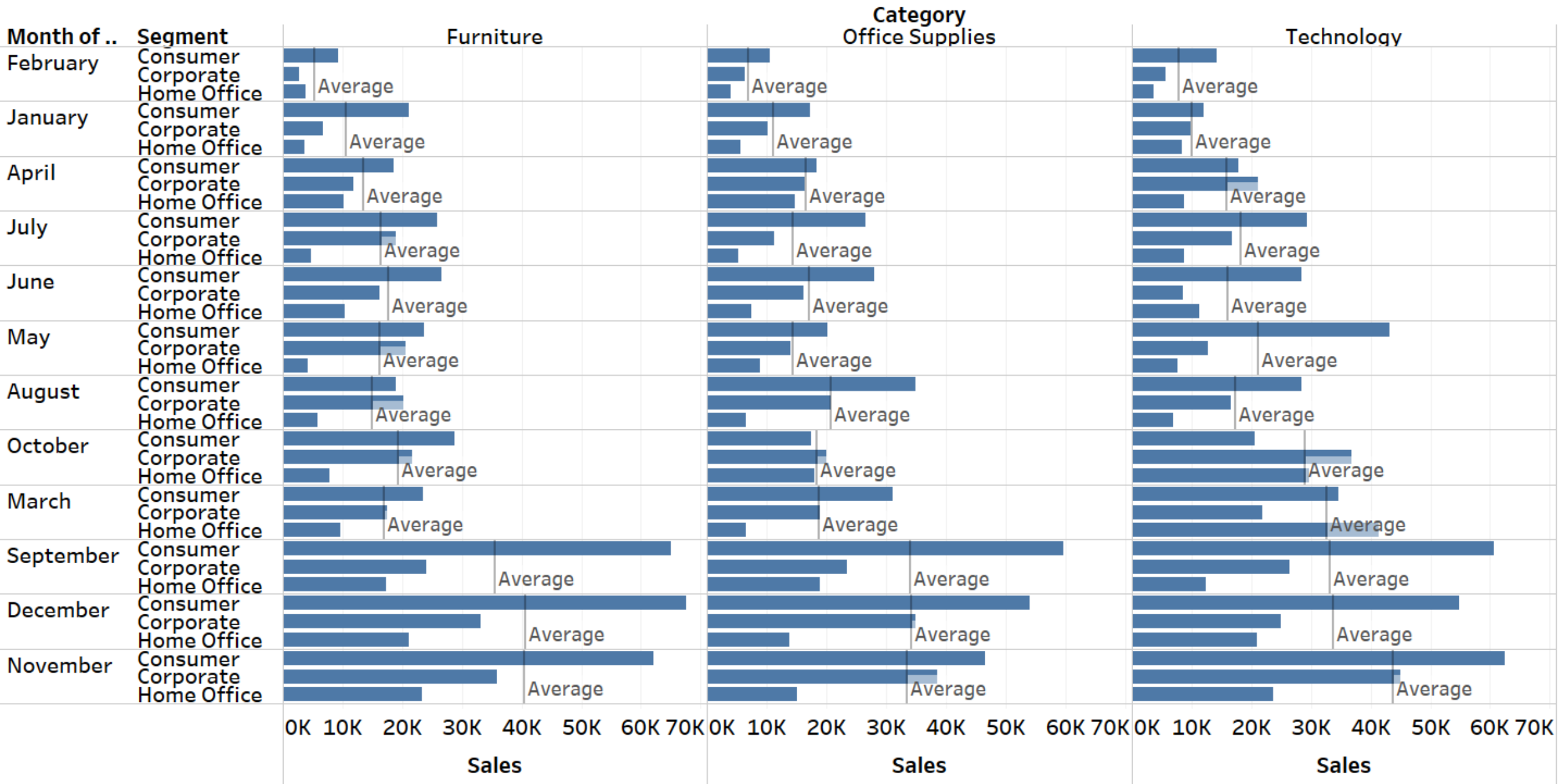
날짜별 매출 분석

Quarter 4 is Very Important

4 쿼터는 아주 중요 한다

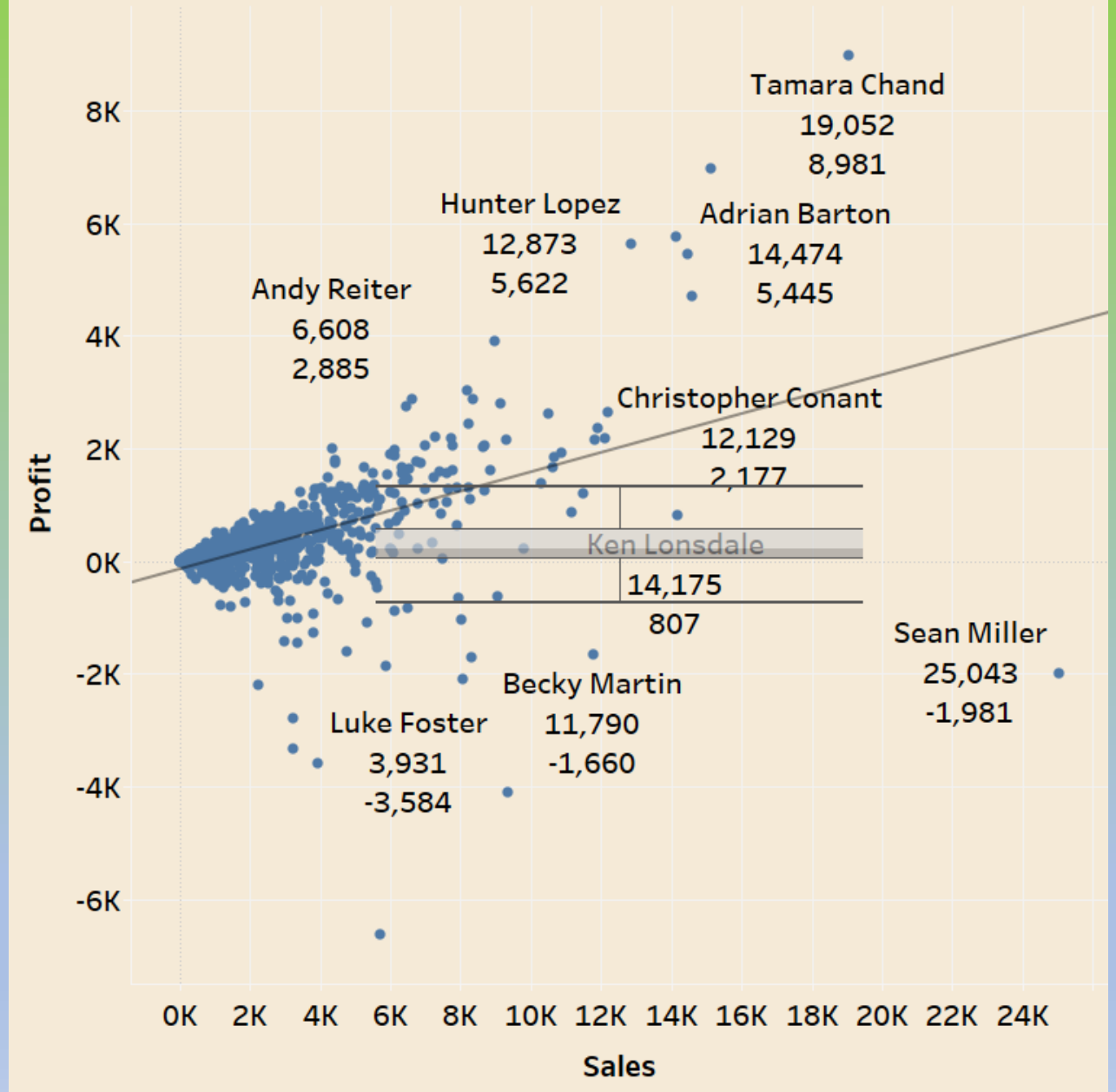
매출과 이익 아주 높다

Category_Month_



Sum of Sales for each Segment broken down by Category vs. Order Date Month.

Sales_Profit_Customer_Id



Sum of Sales vs. sum of Profit. The marks are labeled by Customer Name, sum of Sales and sum of Profit. Details are shown for Customer Name.

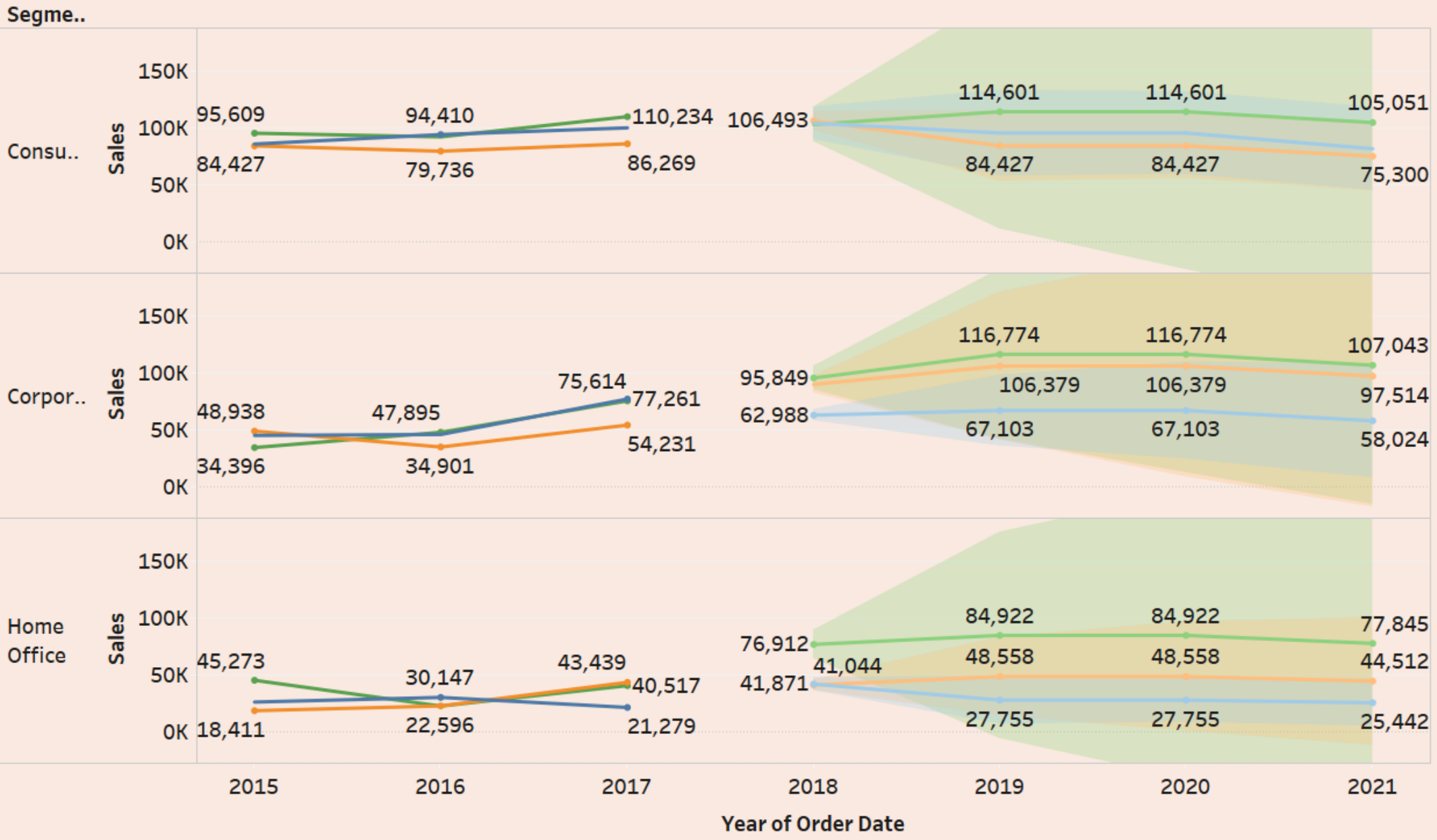
매출, 이익, 고객자 번호 의 관계

그많은 구매자 증에서 Tamara Chand 많이 중요 한 구매자 잦얼수 있다

Sean Miller 에게 물건 많이 팔아도 남는거 없다.

Luke Foster 에게서 손해가 너무 많이 생겼다.

Expected sales

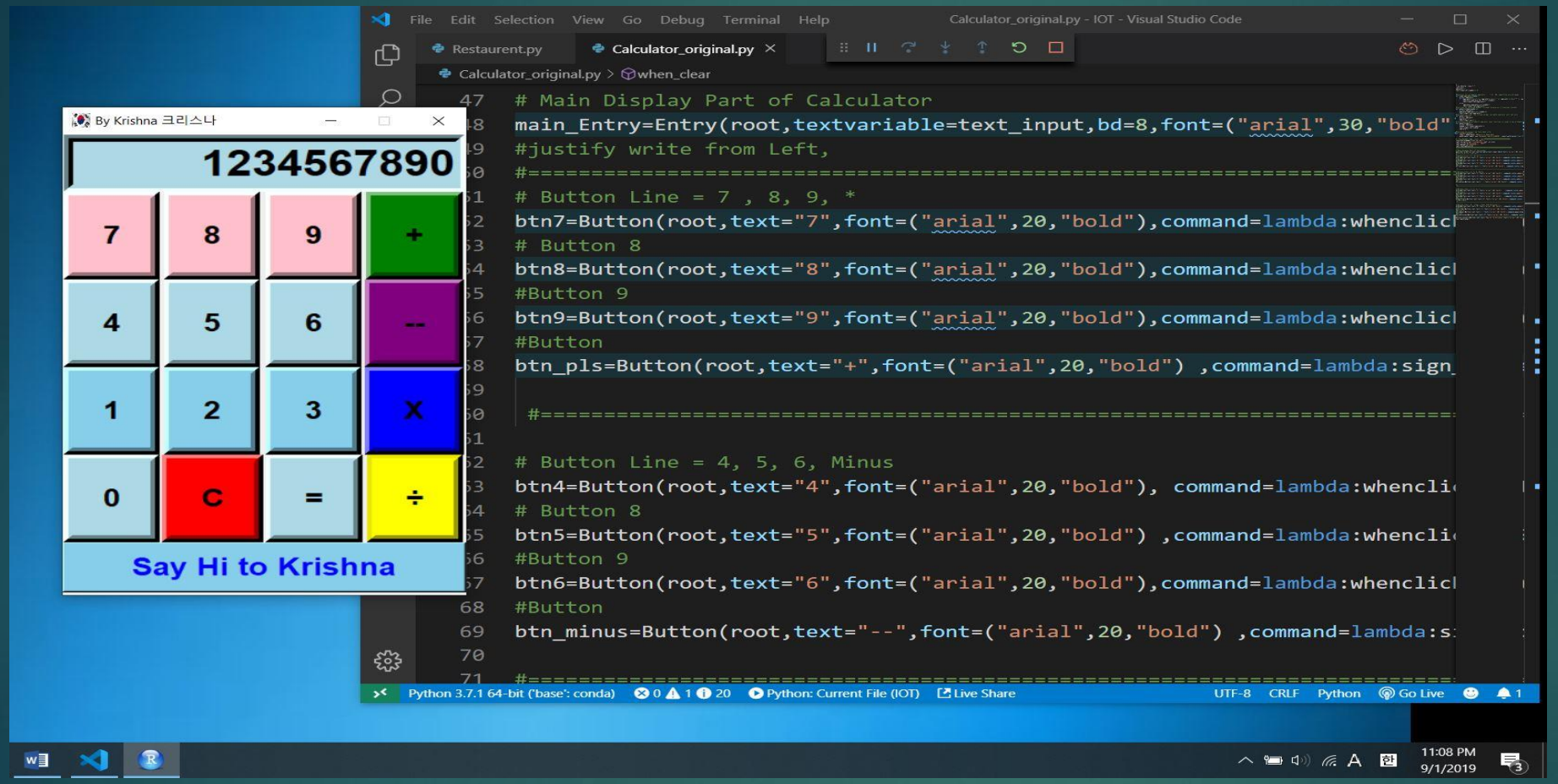


The trend of sum of Sales (actual & forecast) for Order Date Year broken down by Segment. Color shows details about Category and Forecast indicator. The marks are labeled by sum of Sales (actual & forecast) .

예상 매출 (Expected Sales)

- 태블르에서 2022 가지 예상 매출 계산 해뺐다.
- Consumer(소비자) 는 Furniture Items 구매 항률 높아지고 있다.

파이썬 TKINTER library 로 간단한 계산기



자세히 Source GitHub Repository 에 있습니다

Source Code

Color Design

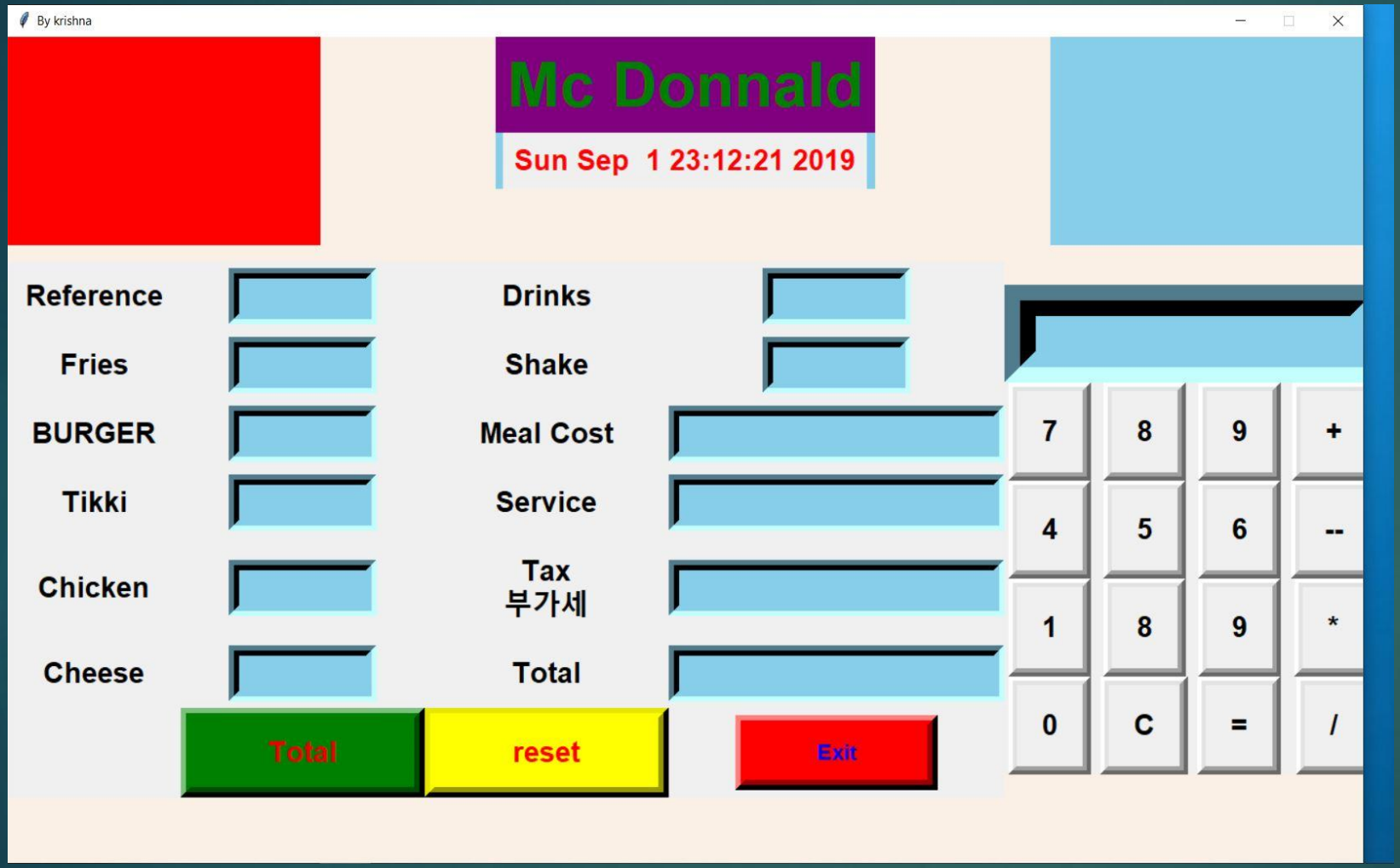
External Link

Big and Fixed Size

Developer Logo on Top

Simple Restaurant Management

Under Construction



개발 중 입니다

Python tkinter library 사용

Source: GitHub>krishdb38

https://github.com/krishdb38/Pytho