230188112_COM7036M

June 9, 2024

0.0.1 Info:This code is done in Google Colaboratory. So, if we run this code in Jupyter Notebook, some of the visualization might not be displayed correctly, might be slightly different or number might not be visible like in heatmap of correlation and confusion metrics.

Here we are going to explore and understand the supply chain dataset used by the company DataCo Global. Also we are performing descriptive, diagnostic and predictive analysis. We are going to make a prediction model to identify fake orders and suspicious orders and also forecast the sales of different products.

As I am going to do this project in Google Colaboratory (Google Colab), this is the code to mount the google drive in the colab.

[99]:
```
from google.colab import drive
drive.mount('/content/drive')/
```

[2]:
```
!pip install plotly
```

```
Requirement already satisfied: plotly in c:\users\gauri\anaconda3\lib\site-
packages (5.9.0)
Requirement already satisfied: tenacity>=6.2.0 in
c:\users\gauri\anaconda3\lib\site-packages (from plotly) (8.2.2)
```

First of all, we have to import all the different python libraries required. All the libraries required is imported in this cell which will be helpful in managing the libraries.

0.1 Importing libraries

[3]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px

from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.metrics import confusion_matrix, classification_report, auc,␣
↪accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score,␣
↪roc_auc_score, roc_curve

import warnings
warnings.filterwarnings('ignore')
%matplotlib inline
```

0.2 Importing dataset

[4]:
```
data = pd.read_csv('DataCoSupplyChainDataset.csv', encoding='ISO-8859-1')
data.head()
```

[4]:
```
       Type  Days for shipping (real)  Days for shipment (scheduled)  \
0     DEBIT                         3                              4
1  TRANSFER                         5                              4
2      CASH                         4                              4
3     DEBIT                         3                              4
4   PAYMENT                         2                              4

   Benefit per order  Sales per customer   Delivery Status  \
0          91.250000          314.640015  Advance shipping
1        -249.089996          311.359985     Late delivery
2        -247.779999          309.720001  Shipping on time
3          22.860001          304.809998  Advance shipping
4         134.210007          298.250000  Advance shipping

   Late_delivery_risk  Category Id   Category Name Customer City  …  \
0                   0           73  Sporting Goods        Caguas  …
1                   1           73  Sporting Goods        Caguas  …
2                   0           73  Sporting Goods      San Jose  …
3                   0           73  Sporting Goods   Los Angeles  …
4                   0           73  Sporting Goods        Caguas  …

  Order Zipcode Product Card Id Product Category Id  Product Description  \
0           NaN            1360                  73                  NaN
1           NaN            1360                  73                  NaN
2           NaN            1360                  73                  NaN
3           NaN            1360                  73                  NaN
4           NaN            1360                  73                  NaN

                               Product Image  Product Name Product Price  \
0  http://images.acmesports.sports/Smart+watch  Smart watch        327.75
1  http://images.acmesports.sports/Smart+watch  Smart watch        327.75
2  http://images.acmesports.sports/Smart+watch  Smart watch        327.75
3  http://images.acmesports.sports/Smart+watch  Smart watch        327.75
```

```
4  http://images.acmesports.sports/Smart+watch   Smart watch         327.75

  Product Status shipping date (DateOrders)   Shipping Mode
0              0             2/3/2018 22:56  Standard Class
1              0            1/18/2018 12:27  Standard Class
2              0            1/17/2018 12:06  Standard Class
3              0            1/16/2018 11:45  Standard Class
4              0            1/15/2018 11:24  Standard Class

[5 rows x 53 columns]
```

Let's copy the data in df variable for further use.

[5]:
```
df = data.copy()
```

1 1. Data Understanding and Preprocessing

Let's check the shape, columns, and other information of the dataset.

[6]:
```
df.shape
```

[6]:
```
(180519, 53)
```

Here are 180519 rows and 53 columns which means 180519 rows of data and 53 columns index.

[7]:
```
df.columns
```

[7]:
```
Index(['Type', 'Days for shipping (real)', 'Days for shipment (scheduled)',
       'Benefit per order', 'Sales per customer', 'Delivery Status',
       'Late_delivery_risk', 'Category Id', 'Category Name', 'Customer City',
       'Customer Country', 'Customer Email', 'Customer Fname', 'Customer Id',
       'Customer Lname', 'Customer Password', 'Customer Segment',
       'Customer State', 'Customer Street', 'Customer Zipcode',
       'Department Id', 'Department Name', 'Latitude', 'Longitude', 'Market',
       'Order City', 'Order Country', 'Order Customer Id',
       'order date (DateOrders)', 'Order Id', 'Order Item Cardprod Id',
       'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id',
       'Order Item Product Price', 'Order Item Profit Ratio',
       'Order Item Quantity', 'Sales', 'Order Item Total',
       'Order Profit Per Order', 'Order Region', 'Order State', 'Order Status',
       'Order Zipcode', 'Product Card Id', 'Product Category Id',
       'Product Description', 'Product Image', 'Product Name', 'Product Price',
       'Product Status', 'shipping date (DateOrders)', 'Shipping Mode'],
      dtype='object')
```

The columns description are as below: 1. Type: Type of transaction made 2. Days for shipping (real): Actual shipping days of the purchased product 3. Days for shipment (scheduled): Days of scheduled delivery of the purchased product 4. Benefit per order: Earnings per order placed 5. Sales per customer: Total sales per customer made per customer 6. Delivery Status: Delivery status of orders: Advance shipping , Late delivery , Shipping canceled , Shipping on tim… 7. Late_delivery_risk: Categorical variable that indicates if sending is late (1), it is not late (0). 8. Category Id: Product category code 9. Category Name: Description of the product category 10. Customer City: City where the customer made the purchase 11. Customer Country: Country where the customer made the purchase 12. Customer Email: Customer's email 13. Customer Fname: Customer name 14. Customer Id: Customer ID 15. Customer Lname: Customer lastname 16. Customer Password: Masked customer key 17. Customer Segment: Types of Customers: Consumer, Corporate , Home Office 18. Customer State: State to which the store where the purchase is registered belongs 19. Customer Street: Street to which the store where the purchase is registered belongs 20. Customer Zipcode: Customer Zipcode 21. Department Id: Department code of store 22. Department Name: Department name of store 23. Latitude: Latitude corresponding to location of store 24. Longitude: Longitude corresponding to location of store 25. Market: Market to where the order is delivered : Africa , Europe , LATAM , Pacific Asia , USCA 26. Order City: Destination city of the order 27. Order Country: Destination country of the order 29. Order Customer Id: Customer order code 30. order date (DateOrders): Date on which the order is made 31. Order Id: Order code 32. Order Item Cardprod Id: Product code generated through the RFID reader 33. Order Item Discount: Order item discount value 34. Order Item Discount Rate: Order item discount percentage 35. Order Item Id: Order item code 36. Order Item Product Price: Price of products without discount 37. Order Item Profit Ratio: Order Item Profit Ratio 38. Order Item Quantity: Number of products per order 39. Sales: Value in sales 40. Order Item Total: Total amount per order 41. Order Profit Per Order: Order Profit Per Order 42. Order Region: Region of the world where the order is delivered : Southeast Asia ,South Asia ,Oceania ,Eastern … 43. Order State: State of the region where the order is delivered 44. Order Status: Order Status : COMPLETE , PENDING , CLOSED , PENDING_PAYMENT ,CANCELED , PROCESSING ,SUSPECTED_FR… 45. Order Zipcode: Order Zipcode 45. Product Card Id: Product code 46. Product Category Id: Product category code 47. Product Description: Product Description 48. Product Image: Link of visit and purchase of the product 49. Product Name: Product Name 50. Product Price: Product Price 51. Product Status: Status of the product stock :If it is 1 not available , 0 the product is available 52. Shipping date (DateOrders): Exact date and time of shipment 53. Shipping Mode: The following shipping modes are presented : Standard Class , First Class , Second Class , Same D…

Now looking at the info and describe of the dataset.

[8]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 180519 entries, 0 to 180518
Data columns (total 53 columns):
 #   Column                         Non-Null Count   Dtype
---  ------                         --------------   -----
 0   Type                           180519 non-null  object
 1   Days for shipping (real)       180519 non-null  int64
 2   Days for shipment (scheduled)  180519 non-null  int64
 3   Benefit per order              180519 non-null  float64
 4   Sales per customer             180519 non-null  float64
 5   Delivery Status                180519 non-null  object
 6   Late_delivery_risk             180519 non-null  int64
 7   Category Id                    180519 non-null  int64
```

```
 8   Category Name                 180519 non-null  object
 9   Customer City                 180519 non-null  object
 10  Customer Country              180519 non-null  object
 11  Customer Email                180519 non-null  object
 12  Customer Fname                180519 non-null  object
 13  Customer Id                   180519 non-null  int64
 14  Customer Lname                180511 non-null  object
 15  Customer Password             180519 non-null  object
 16  Customer Segment              180519 non-null  object
 17  Customer State                180519 non-null  object
 18  Customer Street               180519 non-null  object
 19  Customer Zipcode              180516 non-null  float64
 20  Department Id                 180519 non-null  int64
 21  Department Name               180519 non-null  object
 22  Latitude                      180519 non-null  float64
 23  Longitude                     180519 non-null  float64
 24  Market                        180519 non-null  object
 25  Order City                    180519 non-null  object
 26  Order Country                 180519 non-null  object
 27  Order Customer Id             180519 non-null  int64
 28  order date (DateOrders)       180519 non-null  object
 29  Order Id                      180519 non-null  int64
 30  Order Item Cardprod Id        180519 non-null  int64
 31  Order Item Discount           180519 non-null  float64
 32  Order Item Discount Rate      180519 non-null  float64
 33  Order Item Id                 180519 non-null  int64
 34  Order Item Product Price      180519 non-null  float64
 35  Order Item Profit Ratio       180519 non-null  float64
 36  Order Item Quantity           180519 non-null  int64
 37  Sales                         180519 non-null  float64
 38  Order Item Total              180519 non-null  float64
 39  Order Profit Per Order        180519 non-null  float64
 40  Order Region                  180519 non-null  object
 41  Order State                   180519 non-null  object
 42  Order Status                  180519 non-null  object
 43  Order Zipcode                 24840 non-null   float64
 44  Product Card Id               180519 non-null  int64
 45  Product Category Id           180519 non-null  int64
 46  Product Description           0 non-null       float64
 47  Product Image                 180519 non-null  object
 48  Product Name                  180519 non-null  object
 49  Product Price                 180519 non-null  float64
 50  Product Status                180519 non-null  int64
 51  shipping date (DateOrders)    180519 non-null  object
 52  Shipping Mode                 180519 non-null  object
dtypes: float64(15), int64(14), object(24)
memory usage: 73.0+ MB
```

[9]:
```
df.describe()
```

[9]:
```
       Days for shipping (real)  Days for shipment (scheduled)  \
count             180519.000000                  180519.000000
mean                   3.497654                       2.931847
std                    1.623722                       1.374449
min                    0.000000                       0.000000
25%                    2.000000                       2.000000
50%                    3.000000                       4.000000
75%                    5.000000                       4.000000
max                    6.000000                       4.000000

       Benefit per order  Sales per customer  Late_delivery_risk  \
count      180519.000000       180519.000000       180519.000000
mean           21.974989          183.107609            0.548291
std           104.433526          120.043670            0.497664
min         -4274.979980            7.490000            0.000000
25%             7.000000          104.379997            0.000000
50%            31.520000          163.990005            1.000000
75%            64.800003          247.399994            1.000000
max           911.799988         1939.989990            1.000000

         Category Id    Customer Id  Customer Zipcode  Department Id  \
count  180519.000000  180519.000000     180516.000000  180519.000000
mean       31.851451    6691.379495      35921.126914       5.443460
std        15.640064    4162.918106      37542.461122       1.629246
min         2.000000       1.000000        603.000000       2.000000
25%        18.000000    3258.500000        725.000000       4.000000
50%        29.000000    6457.000000      19380.000000       5.000000
75%        45.000000    9779.000000      78207.000000       7.000000
max        76.000000   20757.000000      99205.000000      12.000000

            Latitude  …  Order Item Quantity          Sales  \
count  180519.000000  …        180519.000000  180519.000000
mean       29.719955  …             2.127638     203.772096
std         9.813646  …             1.453451     132.273077
min       -33.937553  …             1.000000       9.990000
25%        18.265432  …             1.000000     119.980003
50%        33.144863  …             1.000000     199.919998
75%        39.279617  …             3.000000     299.950012
max        48.781933  …             5.000000    1999.989990

       Order Item Total  Order Profit Per Order  Order Zipcode  \
count     180519.000000           180519.000000   24840.000000
mean         183.107609               21.974989   55426.132327
std          120.043670              104.433526   31919.279101
min            7.490000            -4274.979980    1040.000000
```

```
25%          104.379997                  7.000000   23464.000000
50%          163.990005                 31.520000   59405.000000
75%          247.399994                 64.800003   90008.000000
max         1939.989990                911.799988   99301.000000

       Product Card Id  Product Category Id  Product Description  \
count    180519.000000        180519.000000                  0.0
mean        692.509764            31.851451                  NaN
std         336.446807            15.640064                  NaN
min          19.000000             2.000000                  NaN
25%         403.000000            18.000000                  NaN
50%         627.000000            29.000000                  NaN
75%        1004.000000            45.000000                  NaN
max        1363.000000            76.000000                  NaN

       Product Price  Product Status
count  180519.000000        180519.0
mean      141.232550             0.0
std       139.732492             0.0
min         9.990000             0.0
25%        50.000000             0.0
50%        59.990002             0.0
75%       199.990005             0.0
max      1999.989990             0.0

[8 rows x 29 columns]
```

2 2. Data Wranging Operations

2.0.1 Let's look at the missing values

[10]:
```
df.isnull().sum()
```

[10]:
```
Type                                     0
Days for shipping (real)                 0
Days for shipment (scheduled)            0
Benefit per order                        0
Sales per customer                       0
Delivery Status                          0
Late_delivery_risk                       0
Category Id                              0
Category Name                            0
Customer City                            0
Customer Country                         0
Customer Email                           0
Customer Fname                           0
Customer Id                              0
```

```
Customer Lname                        8
Customer Password                     0
Customer Segment                      0
Customer State                        0
Customer Street                       0
Customer Zipcode                      3
Department Id                         0
Department Name                       0
Latitude                              0
Longitude                             0
Market                                0
Order City                            0
Order Country                         0
Order Customer Id                     0
order date (DateOrders)               0
Order Id                              0
Order Item Cardprod Id                0
Order Item Discount                   0
Order Item Discount Rate              0
Order Item Id                         0
Order Item Product Price              0
Order Item Profit Ratio               0
Order Item Quantity                   0
Sales                                 0
Order Item Total                      0
Order Profit Per Order                0
Order Region                          0
Order State                           0
Order Status                          0
Order Zipcode                    155679
Product Card Id                       0
Product Category Id                   0
Product Description              180519
Product Image                         0
Product Name                          0
Product Price                         0
Product Status                        0
shipping date (DateOrders)            0
Shipping Mode                         0
dtype: int64
```

Here, * Customer Lname has 8 null values, no significant since dataset has Customer Fname. * Customer Zipcode has 3 null values. * Order Zipcode has 155679 null values, not critical due to other location features so can be dropped. * Product Description has all the values null so it can be dropped.

Also, we can merge Customer Fname and Customer Lname to get Customer Name.

[11]:
```
df['Customer Name'] = df['Customer Fname'].astype(str) + ' ' + df['Customer␣
↪Lname'].astype(str)
```

Let's explore some Ids columns like Category Id, Customer Id, Department Id, Order Customer Id, Order Id, Order Cardprod Id, Order Item Id, Product Card Id, Product Category Id, etc.

[12]:
```
print((df['Category Id'] == df['Product Category Id']).value_counts())
print((df['Customer Id'] == df['Order Customer Id']).value_counts())
print((df['Order Id'] == df['Order Item Cardprod Id']).value_counts())
print((df['Order Id'] == df['Order Item Id']).value_counts())
print((df['Benefit per order'] == df['Order Profit Per Order']).value_counts())
print((df['Order Item Product Price'] == df['Product Price']).value_counts())
print((df['Sales per customer'] == df['Order Item Total']).value_counts())
print(df['Product Status'].value_counts())
```

```
True    180519
Name: count, dtype: int64
True    180519
Name: count, dtype: int64
False    180519
Name: count, dtype: int64
False    180517
True          2
Name: count, dtype: int64
True    180519
Name: count, dtype: int64
True    180519
Name: count, dtype: int64
True    180519
Name: count, dtype: int64
Product Status
0    180519
Name: count, dtype: int64
```

Here, Category Id and Product Category Id are same, Customer Id and Order Customer Id are also same so one of the columns can be dropped. But Ids columns don't have any significant impact on analysis and prediction so can be dropped.

Now, we dropped the unwanted columns like: * **Product Description** which has all the values null. * **Order Zipcode** no any significant impact. * **Category ID, Customer ID, Department ID, Order Customer ID, Order Id, Order Item ID, Order Item Cardprod Id, Product Card ID, Product Category ID** has no any information * **Customer Fname, Customer Lname** which are merged as Customer Name * **Customer Email, Customer Password, Customer Street, Customer Zipcode, Product Image** can be dropped. * Here the product status is all 0 which means the product are always available. * **Benefit per order** and **Order Benefir per Order** * **Order Item Product Price** and **Product Price** * **Sales Per Customer** and **Order Item Total** are same values so we can remove one column.

[13]:
```
columns_to_be_dropped = ['Days for shipment (scheduled)', 'Category Id',␣
↪'Customer Id', 'Customer Email', 'Customer Fname', 'Customer Lname',
                         'Customer Street', 'Customer Password', 'Department␣
↪Id', 'Order Customer Id', 'Order Item Id', 'Order Item Cardprod Id',
                         'Product Card Id', 'Product Category Id', 'Customer␣
↪Zipcode', 'Customer State', 'Order Zipcode', 'Product Description',
                         'Product Image', 'shipping date (DateOrders)',␣
↪'Product Status', 'Benefit per order', 'Order Item Product Price',
                         'Order Item Total']

df.drop(columns_to_be_dropped, axis = 1, inplace = True)
```

Again check for missing values

[14]:
```
df.isnull().sum()
```

[14]:
```
Type                        0
Days for shipping (real)    0
Sales per customer          0
Delivery Status             0
Late_delivery_risk          0
Category Name               0
Customer City               0
Customer Country            0
Customer Segment            0
Department Name             0
Latitude                    0
Longitude                   0
Market                      0
Order City                  0
Order Country               0
order date (DateOrders)     0
Order Id                    0
Order Item Discount         0
Order Item Discount Rate    0
Order Item Profit Ratio     0
Order Item Quantity         0
Sales                       0
Order Profit Per Order      0
Order Region                0
Order State                 0
Order Status                0
Product Name                0
Product Price               0
Shipping Mode               0
Customer Name               0
dtype: int64
```

Now, there is no any null values in the dataset.

Renaming the column Order Profit Per Order to Profit Per Order.

[15]:
```
df.rename({'Order Profit Per Order': 'Profit Per Order'}, axis=1, inplace=True)
```

Let's look which columns are categorical features and which columns are numerical features

[16]:
```
numerical_features = [f for f in df.columns if df[f].dtypes!='O']
cat_features = [c for c in df.columns if df[c].dtypes=='O']
print("Numerical Features: ", numerical_features)
print(" ")
print("Categorical Features: ", cat_features)
```

```
Numerical Features:  ['Days for shipping (real)', 'Sales per customer',
'Late_delivery_risk', 'Latitude', 'Longitude', 'Order Id', 'Order Item
Discount', 'Order Item Discount Rate', 'Order Item Profit Ratio', 'Order Item
Quantity', 'Sales', 'Profit Per Order', 'Product Price']

Categorical Features:  ['Type', 'Delivery Status', 'Category Name', 'Customer
City', 'Customer Country', 'Customer Segment', 'Department Name', 'Market',
'Order City', 'Order Country', 'order date (DateOrders)', 'Order Region', 'Order
State', 'Order Status', 'Product Name', 'Shipping Mode', 'Customer Name']
```

Looking at the unique values at every categorial features.

[17]:
```
for column in cat_features:
    unique_values = df[column].unique()
    print(f"Column: {column}")
    print(f"Number of Unique Values: {len(unique_values)}")
    print(f"Unique Values: {unique_values}\n")
```

```
Column: Type
Number of Unique Values: 4
Unique Values: ['DEBIT' 'TRANSFER' 'CASH' 'PAYMENT']

Column: Delivery Status
Number of Unique Values: 4
Unique Values: ['Advance shipping' 'Late delivery' 'Shipping on time' 'Shipping
canceled']

Column: Category Name
Number of Unique Values: 50
Unique Values: ['Sporting Goods' 'Cleats' 'Shop By Sport' "Women's Apparel"
'Electronics'
 'Boxing & MMA' 'Cardio Equipment' 'Trade-In' "Kids' Golf Clubs"
 'Hunting & Shooting' 'Baseball & Softball' "Men's Footwear"
 'Camping & Hiking' 'Consumer Electronics' 'Cameras ' 'Computers'
 'Basketball' 'Soccer' "Girls' Apparel" 'Accessories' "Women's Clothing"
 'Crafts' "Men's Clothing" 'Tennis & Racquet' 'Fitness Accessories'
```

```
 'As Seen on  TV!' 'Golf Balls' 'Strength Training' "Children's Clothing"
 'Lacrosse' 'Baby ' 'Fishing' 'Books ' 'DVDs' 'CDs ' 'Garden' 'Hockey'
 'Pet Supplies' 'Health and Beauty' 'Music' 'Video Games' 'Golf Gloves'
 'Golf Bags & Carts' 'Golf Shoes' 'Golf Apparel' "Women's Golf Clubs"
 "Men's Golf Clubs" 'Toys' 'Water Sports' 'Indoor/Outdoor Games']

Column: Customer City
Number of Unique Values: 563
Unique Values: ['Caguas' 'San Jose' 'Los Angeles' 'Tonawanda' 'Miami' 'San
Ramon'
 'Freeport' 'Salinas' 'Peabody' 'Canovanas' 'Paramount' 'Mount Prospect'
 'Long Beach' 'Rancho Cordova' 'Billings' 'Wilkes Barre' 'Roseville'
 'Bellflower' 'Wheaton' 'Detroit' 'Dallas' 'Carlisle' 'Newark'
 'Panorama City' 'Atlanta' 'Fremont' 'Rochester' 'Bayamon' 'Guayama'
 'Juana Diaz' 'Fort Washington' 'Bakersfield' 'Corona' 'Cincinnati'
 'Germantown' 'Carrollton' 'Houston' 'Ewa Beach' 'Lakewood' 'Rome' 'Vista'
 'Fort Worth' 'Fond Du Lac' 'Philadelphia' 'Ontario' 'Oviedo' 'Buffalo'
 'Honolulu' 'Oceanside' 'North Tonawanda' 'Clovis' 'Jamaica'
 'Granite City' 'Medford' 'Pomona' 'Tempe' 'Santa Ana' 'York' 'Aurora'
 'Simi Valley' 'Silver Spring' 'Saint Paul' 'San Antonio' 'Bronx'
 'Greenville' 'Morristown' 'San Diego' 'Oxnard' 'Albuquerque' 'Amarillo'
 'Lutz' 'Bend' 'East Brunswick' 'Lancaster' 'Hampton' 'New York'
 'Porterville' 'Portland' 'Strongsville' 'El Paso' 'Del Rio' 'Bountiful'
 'Kent' 'Chicago' 'Plymouth' 'Far Rockaway' 'Garden Grove' 'Placentia'
 'Mentor' 'Santa Clara' 'Union' 'Westminster' 'Pompano Beach' 'Azusa'
 'Fort Lauderdale' 'Princeton' 'Perth Amboy' 'Loveland' 'Virginia Beach'
 'Louisville' 'Lockport' 'Staten Island' 'Tucson' 'Cleveland' 'Webster'
 'Stockton' 'Martinsburg' 'Cumberland' 'Pekin' 'Tallahassee'
 'Jacksonville' 'Woonsocket' 'Lithonia' 'Oak Lawn' 'Alhambra' 'New Haven'
 'Phoenix' 'Kenner' 'Washington' 'Holland' 'Morrisville' 'Memphis'
 'Federal Way' 'West Covina' 'Ventura' 'Valrico' 'Kaneohe' 'Brooklyn'
 'Lodi' 'Murfreesboro' 'Carlsbad' 'Hamilton' 'Hayward' 'Bridgeton'
 'Bay Shore' 'Palatine' 'Smyrna' 'Van Nuys' 'Opa Locka' 'Edison' 'Baytown'
 'Sylmar' 'Burnsville' 'Huntington Station' 'Sunnyvale' 'Sugar Land'
 'Brighton' 'Bismarck' 'Gaithersburg' 'Lilburn' 'Provo' 'Columbia'
 'Marietta' 'Rio Grande' 'Denver' 'Taylor' 'Saint Charles' 'Cupertino'
 'Springfield' 'Mission Viejo' 'Roswell' 'Ypsilanti' 'Peoria' 'Clementon'
 'Antioch' 'Salt Lake City' 'Granada Hills' 'Hempstead' 'Astoria' 'Gilroy'
 'Lenoir' 'Columbus' 'Albany' 'Humacao' 'Lindenhurst' 'Elyria' 'Riverside'
 'Carson' 'Mesa' 'San Juan' 'Vega Baja' 'Mayaguez' 'Arecibo'
 'San Sebastian' 'Eugene' 'Algonquin' 'Indianapolis' 'Buena Park'
 'Catonsville' 'Jersey City' 'Lombard' 'New Bedford' 'Newburgh' 'Lansdale'
 'Baltimore' 'Fullerton' 'Sacramento' 'Greensboro' 'Roseburg' 'Modesto'
 'Encinitas' 'Watsonville' 'Meridian' 'Endicott' 'Katy' 'Visalia' 'Lompoc'
 'Ogden' 'Raleigh' 'Hacienda Heights' 'Union City' 'Hollywood'
 'Bolingbrook' 'West Lafayette' 'Woodbridge' 'Weslaco' 'Bell Gardens'
 'La Mirada' 'North Bergen' 'Madison' 'South San Francisco'
 'North Las Vegas' 'Methuen' 'Costa Mesa' 'Glen Burnie' 'Fairfield'
```

```
'Winnetka' 'Mcallen' 'Joliet' 'Brownsville' 'Pawtucket'
'Colorado Springs' 'Quincy' 'Pittsfield' 'Chino' 'Marion' 'North Hills'
'Salina' 'Hyattsville' 'North Richland Hills' 'Spring Valley' 'Lawrence'
'Milpitas' 'Rowland Heights' 'Gardena' 'Cicero' 'Asheboro' 'La Crosse'
'Florissant' 'Canyon Country' 'Ithaca' 'Allentown' 'Escondido' 'Martinez'
'Troy' 'Arlington' 'Davis' 'Chandler' 'Elgin' 'Palmdale' 'Massapequa'
'Pittsburg' 'West New York' 'Orlando' 'Hanover' 'Glendale' 'Enfield'
'Baldwin Park' 'Chino Hills' 'Toms River' 'Wyandotte' 'Mililani' 'Harvey'
'Mechanicsburg' 'Opelousas' 'Kailua' 'Norfolk' 'Elmhurst' 'Chillicothe'
'Canoga Park' 'Jackson' 'Moreno Valley' 'New Orleans' 'San Benito'
'New Castle' 'Bloomfield' 'Cypress' 'Marrero' 'Grand Prairie' 'Greeley'
'Littleton' 'Longmont' 'Chesapeake' 'Englewood' 'Arlington Heights'
'Tampa' 'Irvington' 'Forest Hills' 'Dearborn' 'Compton' 'Garland'
'Waipahu' 'Carmichael' 'Tustin' 'Anaheim' 'Canton' 'Stafford'
'South Richmond Hill' 'Middletown' 'West Orange' 'Daly City'
'Powder Springs' 'Parkville' 'Hialeah' 'Beloit' 'Aguadilla' 'Carolina'
'Yauco' 'Saint Peters' 'Augusta' 'Chapel Hill' 'East Lansing' 'Stamford'
'Diamond Bar' 'Milwaukee' 'Lawrenceville' 'Manchester' 'La Puente'
'Victorville' 'Richmond' 'Eagle Pass' 'Fontana' 'Ballwin' 'New Braunfels'
'Las Vegas' 'Goose Creek' 'Pharr' 'Yonkers' 'El Monte' 'Reynoldsburg'
'Hamtramck' 'Medina' 'Highland' 'Jonesboro' 'Elk Grove' 'Montebello'
'San Francisco' 'Glenview' 'Rock Hill' 'Austin' 'Scottsdale' 'Santa Cruz'
'Oregon City' 'Annandale' 'Plano' 'Piscataway' 'El Cajon' 'Hilliard'
'Orange Park' 'Decatur' 'San Pablo' 'Douglasville' 'Henderson'
'College Station' 'Round Rock' 'Mesquite' 'Broken Arrow' 'Redmond'
'Findlay' 'La Habra' 'Laguna Hills' 'San Bernardino' 'Apex'
'South El Monte' 'Irving' 'Blacksburg' 'Dorchester Center' 'Potomac'
'Winter Park' 'Stone Mountain' 'Goleta' 'Hagerstown' 'Alameda'
'Saint Louis' 'Pico Rivera' 'Chula Vista' 'Hollister' 'North Hollywood'
'New Brunswick' 'Beaverton' 'Chicago Heights' 'Hesperia' 'Cary' 'Sanford'
'Laredo' 'Westland' 'Stockbridge' 'Carol Stream' 'Wichita' 'Olathe'
'Flushing' 'Lynwood' 'Revere' 'Westerville' 'Cordova' 'Hanford' 'Rialto'
'Mchenry' 'Mission' 'Salem' 'Duluth' 'Danbury' 'Frankfort' 'Upland'
'Rosemead' 'Mount Pleasant' 'Lake Forest' 'West Chester' 'Woodside'
'Norcross' 'Fresno' 'Zanesville' 'Painesville' 'Lynnwood' 'Massillon'
'Crystal Lake' 'Rego Park' 'Ann Arbor' 'Wyoming' 'La Mesa' 'Edinburg'
'Howell' 'Michigan City' 'Sheboygan' 'Moline' 'Yuma' 'Campbell'
'Charlotte' 'Oakland' 'San Marcos' 'Walnut' 'Harlingen' 'Rio Rancho'
'Nashville' 'Annapolis' 'Laguna Niguel' 'Santee' 'West Jordan' 'Hickory'
'Manati' 'Trujillo Alto' 'Ponce' 'Toa Alta' 'Irwin' 'South Ozone Park'
'Ridgewood' 'Bowling Green' 'Richardson' 'Sun Valley' 'Huntington Beach'
'Fargo' 'Waukegan' 'Highland Park' 'Cerritos' 'Lewisville' 'Alpharetta'
'New Albany' 'Denton' 'Temecula' 'Tinley Park' 'Dundalk' 'Crown Point'
'Lawton' 'Fayetteville' 'Milford' 'Bartlett' 'Reno' 'Passaic' 'Reseda'
'Levittown' 'Wayne' 'Metairie' 'Wheeling' 'Hawthorne' 'Napa' 'Berwyn'
'Fountain Valley' 'Las Cruces' 'Apopka' 'Folsom' 'El Centro'
'Jackson Heights' 'Pacoima' 'Hendersonville' 'Clearfield' 'Seattle'
'Saginaw' 'Conway' 'Sandusky' 'San Pedro' 'Grove City' 'Knoxville'
```

```
 'Huntington Park' 'Greensburg' 'Poway' 'O Fallon' 'Chambersburg' 'Normal'
 'Lynn' 'Bensalem' 'Bristol' 'Williamsport' 'Longview' 'Norwalk' 'Bayonne'
 'Tulare' 'National City' 'Dayton' 'Tracy' 'Summerville' 'Merced'
 'Brockton' 'Vallejo' 'West Haven' 'Pasadena' 'South Gate' 'Warren'
 'Clarksville' 'Muskegon' 'Brandon' 'Rancho Cucamonga' 'Santa Maria'
 'Doylestown' 'Colton' 'Indio' 'Plainfield' 'Bellingham' 'Spring'
 'Livermore' 'Santa Fe' 'Palo Alto' 'Henrico' 'Des Plaines' 'Birmingham'
 'Broomfield' 'Guaynabo' 'Cayey' 'Citrus Heights' 'Spokane' 'Dubuque'
 'Madera' 'Everett' 'Brentwood' 'Morganton' 'Vacaville' 'Malden'
 'Gwynn Oak' 'Toa Baja' 'Taunton' 'Freehold' 'Sumner' 'Wilmington' 'CA']

Column: Customer Country
Number of Unique Values: 2
Unique Values: ['Puerto Rico' 'EE. UU.']

Column: Customer Segment
Number of Unique Values: 3
Unique Values: ['Consumer' 'Home Office' 'Corporate']

Column: Department Name
Number of Unique Values: 11
Unique Values: ['Fitness' 'Apparel' 'Golf' 'Footwear' 'Outdoors' 'Fan Shop'
'Technology'
 'Book Shop' 'Discs Shop' 'Pet Shop' 'Health and Beauty ']

Column: Market
Number of Unique Values: 5
Unique Values: ['Pacific Asia' 'USCA' 'Africa' 'Europe' 'LATAM']

Column: Order City
Number of Unique Values: 3597
Unique Values: ['Bekasi' 'Bikaner' 'Townsville' … 'Tongling' 'Liuyang'
'Nashua']

Column: Order Country
Number of Unique Values: 164
Unique Values: ['Indonesia' 'India' 'Australia' 'China' 'Japón' 'Corea del Sur'
 'Singapur' 'Turquía' 'Mongolia' 'Estados Unidos' 'Nigeria'
 'República Democrática del Congo' 'Senegal' 'Marruecos' 'Alemania'
 'Francia' 'Países Bajos' 'Reino Unido' 'Guatemala' 'El Salvador' 'Panamá'
 'República Dominicana' 'Venezuela' 'Colombia' 'Honduras' 'Brasil'
 'México' 'Uruguay' 'Argentina' 'Cuba' 'Perú' 'Nicaragua' 'Ecuador'
 'Angola' 'Sudán' 'Somalia' 'Costa de Marfil' 'Egipto' 'Italia' 'España'
 'Suecia' 'Austria' 'Canada' 'Madagascar' 'Argelia' 'Liberia' 'Zambia'
 'Níger' 'SudAfrica' 'Mozambique' 'Tanzania' 'Ruanda' 'Israel'
 'Nueva Zelanda' 'Bangladés' 'Tailandia' 'Irak' 'Arabia Saudí' 'Filipinas'
 'Kazajistán' 'Irán' 'Myanmar (Birmania)' 'Uzbekistán' 'Benín' 'Camerún'
 'Kenia' 'Togo' 'Ucrania' 'Polonia' 'Portugal' 'Rumania'
```

```
 'Trinidad y Tobago' 'Afganistán' 'Pakistán' 'Vietnam' 'Malasia'
 'Finlandia' 'Rusia' 'Irlanda' 'Noruega' 'Eslovaquia' 'Bélgica' 'Bolivia'
 'Chile' 'Jamaica' 'Yemen' 'Ghana' 'Guinea' 'Etiopía' 'Bulgaria'
 'Kirguistán' 'Georgia' 'Nepal' 'Emiratos Árabes Unidos' 'Camboya'
 'Uganda' 'Lesoto' 'Lituania' 'Suiza' 'Hungría' 'Dinamarca' 'Haití'
 'Bielorrusia' 'Croacia' 'Laos' 'Baréin' 'Macedonia' 'República Checa'
 'Sri Lanka' 'Zimbabue' 'Eritrea' 'Burkina Faso' 'Costa Rica' 'Libia'
 'Barbados' 'Tayikistán' 'Siria' 'Guadalupe' 'Papúa Nueva Guinea'
 'Azerbaiyán' 'Turkmenistán' 'Paraguay' 'Jordania' 'Hong Kong' 'Martinica'
 'Moldavia' 'Qatar' 'Mali' 'Albania' 'República del Congo'
 'Bosnia y Herzegovina' 'Omán' 'Túnez' 'Sierra Leona' 'Yibuti' 'Burundi'
 'Montenegro' 'Gabón' 'Sudán del Sur' 'Luxemburgo' 'Namibia' 'Mauritania'
 'Grecia' 'Suazilandia' 'Guyana' 'Guayana Francesa'
 'República Centroafricana' 'Taiwán' 'Estonia' 'Líbano' 'Chipre'
 'Guinea-Bissau' 'Surinam' 'Belice' 'Eslovenia' 'República de Gambia'
 'Botsuana' 'Armenia' 'Guinea Ecuatorial' 'Kuwait' 'Bután' 'Chad' 'Serbia'
 'Sáhara Occidental']

Column: order date (DateOrders)
Number of Unique Values: 65752
Unique Values: ['1/31/2018 22:56' '1/13/2018 12:27' '1/13/2018 12:06' …
 '1/21/2016 2:47' '1/20/2016 7:10' '1/17/2016 5:56']

Column: Order Region
Number of Unique Values: 23
Unique Values: ['Southeast Asia' 'South Asia' 'Oceania' 'Eastern Asia' 'West
Asia'
 'West of USA ' 'US Center ' 'West Africa' 'Central Africa' 'North Africa'
 'Western Europe' 'Northern Europe' 'Central America' 'Caribbean'
 'South America' 'East Africa' 'Southern Europe' 'East of USA' 'Canada'
 'Southern Africa' 'Central Asia' 'Eastern Europe' 'South of  USA ']

Column: Order State
Number of Unique Values: 1089
Unique Values: ['Java Occidental' 'Rajastán' 'Queensland' … 'Bistrita-Nasaud'
'Tottori'
 'Khorezm']

Column: Order Status
Number of Unique Values: 9
Unique Values: ['COMPLETE' 'PENDING' 'CLOSED' 'PENDING_PAYMENT' 'CANCELED'
'PROCESSING'
 'SUSPECTED_FRAUD' 'ON_HOLD' 'PAYMENT_REVIEW']

Column: Product Name
Number of Unique Values: 118
Unique Values: ['Smart watch ' 'Perfect Fitness Perfect Rip Deck'
 "Under Armour Girls' Toddler Spine Surge Runni"
```

```
"Nike Men's Dri-FIT Victory Golf Polo"
"Under Armour Men's Compression EV SL Slide"
"Under Armour Women's Micro G Skulpt Running S"
"Nike Men's Free 5.0+ Running Shoe"
"Glove It Women's Mod Oval 3-Zip Carry All Gol"
'Bridgestone e6 Straight Distance NFL San Dieg'
"Columbia Men's PFG Anchor Tough T-Shirt" 'Titleist Pro V1x Golf Balls'
'Bridgestone e6 Straight Distance NFL Tennesse'
'Polar FT4 Heart Rate Monitor' 'ENO Atlas Hammock Straps'
"adidas Men's F10 Messi TRX FG Soccer Cleat"
"Brooks Women's Ghost 6 Running Shoe"
"Nike Men's CJ Elite 2 TD Football Cleat"
"Diamondback Women's Serene Classic Comfort Bi"
'Industrial consumer electronics' 'Web Camera' 'Dell Laptop'
'SOLE E25 Elliptical' 'Elevation Training Mask 2.0'
"adidas Men's Germany Black Crest Away Tee"
'Team Golf Pittsburgh Steelers Putter Grip'
'Glove It Urban Brick Golf Towel' 'Team Golf Texas Longhorns Putter Grip'
"Nike Men's Deutschland Weltmeister Winners Bl"
'Team Golf St. Louis Cardinals Putter Grip' 'Summer dresses'
'Porcelain crafts' "Men's gala suit"
'Team Golf Tennessee Volunteers Putter Grip'
'Team Golf San Francisco Giants Putter Grip'
'Glove It Imperial Golf Towel' "Nike Men's Comfort 2 Slide"
'Under Armour Hustle Storm Medium Duffle Bag'
"Under Armour Kids' Mercenary Slide"
"Under Armour Women's Ignite PIP VI Slide"
"Nike Men's Free TR 5.0 TB Training Shoe"
'adidas Youth Germany Black/Red Away Match Soc'
"TYR Boys' Team Digi Jammer" "Glove It Women's Imperial Golf Glove"
'Titleist Pro V1x High Numbers Golf Balls'
'Bridgestone e6 Straight Distance NFL Carolina'
"Under Armour Women's Ignite Slide"
'Titleist Pro V1x High Numbers Personalized Go'
'GoPro HERO3+ Black Edition Camera' 'Total Gym 1400' "Children's heaters"
'Team Golf New England Patriots Putter Grip'
"adidas Kids' F5 Messi FG Soccer Cleat" "Nike Women's Tempo Shorts"
"Glove It Women's Mod Oval Golf Glove"
'Titleist Pro V1 High Numbers Personalized Gol'
"Under Armour Men's Tech II T-Shirt" 'Baby sweater'
'Mio ALPHA Heart Rate Monitor/Sport Watch'
'Field & Stream Sportsman 16 Gun Fire Safe' 'Sports Books '
"Diamondback Boys' Insight 24 Performance Hybr"
'Polar Loop Activity Tracker' 'Garmin Forerunner 910XT GPS Watch' 'DVDs '
'CDs of rock' "Nike Kids' Grade School KD VI Basketball Shoe"
"Nike Women's Free 5.0 TR FIT PRT 4 Training S"
"Hirzl Women's Soffft Flex Golf Glove"
"The North Face Women's Recon Backpack" 'Lawn mower'
```

```
 'Nike Dri-FIT Crew Sock 6 Pack' "Nike Women's Legend V-Neck T-Shirt"
 'Garmin Approach S4 Golf GPS Watch' 'insta-bed Neverflat Air Mattress'
 "Nike Men's Kobe IX Elite Low Basketball Shoe" 'Adult dog supplies'
 'First aid kit' 'Garmin Approach S3 Golf GPS Watch' 'Rock music'
 'Fighting video games' 'Fitbit The One Wireless Activity & Sleep Trac'
 'Stiga Master Series ST3100 Competition Indoor'
 "Diamondback Girls' Clarity 24 Hybrid Bike 201"
 'adidas Brazuca 2014 Official Match Ball' 'GolfBuddy VT3 GPS Watch'
 'Bushnell Pro X7 Jolt Slope Rangefinder'
 'Yakima DoubleDown Ace Hitch Mount 4-Bike Rack'
 "Nike Men's Fingertrap Max Training Shoe"
 'Bowflex SelectTech 1090 Dumbbells' 'SOLE E35 Elliptical'
 "Hirzl Women's Hybrid Golf Glove" "Hirzl Men's Hybrid Golf Glove"
 'TaylorMade 2014 Purelite Stand Bag' 'Bag Boy Beverage Holder'
 'Bag Boy M330 Push Cart' 'Clicgear 8.0 Shoe Brush'
 'Titleist Small Wheeled Travel Cover' 'Clicgear Rovic Cooler Bag'
 'Titleist Club Glove Travel Cover' 'Ogio Race Golf Shoes'
 "LIJA Women's Argyle Golf Polo"
 "LIJA Women's Eyelet Sleeveless Golf Polo"
 "LIJA Women's Button Golf Dress"
 "LIJA Women's Mid-Length Panel Golf Shorts"
 "TaylorMade Women's RBZ SL Rescue"
 "Cleveland Golf Women's 588 RTX CB Satin Chrom"
 "Top Flite Women's 2014 XL Hybrid" 'MDGolf Pittsburgh Penguins Putter'
 'TaylorMade White Smoke IN-12 Putter'
 'Cleveland Golf Collegiate My Custom Wedge 588'
 "Merrell Men's All Out Flash Trail Running Sho"
 "Merrell Women's Grassbow Sport Waterproof Hik"
 "Merrell Women's Siren Mid Waterproof Hiking B"
 "Merrell Women's Grassbow Sport Hiking Shoe" 'Toys '
 'Pelican Sunstream 100 Kayak' 'Pelican Maverick 100X Kayak'
 "O'Brien Men's Neoprene Life Vest"]

Column: Shipping Mode
Number of Unique Values: 4
Unique Values: ['Standard Class' 'First Class' 'Second Class' 'Same Day']

Column: Customer Name
Number of Unique Values: 14033
Unique Values: ['Cally Holloway' 'Irene Luna' 'Gillian Maldonado' … 'Anika
Davenport'
 'Yuri Smith' 'Hyacinth Witt']
```

Converting the date columns into datetime objects

[18]:
```
df["order date (DateOrders)"]=pd.to_datetime(df["order date (DateOrders)"])
df=df.sort_values(by="order date (DateOrders)")
```

Lets change the columns name of order date (DateOrders) and shipping date (DateOrders).

[19]:
```
df.rename({
    'order date (DateOrders)': 'Order date',
}, axis = 1, inplace=True)
```

3 3. Descriptive and Diagnostic Analysis

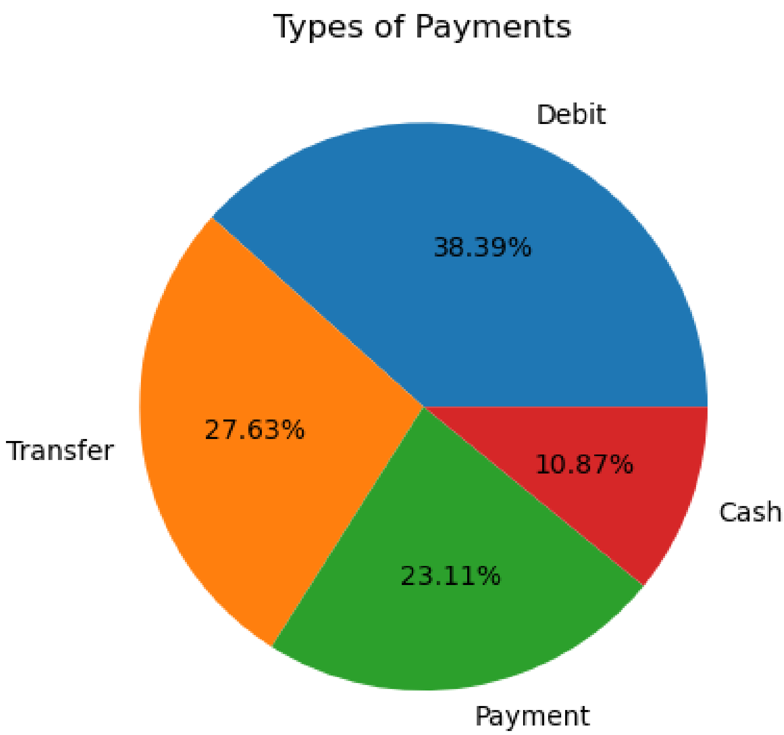
3.0.1 First of all, let's do univariate analysis.

Let's look at the value counts of type of payments. How many transactions are done by what type of payments?

[20]:
```
types_of_payment = df.Type.value_counts()
types_of_payment
```

[20]:
```
Type
DEBIT       69295
TRANSFER    49883
PAYMENT     41725
CASH        19616
Name: count, dtype: int64
```

[21]:
```
label=["Debit", "Transfer", "Payment", "Cash"]
plt.pie(types_of_payment ,labels=label, autopct="%.2f%%")
plt.title("Types of Payments")
plt.show()
```



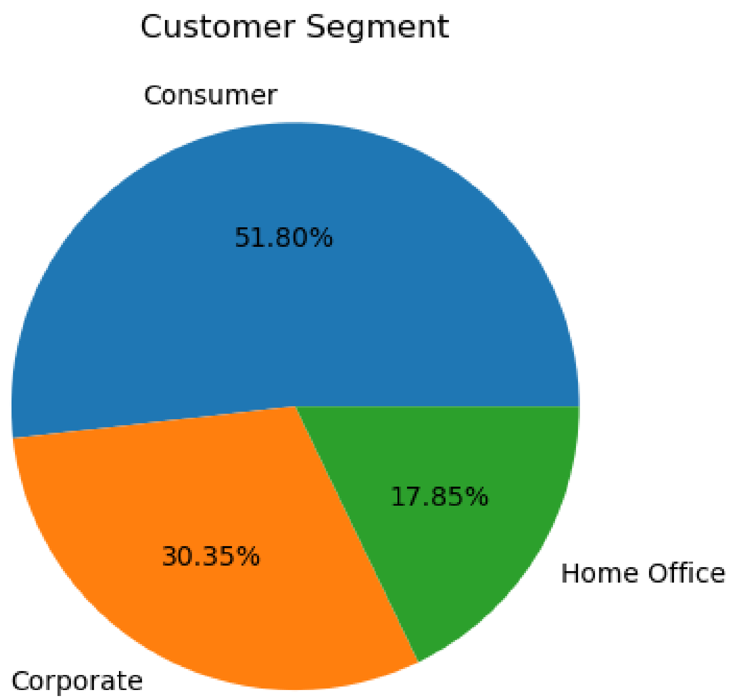
Customer Segment

[22]:
```
customer_segment=df["Customer Segment"].value_counts()
customer_segment
```

[22]:
```
Customer Segment
Consumer       93504
Corporate      54789
Home Office    32226
Name: count, dtype: int64
```

[23]:
```
label=["Consumer","Corporate","Home Office"]
plt.pie(customer_segment,labels=label, autopct="%.2f%%")
plt.title("Customer Segment")
```

[23]:
```
Text(0.5, 1.0, 'Customer Segment')
```

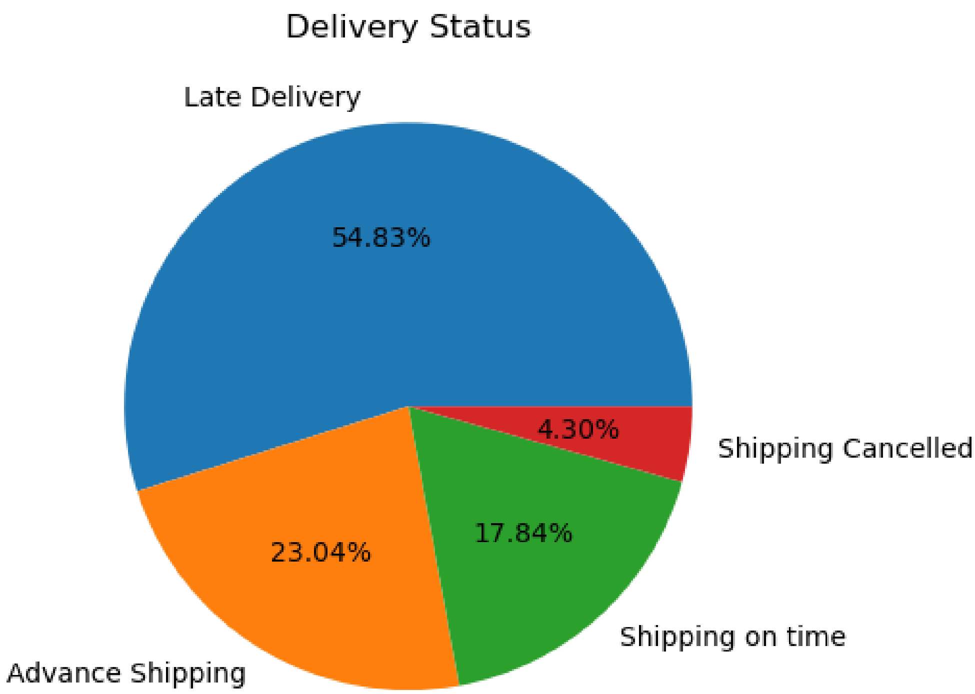


Lets look at the delivery status.

[24]:
```
delivery_status = df['Delivery Status'].value_counts()
delivery_status
```

[24]:
```
Delivery Status
Late delivery        98977
Advance shipping     41592
Shipping on time     32196
Shipping canceled     7754
Name: count, dtype: int64
```

[25]:
```
label=["Late Delivery", "Advance Shipping", "Shipping on time", "Shipping␣
↪Cancelled"]
plt.pie(delivery_status, labels=label, autopct="%.2f%%")
plt.title("Delivery Status")
plt.show()
```

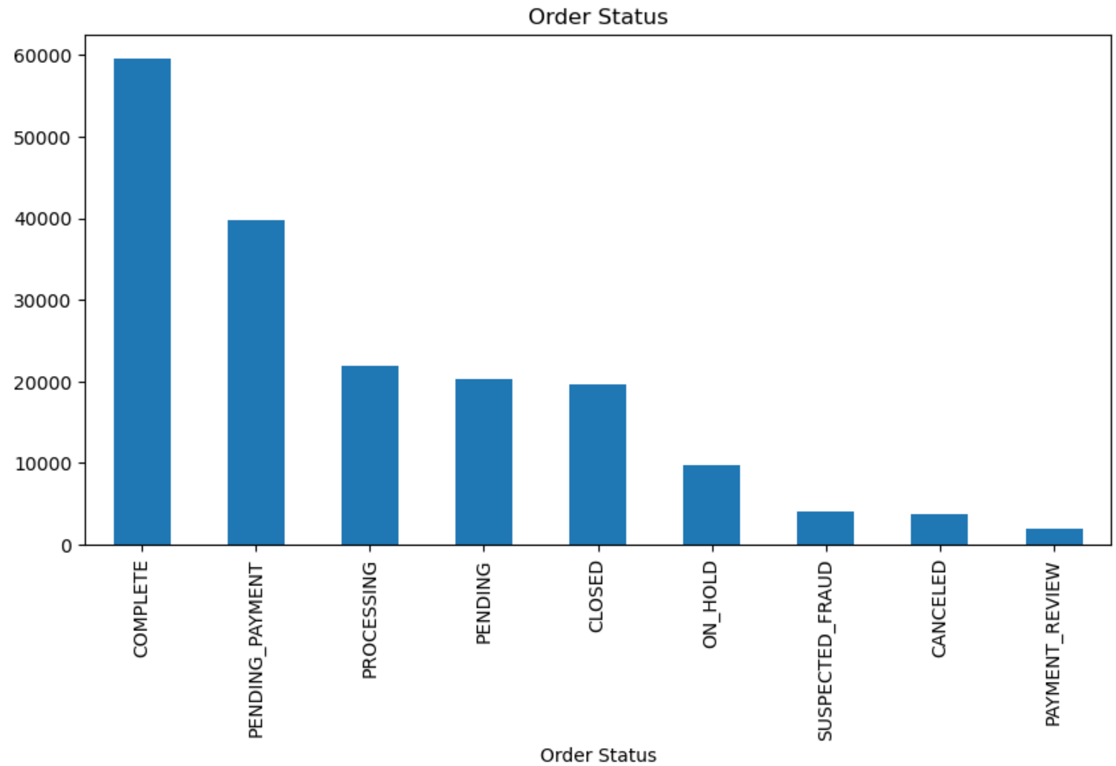


Here, we can see more than half of the delivery are late.

[26]:
```
order_status = df['Order Status'].value_counts()
order_status
```

[26]:
```
Order Status
COMPLETE           59491
PENDING_PAYMENT    39832
PROCESSING         21902
PENDING            20227
CLOSED             19616
ON_HOLD             9804
SUSPECTED_FRAUD     4062
CANCELED            3692
PAYMENT_REVIEW      1893
Name: count, dtype: int64
```

[27]:
```
order_status.plot.bar(figsize=(10,5))
plt.title("Order Status")
plt.show()
```

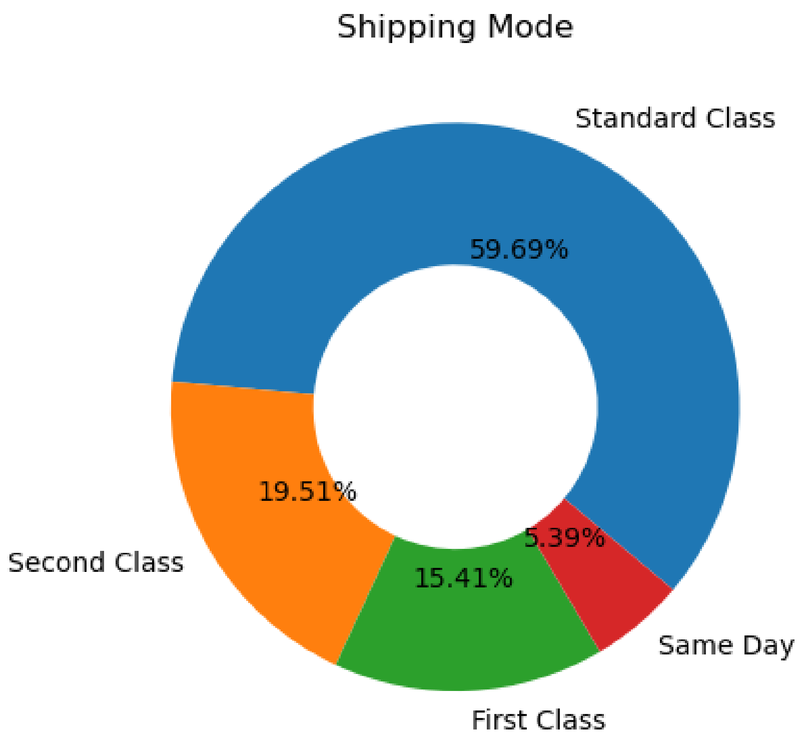


Let's have a look at the shipping mode.

[28]:
```
shipping_mode = df['Shipping Mode'].value_counts()
shipping_mode
```

[28]:
```
Shipping Mode
Standard Class    107752
Second Class       35216
First Class        27814
Same Day            9737
Name: count, dtype: int64
```

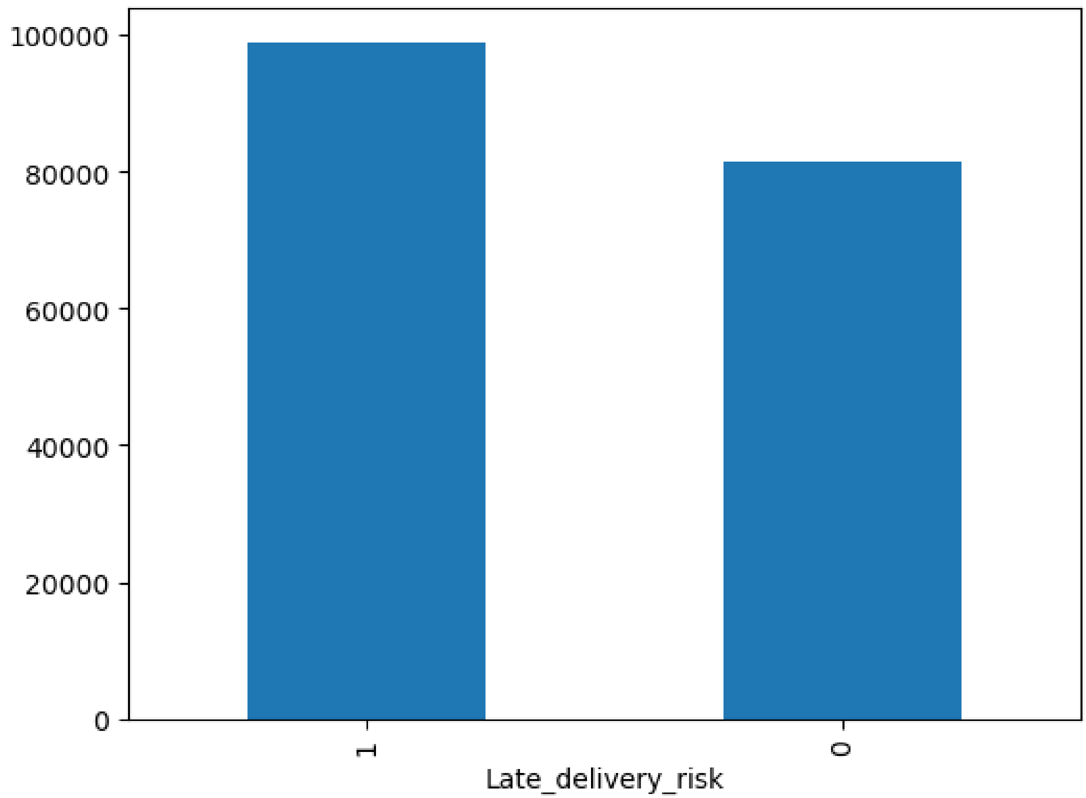
[29]:
```
label=["Standard Class", "Second Class", "First Class", "Same Day"]
plt.pie(shipping_mode ,labels=label, autopct="%.2f%%", wedgeprops=dict(width=0.
 ↪5), startangle=-40)
plt.title("Shipping Mode")
plt.show()
```



Looking at Late Delivery Risk

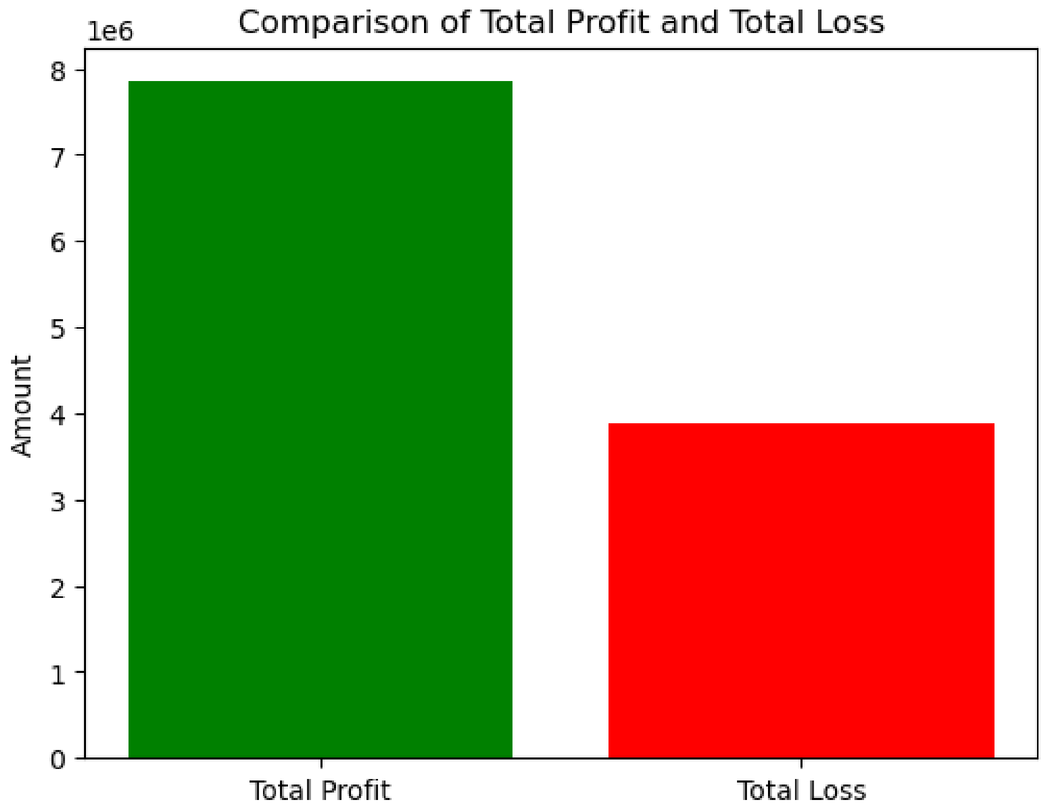
[30]:
```
print(df['Late_delivery_risk'].value_counts())
df['Late_delivery_risk'].value_counts().plot.bar()
plt.show()
```

```
Late_delivery_risk
1    98977
0    81542
Name: count, dtype: int64
```



Looking profit per order

[31]:
```
data = df['Profit Per Order']
total_profit = data[data> 0].sum()
total_loss = data[data < 0].sum()

categories = ['Total Profit', 'Total Loss']
values = [total_profit, abs(total_loss)]

# plt.figure(figsize=(8, 6))
plt.bar(categories, values, color=['green', 'red'])
plt.title('Comparison of Total Profit and Total Loss')
plt.ylabel('Amount')
plt.show()
```

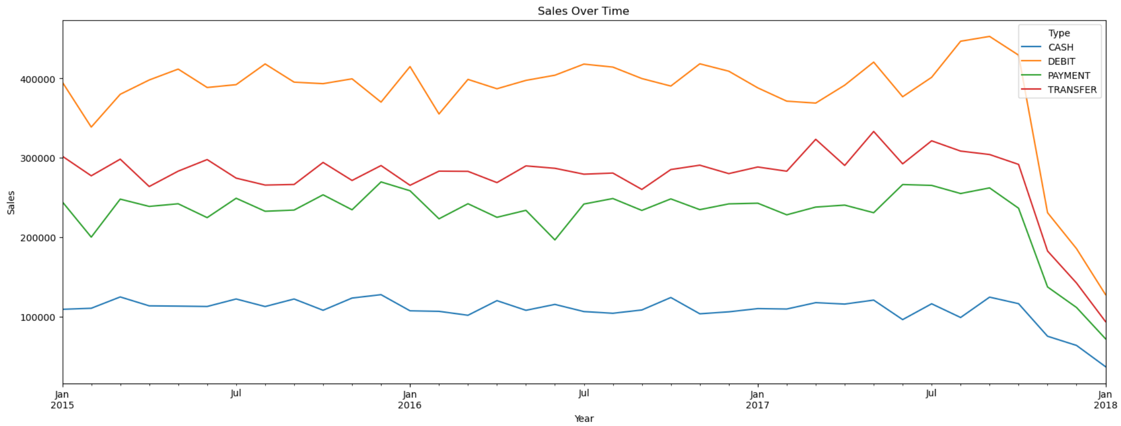


3.0.2 Now doing the bivariate and multivariate analysis

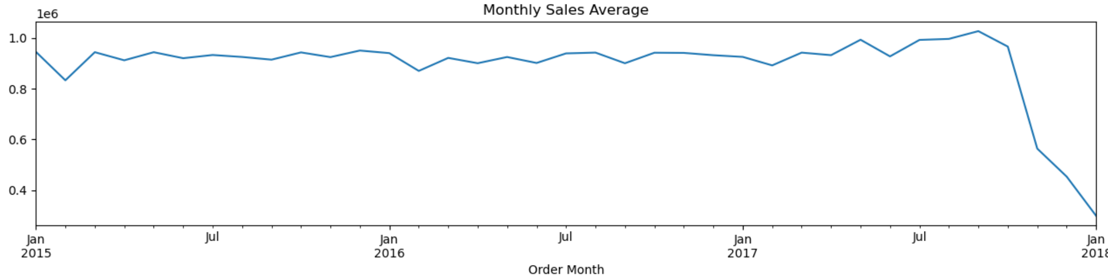
3.1 Looking at the Sales with different features

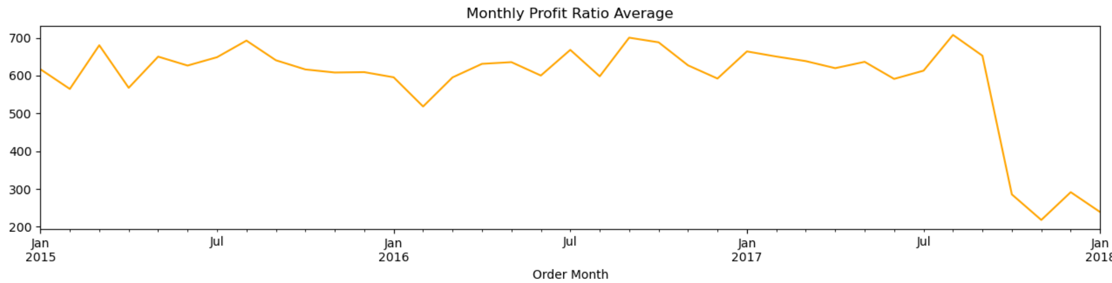
3.1.1 1. Sales trend over the year according to type

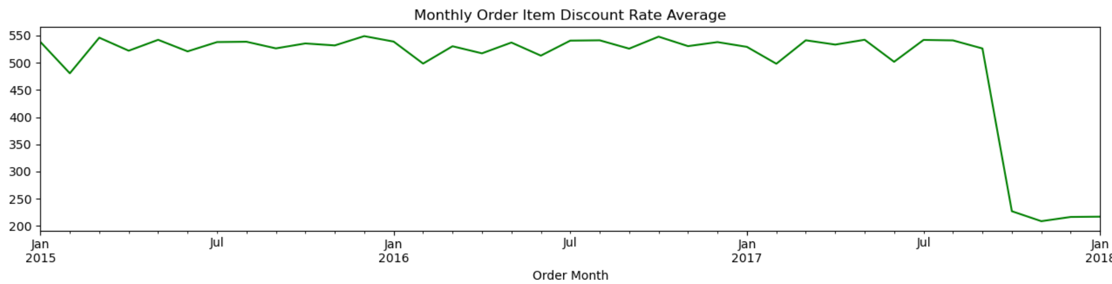
[32]:
```
df1 = df.copy()
df1['Order Month'] = df1['Order date'].dt.to_period('M')
sales = df1.groupby(['Order Month', 'Type'])['Sales'].sum().unstack()

sales.plot(kind = 'line',
           title = 'Sales Over Time',
           xlabel = 'Year',
           ylabel = 'Sales',
           figsize = (20, 7))
plt.show()
```



[33]:
```
df1 = df.copy()
df1['Order Month'] = df1['Order date'].dt.to_period('M')
monthly_sales_avg = df1.groupby(['Order Month'])['Sales per customer'].sum()
monthly_profit_ratio_avg = df1.groupby(['Order Month'])['Order Item Profit␣
↪Ratio'].sum()
monthly_discount_rate_avg = df1.groupby(['Order Month'])['Order Item Discount␣
↪Rate'].sum()

plt.figure(figsize=(13, 10))
plt.subplot(3, 1, 1)
monthly_sales_avg.plot(title='Monthly Sales Average')
plt.subplot(3, 1, 2)
monthly_profit_ratio_avg.plot(title='Monthly Profit Ratio Average',␣
↪color='orange')
plt.subplot(3, 1, 3)
monthly_discount_rate_avg.plot(title='Monthly Order Item Discount Rate␣
↪Average', color='green')
plt.tight_layout()
plt.show()
```

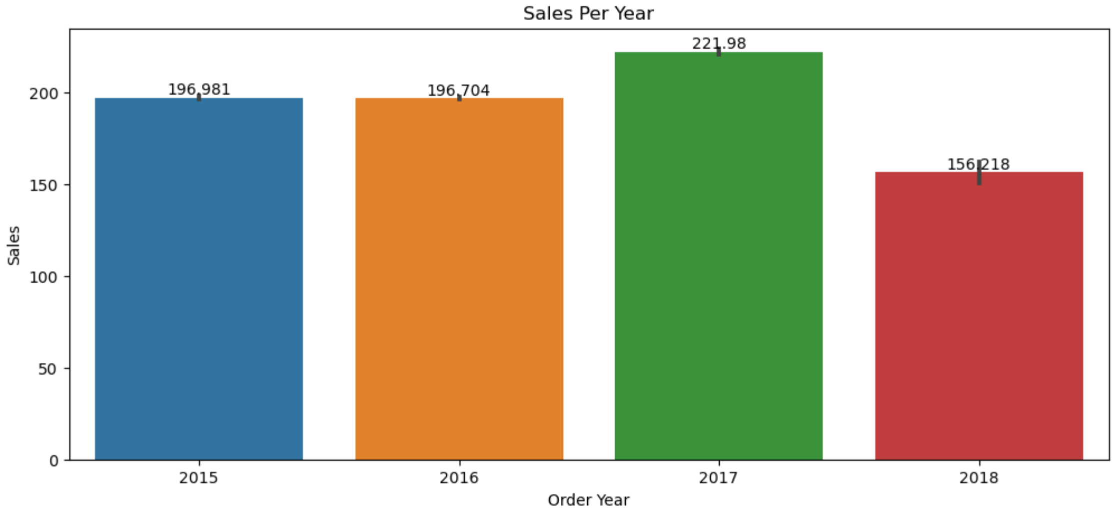






3.1.2 Sales according to different years

[34]:
```
df1['Order Year'] = df1['Order date'].dt.year
plt.figure(figsize = (12,5))
plt.title('Sales Per Year')
ax = sns.barplot(data = df1,
                 x = 'Order Year',
                 y = 'Sales');

for i in ax.containers:
    ax.bar_label(i)
```



[35]:
```
df1['Sales']
```

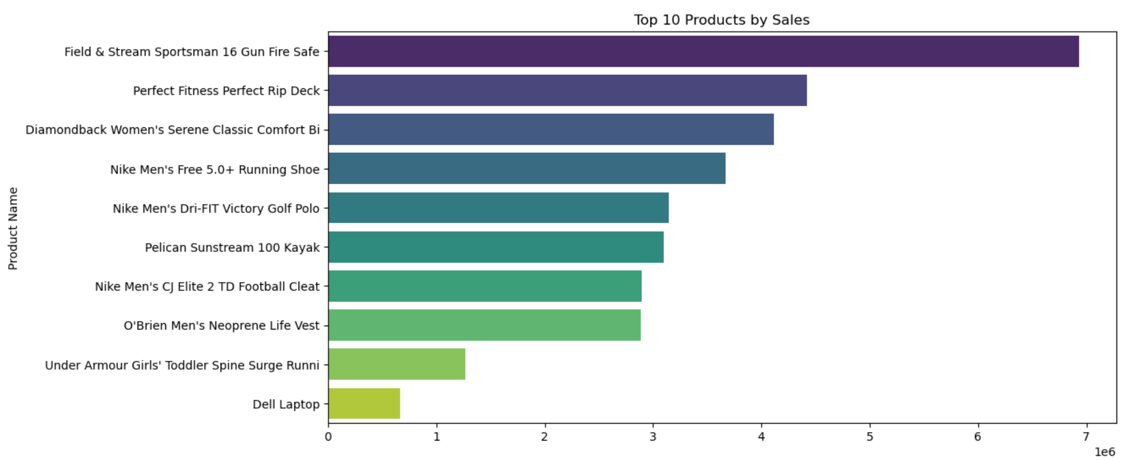
[35]:
```
33833     299.980011
77011     199.990005
109322    250.000000
87884     129.990005
114915    199.919998
             …
160537    215.820007
93905     215.820007
0         327.750000
52147      11.540000
17863      39.750000
Name: Sales, Length: 180519, dtype: float64
```

3.1.3 2. Top 10 Sales according to the Category

[36]:
```
sales_with_category = df.groupby('Category Name')['Sales'].sum().reset_index().
↪sort_values(by='Sales', ascending=False).head(10)
fig = px.bar(sales_with_category, x='Category Name', y='Sales', title='Top 10␣
↪category according to Sales')
fig.show()
```

3.1.4 3. Top 10 Product by Sales

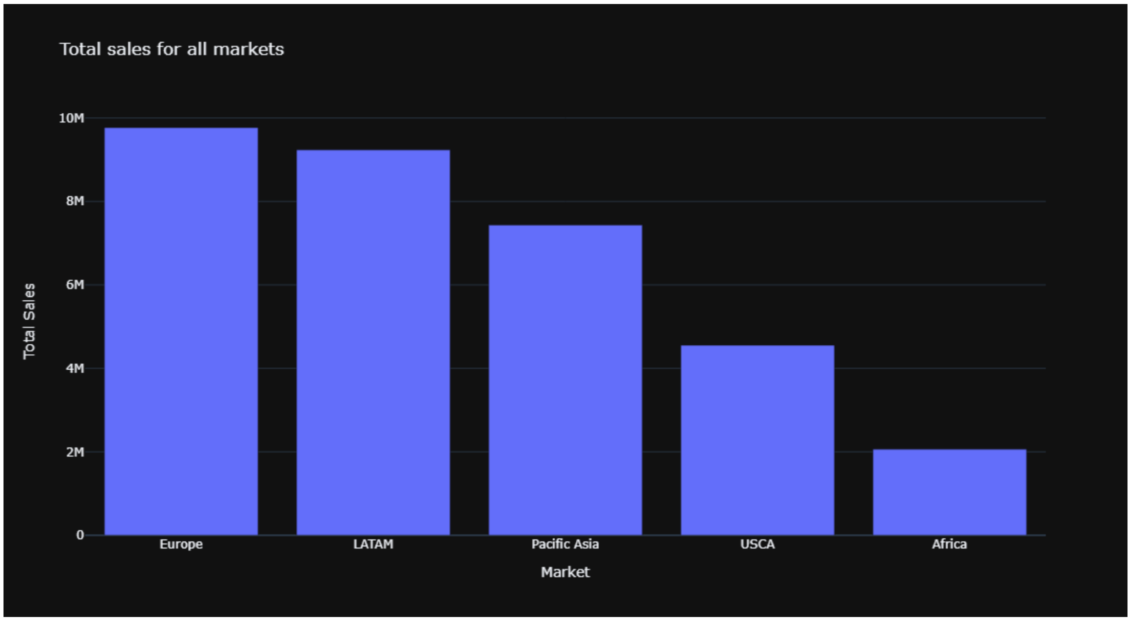
[37]:
```
top_products = df.groupby('Product Name')['Sales'].sum().
  ↪sort_values(ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(y=top_products.index, x=top_products.values, palette='viridis')
plt.title('Top 10 Products by Sales')
plt.show()
```



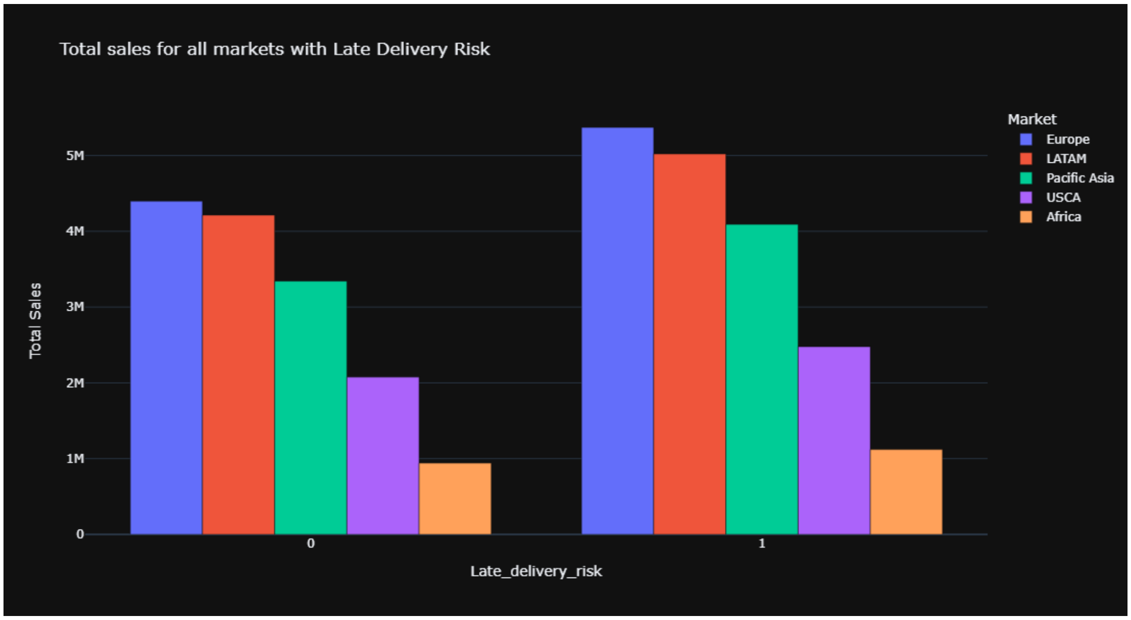
3.1.5 4. Sales by Market

[38]:
```
market = df.groupby('Market')

fig = px.bar(
    market['Sales per customer'].sum().sort_values(ascending=False).
  ↪reset_index(),
    x='Market',
```

```
    y='Sales per customer',
    title="Total sales for all markets",
    labels={'Sales per customer': 'Total Sales'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```

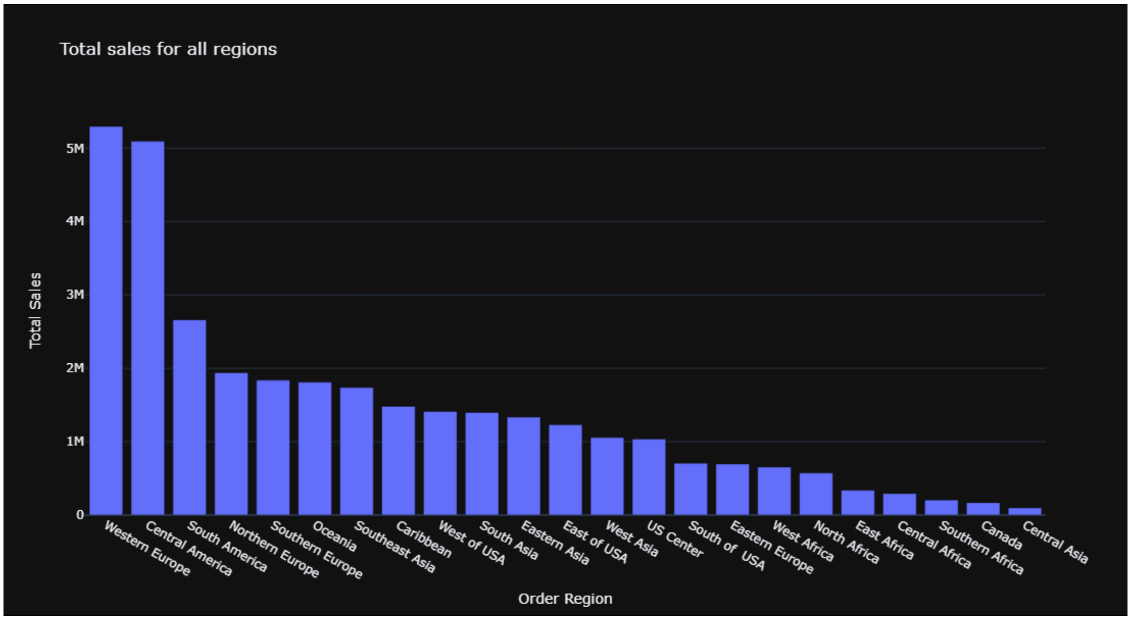


[39]:
```
market = df.groupby(['Market', 'Late_delivery_risk'])

fig = px.bar(
    market['Sales per customer'].sum().sort_values(ascending=False).
 ↪reset_index(),
    x='Late_delivery_risk',
    y='Sales per customer',
    title="Total sales for all markets with Late Delivery Risk",
    color='Market',
    barmode='group',
    labels={'Sales per customer': 'Total Sales'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



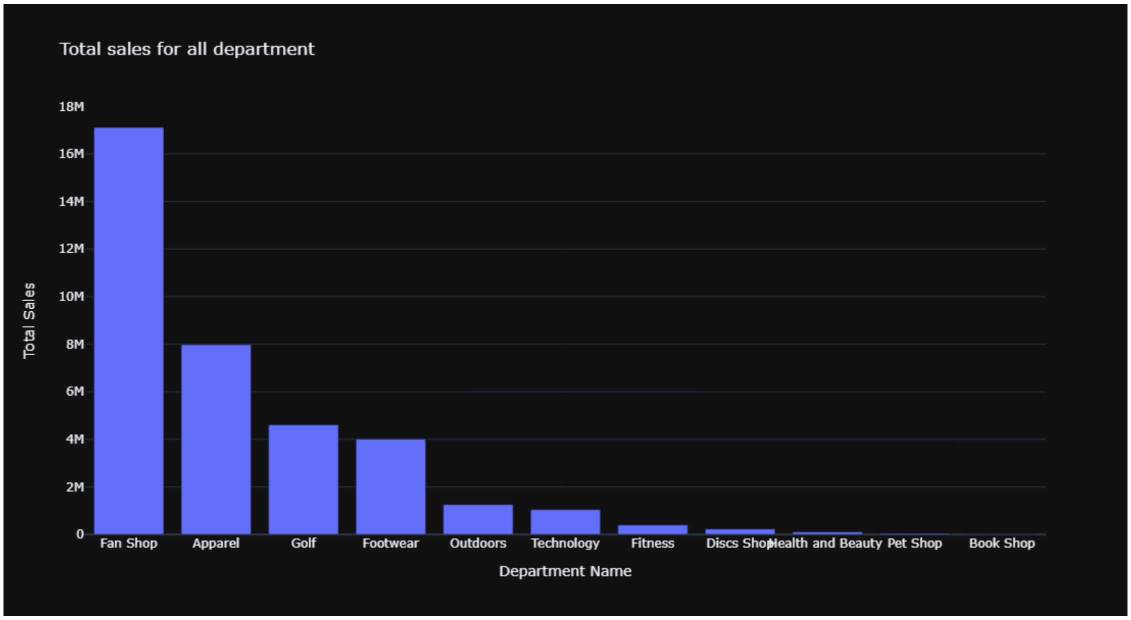
3.1.6 5. Sales by Region

[40]:
```
region = df.groupby('Order Region')

region_sales_per_customer = region['Sales per customer'].sum().
 ↪sort_values(ascending=False).reset_index()

fig = px.bar(
    region_sales_per_customer,
    x='Order Region',
    y='Sales per customer',
    title="Total sales for all regions",
    labels={'Sales per customer': 'Total Sales'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



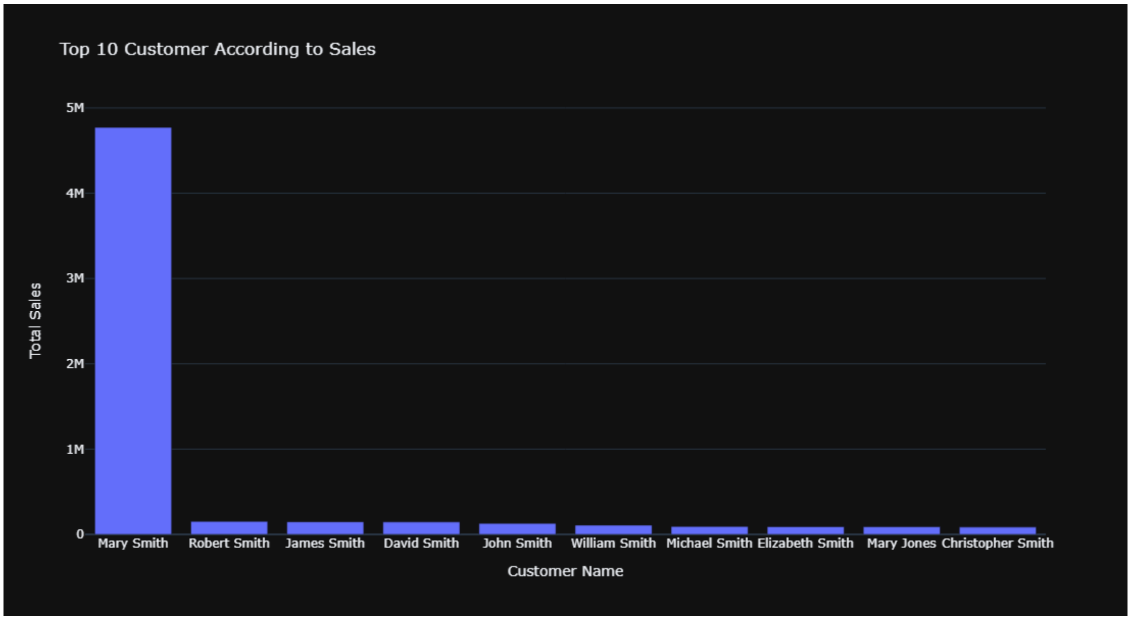
3.1.7 6. Sales Across all Department

[41]:
```
department = df.groupby('Department Name')

department_sales = department['Sales'].sum().sort_values(ascending=False).
↪reset_index()

fig = px.bar(
    department_sales,
    x='Department Name',
    y='Sales',
    title="Total sales for all department",
    labels={'Sales': 'Total Sales'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



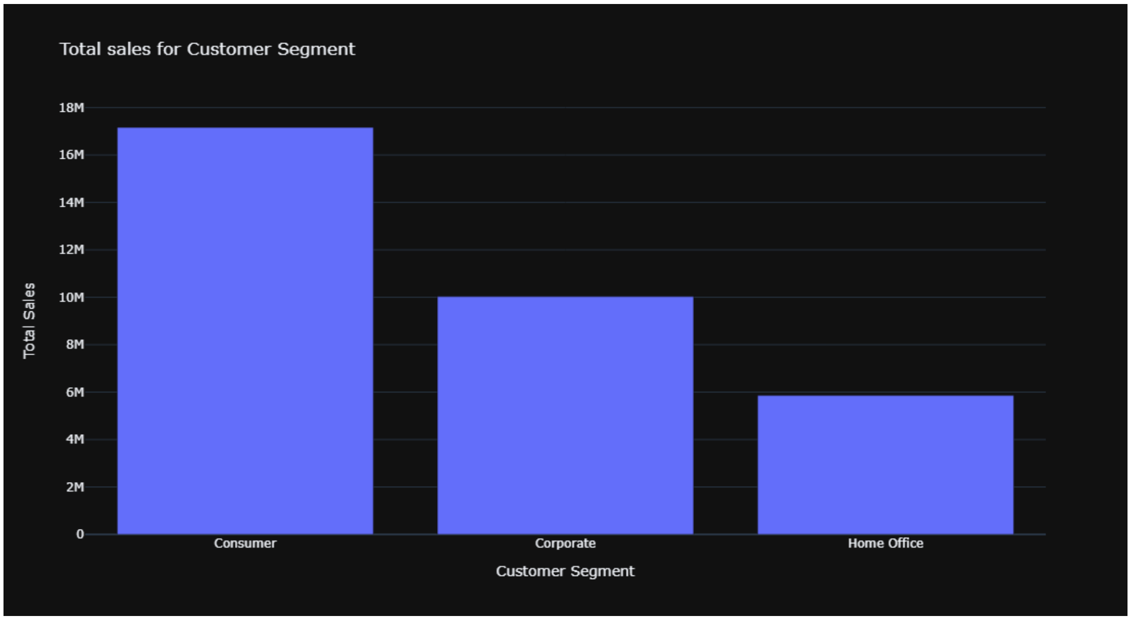
3.1.8 7. Top 10 Customer according to Sales

[42]:
```python
customer = df.groupby('Customer Name')

customer_sales = customer['Sales'].sum().sort_values(ascending=False).
↪reset_index().head(10)

fig = px.bar(
    customer_sales,
    x='Customer Name',
    y='Sales',
    title="Top 10 Customer According to Sales",
    labels={'Sales': 'Total Sales'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



3.1.9 8. Sales accroding to the Customer Segment

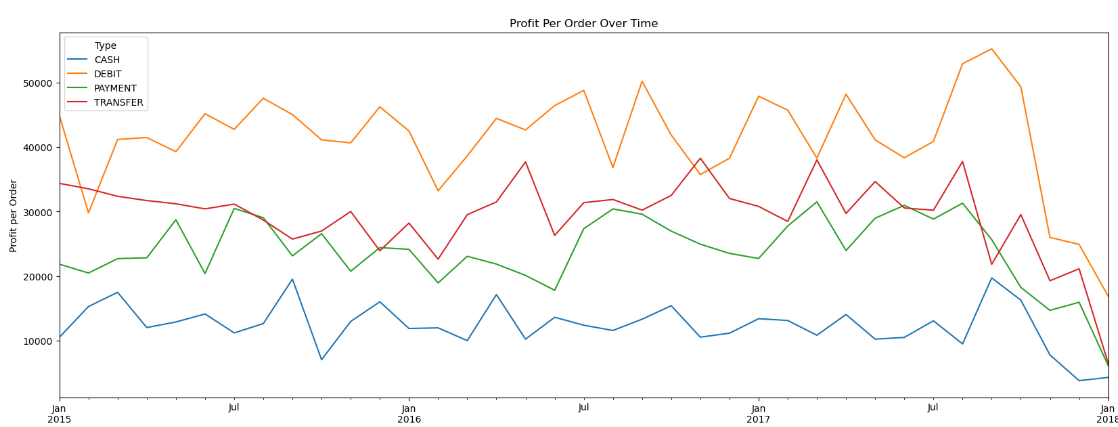
[43]:
```
customer_segment = df.groupby('Customer Segment')

fig = px.bar(
    customer_segment['Sales per customer'].sum().sort_values(ascending=False).
 ↪reset_index(),
    x='Customer Segment',
    y='Sales per customer',
    title="Total sales for Customer Segment",
    labels={'Sales per customer': 'Total Sales'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



3.2 Let's dive into the profit according to different features

3.2.1 1. Profit per order trend over the year with the type of payments

[44]:
```python
sales = df1.groupby(['Order Month', 'Type'])['Profit Per Order'].sum().unstack()

sales.plot(kind='line',
           title = 'Profit Per Order Over Time',
           xlabel = 'Year',
           ylabel = 'Profit per Order',
           figsize = (20, 7))
plt.show()
```



3.2.2 2. Top 10 Profit gaining Category

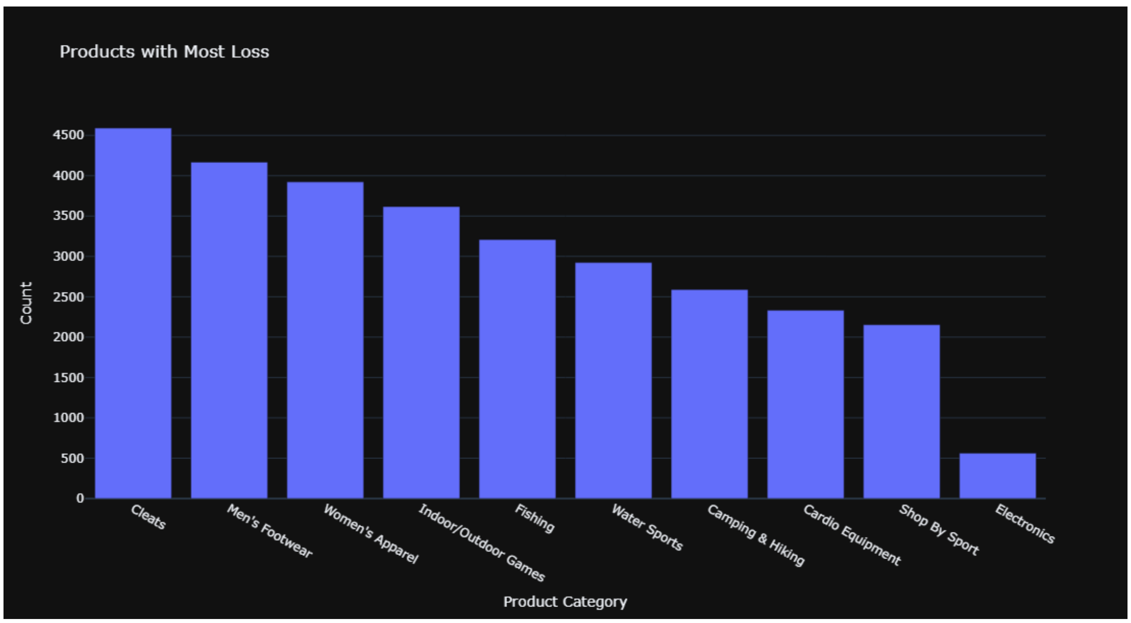
[45]:
```
sales_with_category = df.groupby('Category Name')['Profit Per Order'].sum().
    ↪reset_index().sort_values(by='Profit Per Order', ascending=False).head(10)
fig = px.bar(sales_with_category, x='Category Name', y='Profit Per Order',␣
    ↪title='Top 10 category according to Overall Profit')
fig.show()
```

3.2.3 3. Top 10 Less Profit Making Category

[46]:
```
sales_with_category = df.groupby('Category Name')['Profit Per Order'].sum().
    ↪reset_index().sort_values(by='Profit Per Order', ascending=True).head(10)
fig = px.bar(sales_with_category, x='Category Name', y='Profit Per Order',␣
    ↪title='Overall Bottom 10 category according to Profit')
fig.show()
```

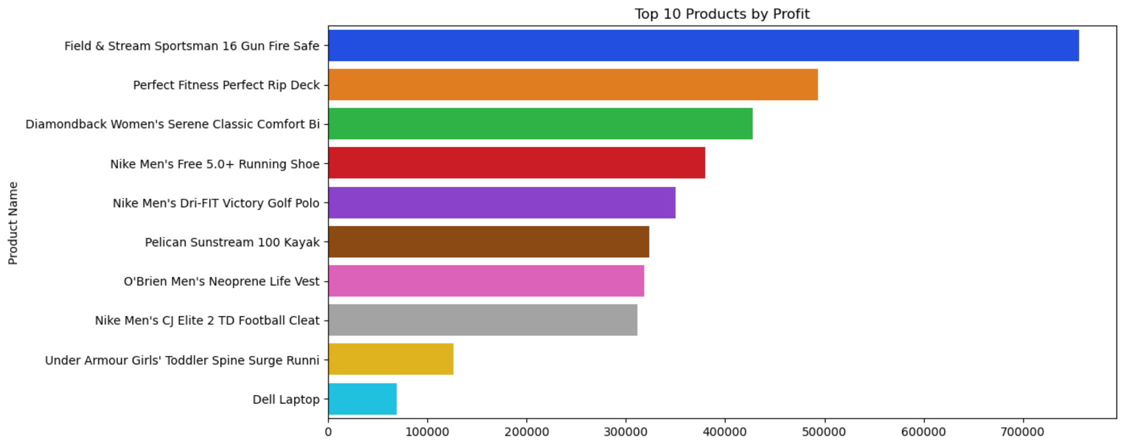
3.2.4 Category that have gone in loss that is profit less than 0.

[47]:
```
loss = df[df['Profit Per Order'] < 0]
loss_by_category = loss['Category Name'].value_counts().nlargest(10).
 ↪reset_index()
loss_by_category.columns = ['Category Name', 'Count']
fig1 = px.bar(
    loss_by_category,
    x='Category Name',
    y='Count',
    title='Products with Most Loss',
    labels={'Category Name': 'Product Category', 'Count': 'Count'},
    template='plotly_dark',
    width=800,
    height=600
)
fig1.show()
```



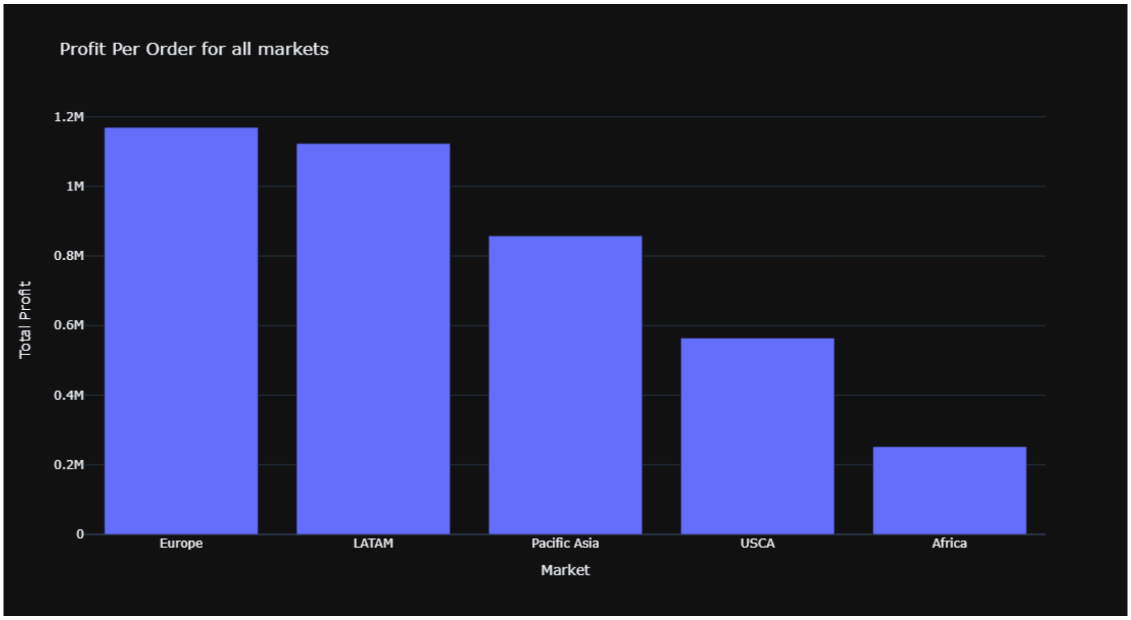
3.2.5 4. Top 10 Profit Gaining Products

[48]:
```
top_products = df.groupby('Product Name')['Profit Per Order'].sum().
 ↪sort_values(ascending=False).head(10)
plt.figure(figsize=(12,6))
sns.barplot(y=top_products.index, x=top_products.values, palette='bright')
plt.title('Top 10 Products by Profit')
plt.show()
```



3.2.6 5. Top 10 Profit Gaining Market

[49]:
```
market = df.groupby('Market')

fig = px.bar(
    market['Profit Per Order'].sum().sort_values(ascending=False).reset_index(),
    x='Market',
    y='Profit Per Order',
    title="Profit Per Order for all markets",
    labels={'Profit Per Order': 'Total Profit'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



3.2.7 Profit According to Geographical Region (Country, City, Latitude, Longitude)

[50]:
```
geographical = df.groupby(['Order Country', 'Order City'])['Profit Per Order'].
  ↪sum().reset_index(name='Profit of Orders').sort_values(by='Profit of␣
  ↪Orders', ascending=False)
print(geographical.head())

geographical_with_coords = geographical.merge(
    df[['Order Country', 'Order City', 'Latitude', 'Longitude']].
  ↪drop_duplicates(),
    on=['Order Country', 'Order City'],
    how='left'  # Use a left join to keep all rows from the geographical␣
  ↪DataFrame
)

print(geographical_with_coords.head())
geographical_with_coords_unique = geographical_with_coords.drop_duplicates(
    subset=['Order Country', 'Order City'],
    keep='first'  # Keep the first occurrence of each unique pair
)

print(geographical_with_coords_unique.head())

fig = px.choropleth(
    geographical,
    locationmode='country names',
```

```
    locations='Order Country',
    color='Profit of Orders',
    hover_name='Order Country',
    color_continuous_scale=px.colors.sequential.YlOrRd
)

highlight_points = geographical_with_coords_unique.loc[
    ((geographical_with_coords_unique['Order Country'].str.lower() == 'repblica␣
 ↪dominicana'.lower()) &
    (geographical_with_coords_unique['Order City'].str.lower() == 'santo␣
 ↪domingo'.lower())) |
    ((geographical_with_coords_unique['Order Country'].str.lower() == 'estados␣
 ↪unidos'.lower()) &
    (geographical_with_coords_unique['Order City'].str.lower() == 'new york␣
 ↪city'.lower()))
]

print(highlight_points)

highlight_scatter = px.scatter_geo(
    highlight_points,
    lat='Latitude',
    lon='Longitude',
    text='Profit of Orders',
    size='Profit of Orders',
    size_max=10,
    color_discrete_sequence=['red']
)

fig.add_trace(highlight_scatter.data[0])
fig.show()
```

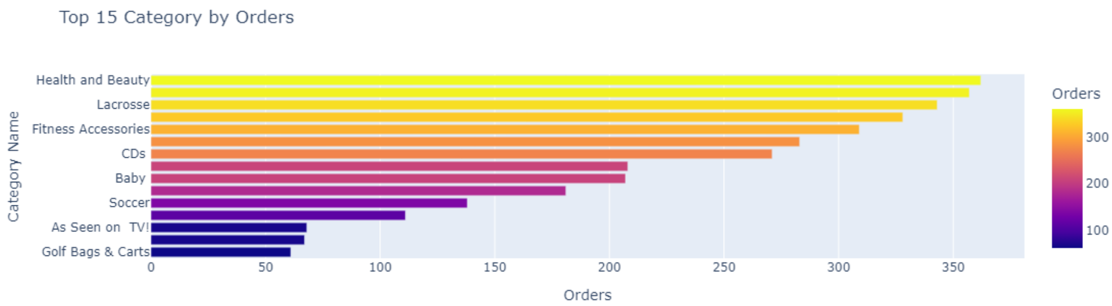
```
             Order Country     Order City  Profit of Orders
3260  República Dominicana  Santo Domingo      51111.670019
1492        Estados Unidos  New York City      47889.759868
2152              Honduras    Tegucigalpa      40973.640056
1430        Estados Unidos    Los Angeles      38014.360024
2837             Nicaragua        Managua      34319.950107
          Order Country     Order City  Profit of Orders   Latitude  \
0  República Dominicana  Santo Domingo      51111.670019  40.763580
1  República Dominicana  Santo Domingo      51111.670019  18.265211
2  República Dominicana  Santo Domingo      51111.670019  29.384306
3  República Dominicana  Santo Domingo      51111.670019  40.659874
4  República Dominicana  Santo Domingo      51111.670019  18.285450

   Longitude
0 -73.830040
```

```
1   -66.370552
2 -100.750252
3 -112.002869
4   -66.370621
             Order Country     Order City  Profit of Orders   Latitude  \
0     República Dominicana  Santo Domingo      51111.670019  40.763580
694         Estados Unidos  New York City      47889.759868  28.632990
1383              Honduras    Tegucigalpa      40973.640056  34.086327
1925        Estados Unidos    Los Angeles      38014.360024  18.203976
2519             Nicaragua        Managua      34319.950107  18.218407

       Longitude
0     -73.830040
694   -81.454185
1383 -117.959534
1925  -66.370544
2519  -66.370544
      Order Country     Order City  Profit of Orders  Latitude  Longitude
694  Estados Unidos  New York City      47889.759868  28.63299 -81.454185
```

3.3 Diving into Orders with other features

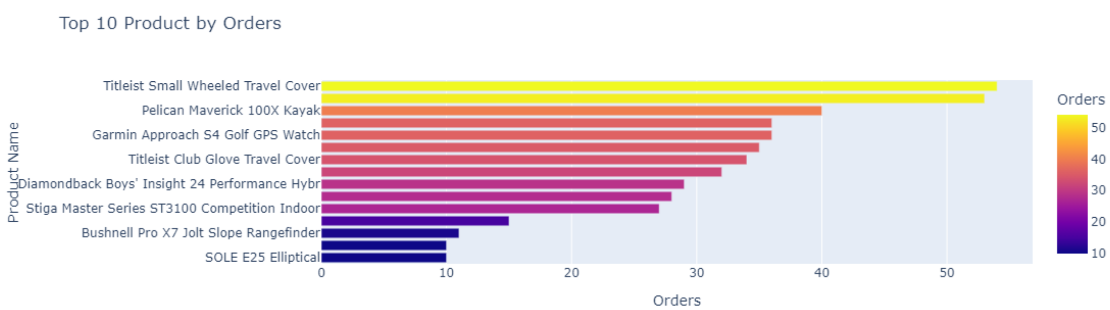
3.3.1 1. Looking at the Maximum Order according to Category

[51]:
```
category_name=df.groupby(["Category Name"])["Order Id"].count().
 ↪reset_index(name="Orders").sort_values(by="Orders",ascending=True).head(15)
px.bar(category_name,x=category_name["Orders"],y=category_name["Category␣
 ↪Name"], color=category_name["Orders"], labels={"Category Name":"Category␣
 ↪Name","Orders":"Orders"},
       title="Top 15 Category by Orders")
```



3.3.2 2. Top 10 Product with highest order

[52]:
```
product_name = df.groupby(["Product Name"])["Order Id"].count().reset_index(name="Orders").sort_values(by="Orders",ascending=True).head(15)
px.bar(product_name, x=product_name["Orders"], y=product_name["Product Name"], color=product_name["Orders"],
       labels={"Product Name":"Product Name","Orders":"Orders"}, title="Top 10 Product by Orders")
```

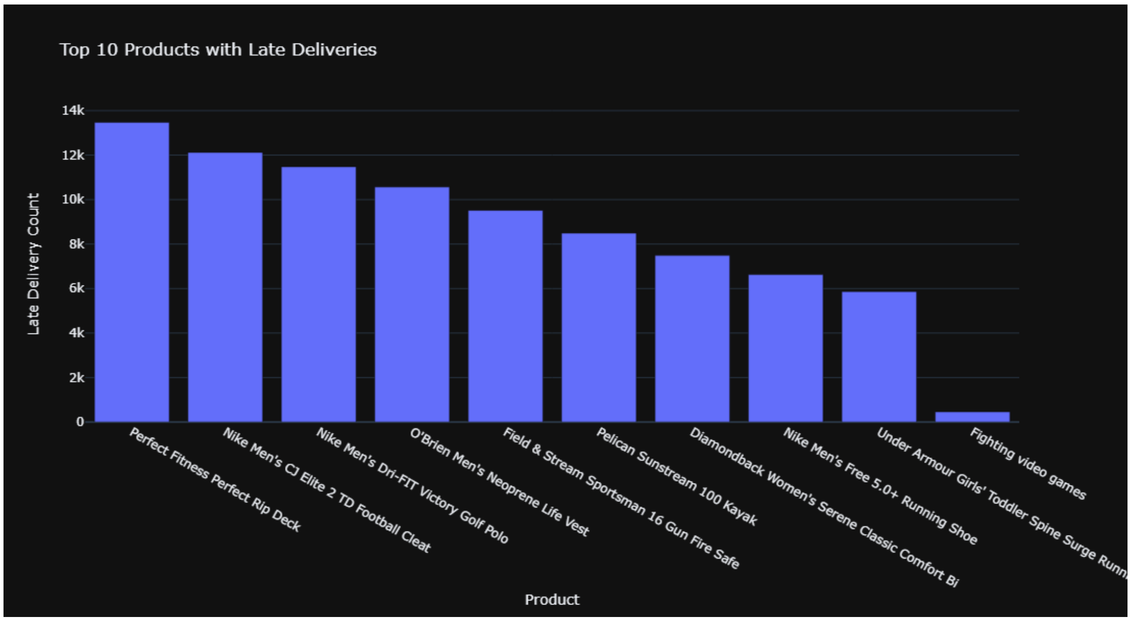


3.3.3 Let's Dive into Delivery Status

3.3.4 1. Top 10 Late delivered Product

[53]:
```
late_delivery_data = df[df['Delivery Status'] == 'Late delivery']
late_by_product = late_delivery_data['Product Name'].value_counts().nlargest(10).reset_index()
late_by_product.columns = ['Product Name', 'Late Deliveries']
fig = px.bar(
    late_by_product,
    x='Product Name',
```

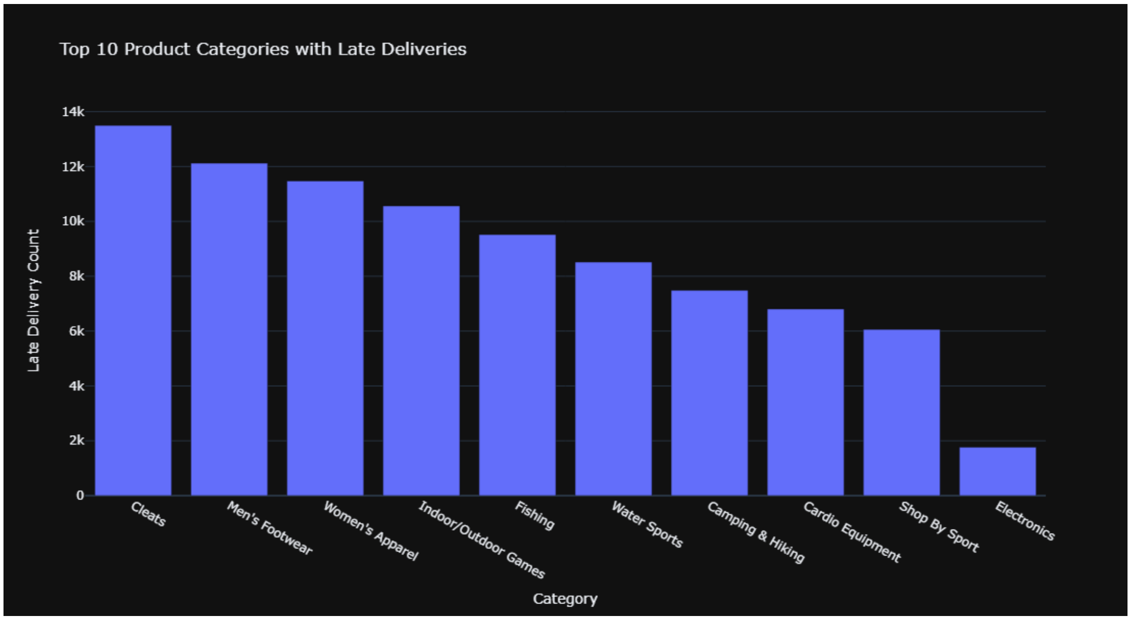
```
    y='Late Deliveries',
    title='Top 10 Products with Late Deliveries',
    labels={'Product Name': 'Product', 'Late Deliveries': 'Late Delivery␣
↪Count'},
    template='plotly_dark',
    width=1000,
    height=600
)
fig.show()
```



3.3.5 2. Top 10 Late Delivered Category

[54]:
```
late_delivery_data = df[df['Delivery Status'] == 'Late delivery']
late_by_product = late_delivery_data['Category Name'].value_counts().
↪nlargest(10).reset_index()
late_by_product.columns = ['Category Name', 'Late Deliveries']
fig = px.bar(
    late_by_product,
    x='Category Name',
    y='Late Deliveries',
    title='Top 10 Product Categories with Late Deliveries',
    labels={'Category Name': 'Category', 'Late Deliveries': 'Late Delivery␣
↪Count'},
    template='plotly_dark',
    width=800,
    height=600
```

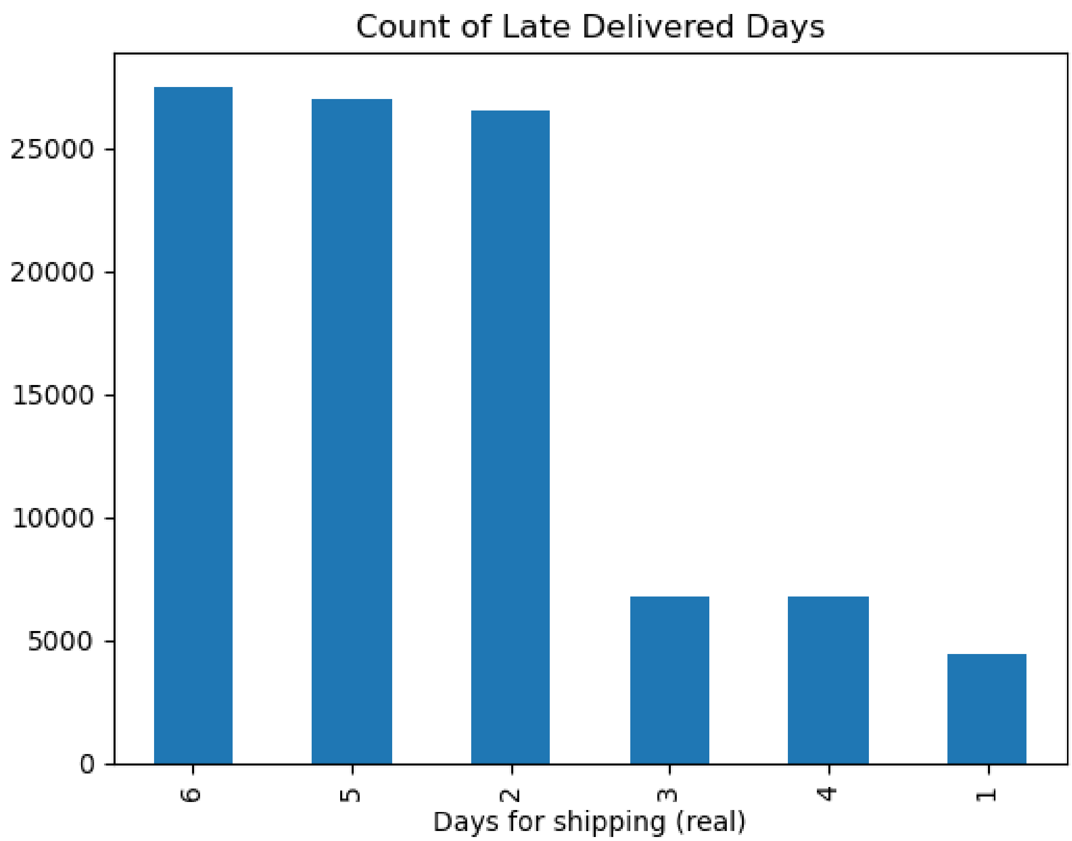
```
)
fig.show()
```



[55]:
```
late_delivered_day = late_delivery_data['Days for shipping (real)'].
    ↪value_counts()
late_delivered_day
```

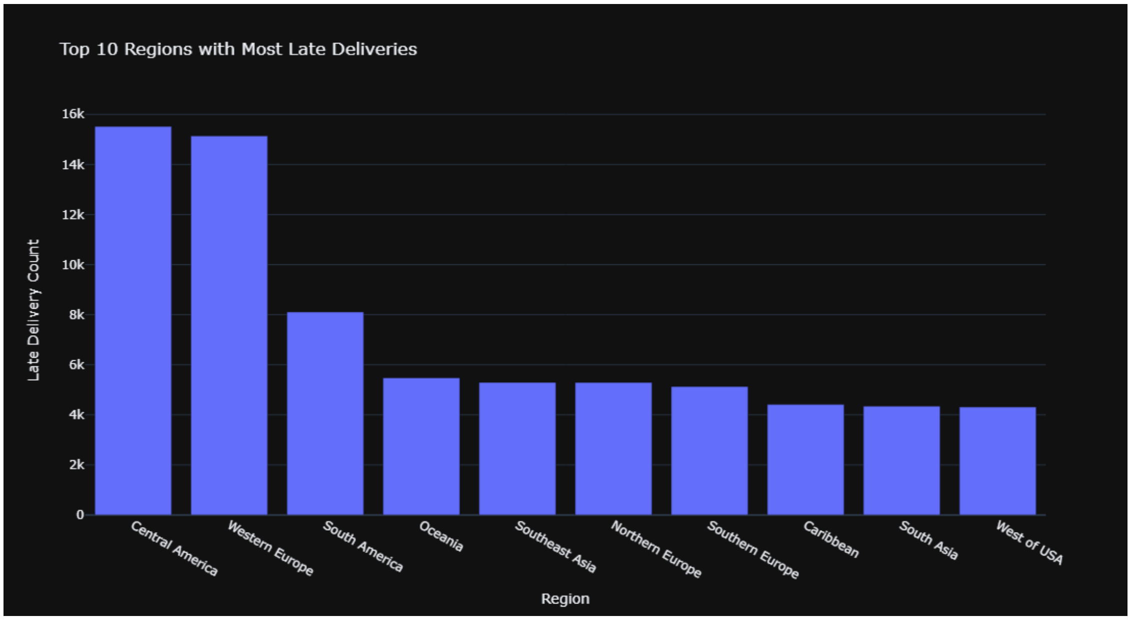
[55]:
```
Days for shipping (real)
6    27489
5    27003
2    26513
3     6759
4     6759
1     4454
Name: count, dtype: int64
```

[56]:
```
late_delivered_day.plot(kind='bar',
                        title='Count of Late Delivered Days')
plt.show()
```



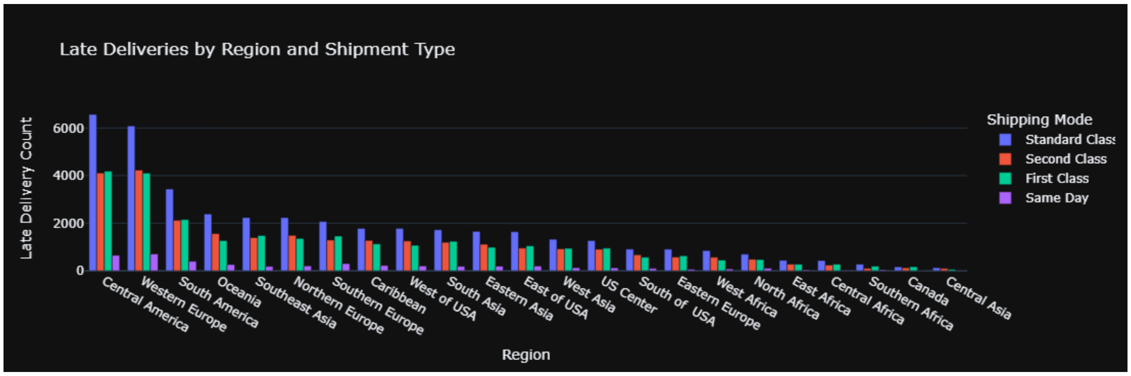
3.3.6 3. Top 10 Region with Late Deliveries

[57]:
```
late_by_region = late_delivery_data['Order Region'].value_counts().nlargest(10).
    ↪reset_index()
late_by_region.columns = ['Order Region', 'Late Deliveries']
fig = px.bar(
    late_by_region,
    x='Order Region',
    y='Late Deliveries',
    title='Top 10 Regions with Most Late Deliveries',
    labels={'Order Region': 'Region', 'Late Deliveries': 'Late Delivery Count'},
    template='plotly_dark',
    width=800,
    height=600
)

# Show the plot
fig.show()
```



3.3.7 4. Region with Late Deliveries and Shipment Mode

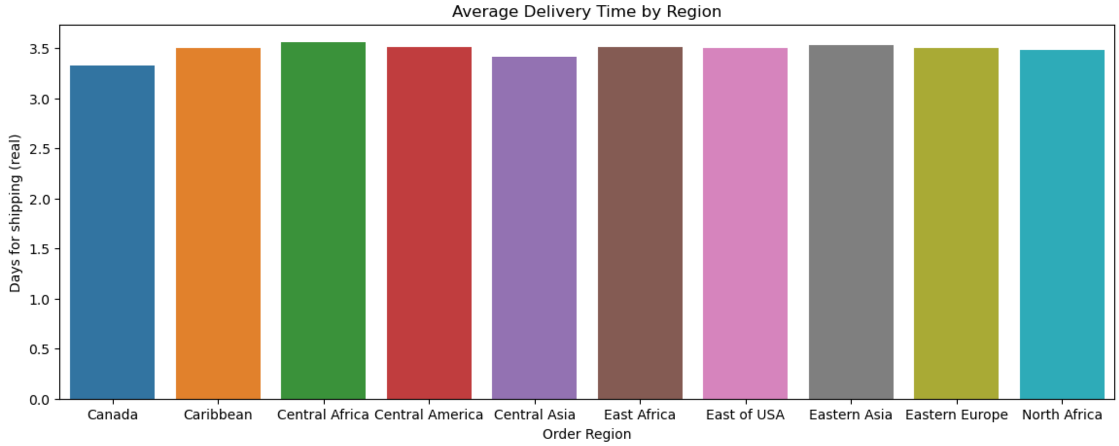
[58]:
```
late_by_region_shipment = late_delivery_data.groupby(['Order Region', 'Shipping␣
↪Mode']).size().reset_index(name='Late Deliveries')
late_by_region_shipment = late_by_region_shipment.sort_values(by='Late␣
↪Deliveries', ascending=False)
# Plotting the late deliveries by region and shipment type using Plotly Express␣
↪bar plot
fig = px.bar(
    late_by_region_shipment,
    x='Order Region',
    y='Late Deliveries',
    color='Shipping Mode',
    barmode='group',
    title='Late Deliveries by Region and Shipment Type',
    labels={'Order Region': 'Region', 'Late Deliveries': 'Late Delivery Count'},
    template='plotly_dark',

)

fig.show()
```



3.3.8 Average Delivery Days

[59]:
```python
# Example: Group by region and calculate mean delivery time
region_analysis = df.groupby('Order Region')['Days for shipping (real)'].mean().
    ↪reset_index().head(10)
print(region_analysis)

# Visualize the results
plt.figure(figsize=(14, 5))
sns.barplot(x='Order Region', y='Days for shipping (real)',␣
    ↪data=region_analysis)
plt.title('Average Delivery Time by Region')
plt.show()
```

```
     Order Region  Days for shipping (real)
0          Canada                  3.331595
1       Caribbean                  3.507213
2  Central Africa                  3.560525
3 Central America                  3.510462
4    Central Asia                  3.417722
5     East Africa                  3.514579
6     East of USA                  3.500940
7    Eastern Asia                  3.527335
8  Eastern Europe                  3.500765
9    North Africa                  3.480507
```



3.3.9 Finding which payment method is used to conduct frauds can be useful to prevent fraud from happening in future.

[60]:
```python
data=df[(df['Order Status'] == 'SUSPECTED_FRAUD')]
data['Type'].value_counts()
```

[60]:
```
Type
TRANSFER    4062
Name: count, dtype: int64
```

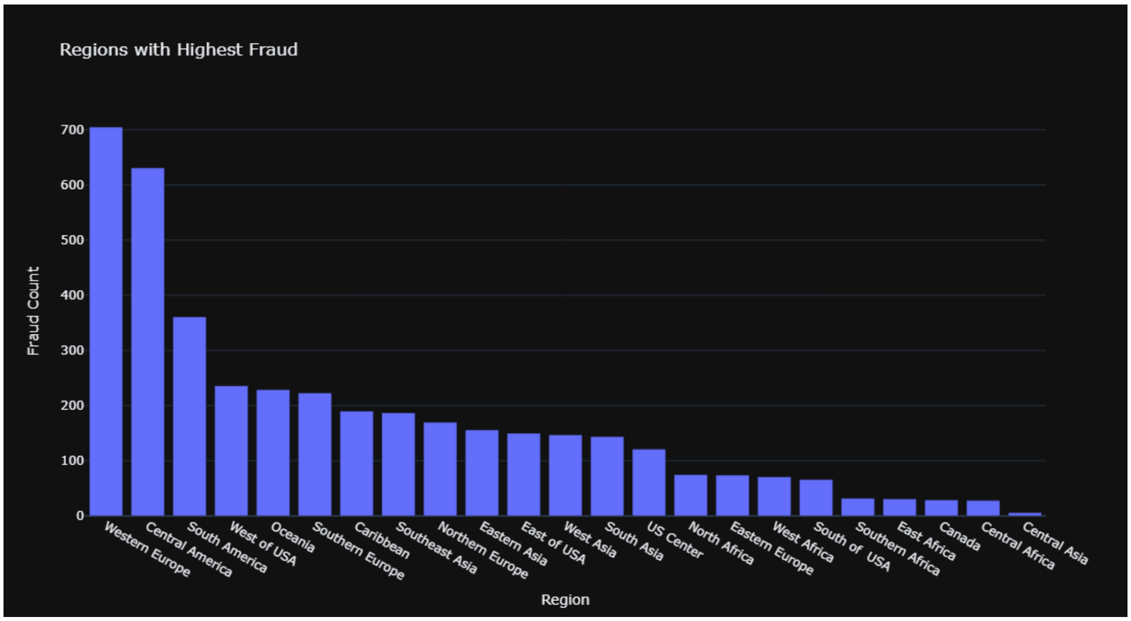
[61]:
```python
data=df[(df['Type'] != 'TRANSFER')&(df['Order Status'] == 'SUSPECTED_FRAUD')]
data['Order Region'].value_counts()
```

[61]:
```
Series([], Name: count, dtype: int64)
```

3.3.10 1. Regions with most frauds

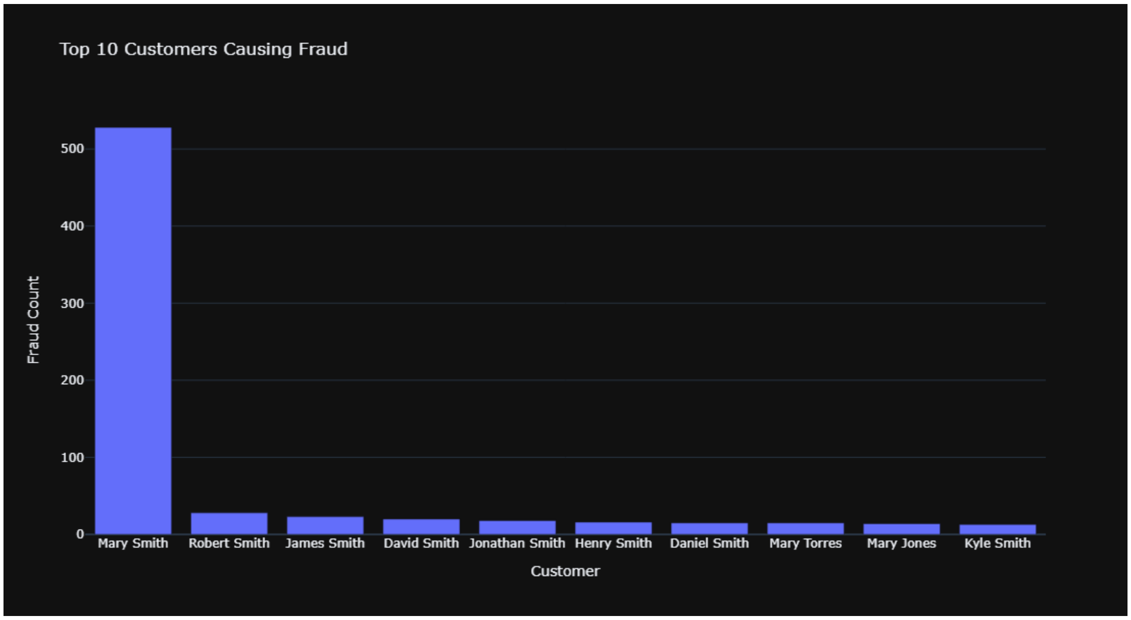
[62]:
```python
fraud_data = df[(df['Order Status'] == 'SUSPECTED_FRAUD') & (df['Type'] ==␣
↪'TRANSFER')]

fraud_by_region = fraud_data['Order Region'].value_counts().reset_index()

fraud_by_region.columns = ['Order Region', 'Count']

fraud_by_region = fraud_by_region.sort_values(by='Count', ascending=False)

fig = px.bar(
    fraud_by_region,
    x='Order Region',
    y='Count',
    title='Regions with Highest Fraud',
```

```
    labels={'Order Region': 'Region', 'Count': 'Fraud Count'},
    template='plotly_dark',
    width=800,
    height=600
)

fig.show()
```

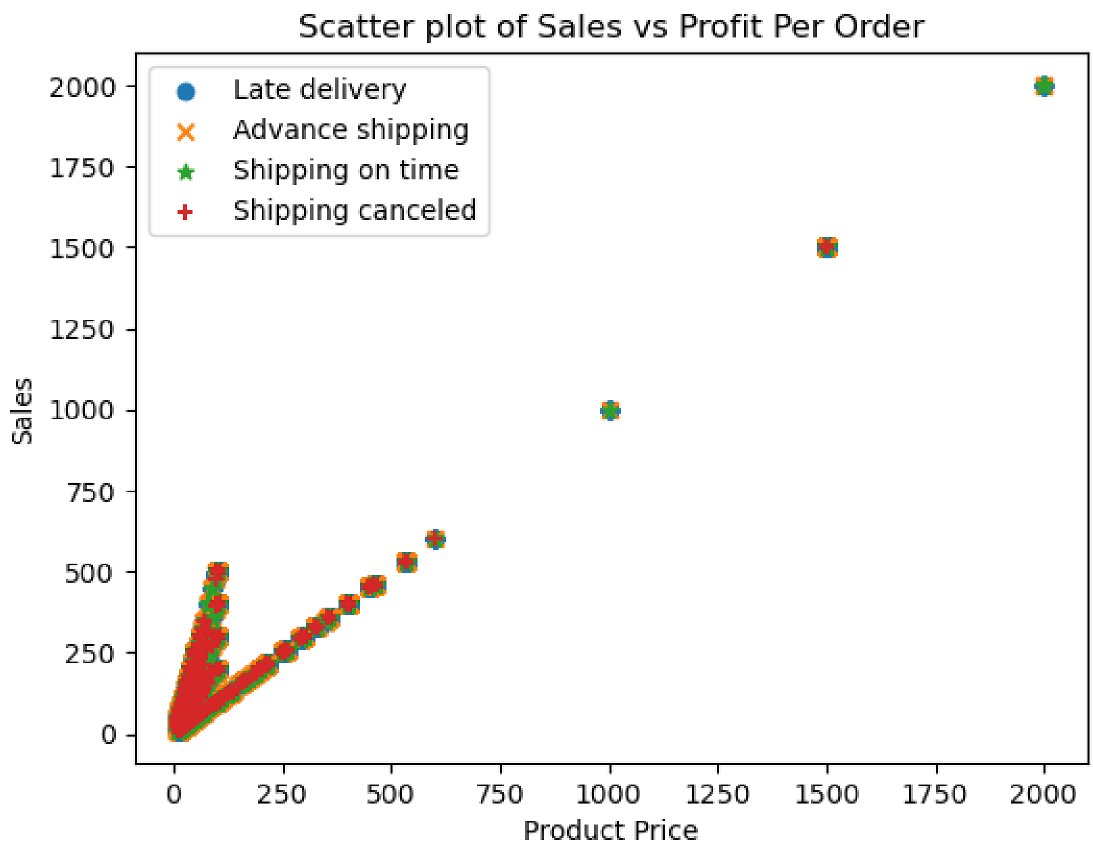


3.3.11 2. Customers with most frauds

[63]:
```
suspected_fraud_data = df[df['Order Status'] == 'SUSPECTED_FRAUD']
fraud_by_customer = suspected_fraud_data['Customer Name'].value_counts().
 ↪nlargest(10).reset_index()
fraud_by_customer.columns = ['Customer Name', 'Fraud Count']
fig = px.bar(
    fraud_by_customer,
    x='Customer Name',
    y='Fraud Count',
    title='Top 10 Customers Causing Fraud',
    labels={'Customer Name': 'Customer', 'Fraud Count': 'Fraud Count'},
    template='plotly_dark',
    width=800,
    height=600
)
fig.show()
```



[64]:
```
plt.scatter(df[df["Delivery Status"] == 'Late delivery']["Product Price"],␣
    ↪df[df["Delivery Status"] == 'Late delivery']["Sales"], label='Late␣
    ↪delivery', marker='o')
plt.scatter(df[df["Delivery Status"] == 'Advance shipping']["Product Price"],␣
    ↪df[df["Delivery Status"] == 'Advance shipping']["Sales"], label='Advance␣
    ↪shipping', marker='x')
plt.scatter(df[df["Delivery Status"] == 'Shipping on time']["Product Price"],␣
    ↪df[df["Delivery Status"] == 'Shipping on time']["Sales"], label='Shipping on␣
    ↪time', marker='*')
plt.scatter(df[df["Delivery Status"] == 'Shipping canceled']["Product Price"],␣
    ↪df[df["Delivery Status"] == 'Shipping canceled']["Sales"], label='Shipping␣
    ↪canceled', marker='+')
# Set the title and labels for the axes
plt.title('Scatter plot of Sales vs Profit Per Order')
plt.xlabel('Product Price')
plt.ylabel('Sales')
# Add a legend
plt.legend()
# Show the plot
plt.show()
```



4 4. Prediction Model to Detect Fake Orders and Suspicious Transactions

Here we can look at the columns which are not useful or no impact in forcasting sales of different product and also for detecting fake orders and suspicious transactions.

From here, we can drop 'Days for shipment (scheduled)' as it is the scheduled by the organization, Latitude and Longitude can also be dropped, Order Item Discount can be dropped as Order Item Discount Rate is present, Sales Per Customer can also be dropped because it is related with Sales and Order Item Discount Rate.

[65]:
```
df2 = df.copy()
```

[66]:
```
df2.columns
```

[66]:
```
Index(['Type', 'Days for shipping (real)', 'Sales per customer',
       'Delivery Status', 'Late_delivery_risk', 'Category Name',
       'Customer City', 'Customer Country', 'Customer Segment',
       'Department Name', 'Latitude', 'Longitude', 'Market', 'Order City',
       'Order Country', 'Order date', 'Order Id', 'Order Item Discount',
```

```
       'Order Item Discount Rate', 'Order Item Profit Ratio',
       'Order Item Quantity', 'Sales', 'Profit Per Order', 'Order Region',
       'Order State', 'Order Status', 'Product Name', 'Product Price',
       'Shipping Mode', 'Customer Name'],
      dtype='object')
```

[67]:
```python
numerical_features = [f for f in df.columns if df[f].dtypes!='O']
```

[68]:
```python
df2[numerical_features].head(2)
```

[68]:
```
       Days for shipping (real)  Sales per customer  Late_delivery_risk  \
33833                         2          239.979996                   0
77011                         3          193.990005                   0

        Latitude  Longitude          Order date  Order Id  \
33833  35.776661 -81.362625 2015-01-01 00:00:00         1
77011  41.832722 -87.980484 2015-01-01 00:21:00         2

       Order Item Discount  Order Item Discount Rate  Order Item Profit Ratio  \
33833                 60.0                      0.20                     0.37
77011                  6.0                      0.03                     0.47

       Order Item Quantity       Sales  Profit Per Order  Product Price
33833                    1  299.980011         88.790001     299.980011
77011                    1  199.990005         91.180000     199.990005
```

[69]:
```python
df2[numerical_features].corr()
```

[69]:
```
                          Days for shipping (real)  Sales per customer  \
Days for shipping (real)                  1.000000            0.001757
Sales per customer                        0.001757            1.000000
Late_delivery_risk                        0.401415           -0.003791
Latitude                                 -0.004073           -0.000223
Longitude                                 0.003911            0.001444
Order date                               -0.001711            0.079000
Order Id                                 -0.001711            0.079000
Order Item Discount                       0.002231            0.498734
Order Item Discount Rate                  0.001467           -0.119469
Order Item Profit Ratio                  -0.004638           -0.001439
Order Item Quantity                      -0.000811            0.105413
Sales                                     0.001962            0.989744
Profit Per Order                         -0.005101            0.133484
Product Price                             0.002185            0.781781

                          Late_delivery_risk  Latitude  Longitude  Order date  \
Days for shipping (real)            0.401415 -0.004073   0.003911   -0.001711
Sales per customer                 -0.003791 -0.000223   0.001444    0.079000
```

```
Late_delivery_risk               1.000000  0.000679  -0.001915   -0.001293
Latitude                         0.000679  1.000000  -0.525122   -0.002984
Longitude                       -0.001915 -0.525122   1.000000    0.002540
Order date                      -0.001293 -0.002984   0.002540    1.000000
Order Id                        -0.001293 -0.002984   0.002540    1.000000
Order Item Discount             -0.000750 -0.002997   0.002343    0.049385
Order Item Discount Rate         0.000404 -0.003889   0.000526    0.000484
Order Item Profit Ratio         -0.002316 -0.000081  -0.003582    0.002760
Order Item Quantity             -0.000139 -0.001853   0.004467   -0.087073
Sales                           -0.003564 -0.000696   0.001696    0.079835
Profit Per Order                -0.003727  0.000338  -0.002521    0.013716
Product Price                   -0.002175  0.000471  -0.000894    0.115324

                          Order Id  Order Item Discount  \
Days for shipping (real) -0.001711             0.002231
Sales per customer        0.079000             0.498734
Late_delivery_risk       -0.001293            -0.000750
Latitude                 -0.002984            -0.002997
Longitude                 0.002540             0.002343
Order date                1.000000             0.049385
Order Id                  1.000000             0.049385
Order Item Discount       0.049385             1.000000
Order Item Discount Rate  0.000484             0.659955
Order Item Profit Ratio   0.002760            -0.002788
Order Item Quantity      -0.087073             0.065379
Sales                     0.079835             0.617438
Profit Per Order          0.013716             0.064756
Product Price             0.115324             0.488101

                          Order Item Discount Rate  Order Item Profit Ratio  \
Days for shipping (real)                  0.001467                -0.004638
Sales per customer                       -0.119469                -0.001439
Late_delivery_risk                        0.000404                -0.002316
Latitude                                 -0.003889                -0.000081
Longitude                                 0.000526                -0.003582
Order date                                0.000484                 0.002760
Order Id                                  0.000484                 0.002760
Order Item Discount                       0.659955                -0.002788
Order Item Discount Rate                  1.000000                -0.002691
Order Item Profit Ratio                  -0.002691                 1.000000
Order Item Quantity                      -0.000028                 0.001128
Sales                                     0.000346                -0.001766
Profit Per Order                         -0.018644                 0.823689
Product Price                             0.000345                -0.002043

                          Order Item Quantity     Sales  Profit Per Order  \
Days for shipping (real)            -0.000811  0.001962         -0.005101
```

```
Sales per customer                    0.105413  0.989744           0.133484
Late_delivery_risk                   -0.000139 -0.003564          -0.003727
Latitude                             -0.001853 -0.000696           0.000338
Longitude                             0.004467  0.001696          -0.002521
Order date                           -0.087073  0.079835           0.013716
Order Id                             -0.087073  0.079835           0.013716
Order Item Discount                   0.065379  0.617438           0.064756
Order Item Discount Rate             -0.000028  0.000346          -0.018644
Order Item Profit Ratio               0.001128 -0.001766           0.823689
Order Item Quantity                   1.000000  0.106442           0.015696
Sales                                 0.106442  1.000000           0.131816
Profit Per Order                      0.015696  0.131816           1.000000
Product Price                        -0.476232  0.789948           0.103459

                          Product Price
Days for shipping (real)       0.002185
Sales per customer             0.781781
Late_delivery_risk            -0.002175
Latitude                       0.000471
Longitude                     -0.000894
Order date                     0.115324
Order Id                       0.115324
Order Item Discount            0.488101
Order Item Discount Rate       0.000345
Order Item Profit Ratio       -0.002043
Order Item Quantity           -0.476232
Sales                          0.789948
Profit Per Order               0.103459
Product Price                  1.000000
```

[70]:
```python
df2.drop(['Sales per customer', 'Latitude', 'Longitude', 'Order Item Discount␣
↪Rate', 'Customer Name', 'Order date'], axis=1, inplace=True)
```

Before encoding the categorical features, we have to look the order status which is our target column for fake order and suspicious transaction detection prediction model.

[71]:
```python
df2['Order Status'].value_counts()
```

[71]:
```
Order Status
COMPLETE           59491
PENDING_PAYMENT    39832
PROCESSING         21902
PENDING            20227
CLOSED             19616
ON_HOLD             9804
SUSPECTED_FRAUD     4062
CANCELED            3692
PAYMENT_REVIEW      1893
```

```
Name: count, dtype: int64
```

Now, Making Order Status column contain only two values that is *suspected_fraud* or *no_suspected_fraud*. * suspected_fraud = 1 * no_suspected_fraud = 0

[72]:
```
df2['Order Status']= [1 if i =='SUSPECTED_FRAUD' else 0 for i in df2['Order␣
↪Status']]
df2['Order Status'].value_counts()
```

[72]:
```
Order Status
0    176457
1      4062
Name: count, dtype: int64
```

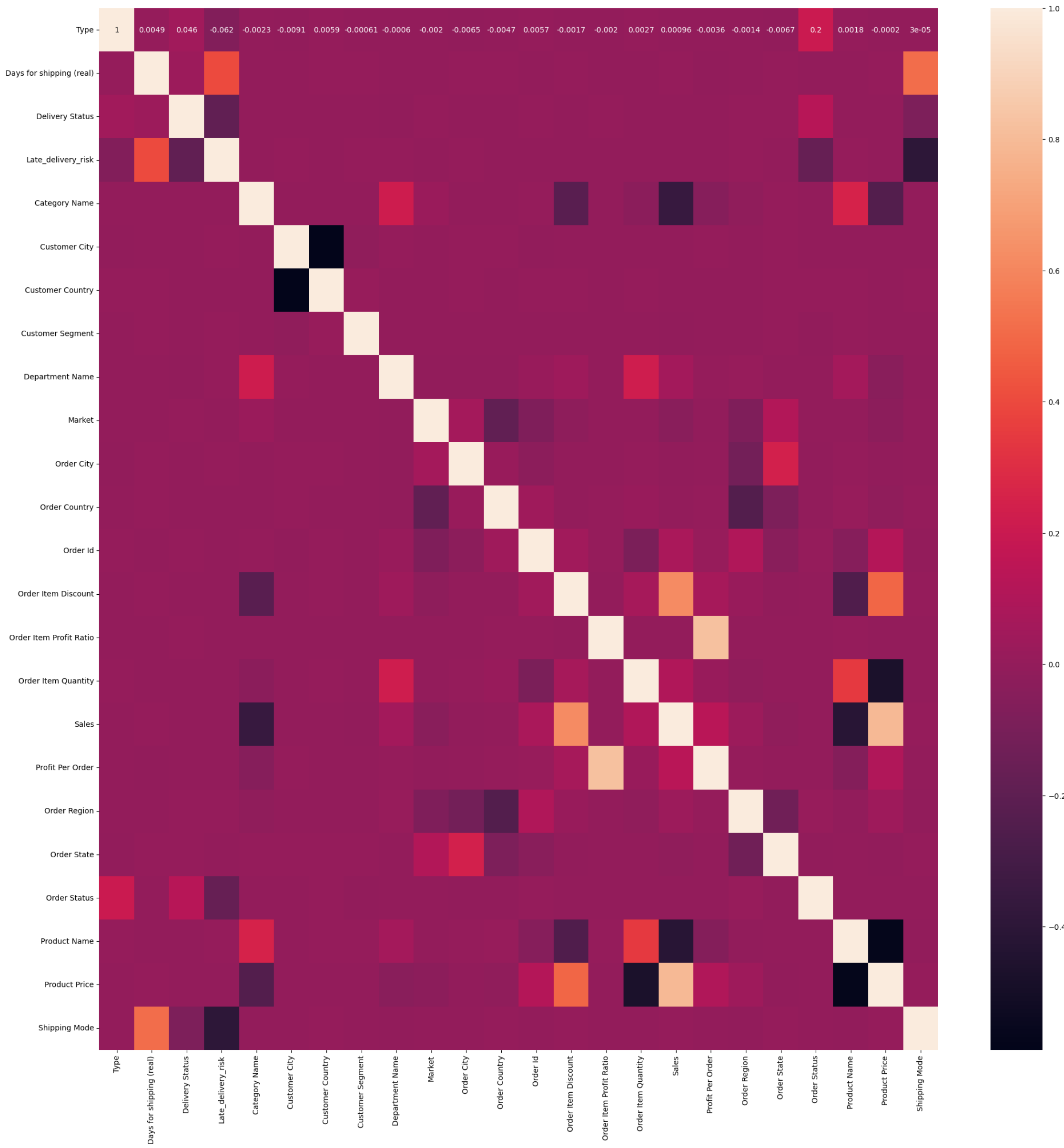
Now looking at the categorical features from where columns can be dropped for model building.

[73]:
```
cat_features = [c for c in df2.columns if df2[c].dtypes=='O']
```

First of all, lets encode the categorical features i.e. converting the categorical features into numerical features because machine learning model only accepts the numerical values.

[74]:
```
le = LabelEncoder()
for features in cat_features:
    df2[features] = le.fit_transform(df2[features])
```

[75]:
```
plt.figure(figsize = (25,25))
sns.heatmap(df2.corr(), annot=True)
plt.show()
```



Now, separating the features and target to train the model.

[76]:
```
X_nc = df2.drop(['Order Status', 'Sales'], axis=1)
y = df2['Order Status']
```

Let's scale the features with standard scalar.

[77]:
```
ss = StandardScaler()
X = ss.fit_transform(X_nc)
```

Separating the features and target in to train and test set.

[78]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=0)
```

4.0.1 Model building

[79]:
```
fod_model = RandomForestClassifier()
```

[80]:
```
fod_model.fit(X_train, y_train)
```

[80]:
```
RandomForestClassifier()
```

[81]:
```
print('The accuracy of Random Forest Classifier: ', fod_model.score(X_test,␣
↪y_test))
y_pred = fod_model.predict(X_test)
y_pred_proba = fod_model.predict_proba(X_test)[:, 1]
fake_order_predict = pd.DataFrame({'actual' : y_test,
                                   'predicted' : y_pred})
fake_order_predict.head()
```

```
The accuracy of Random Forest Classifier:  0.9916906713937513
```

[81]:

	actual	predicted
148115	0	0
133492	0	0
90826	0	0
141991	0	0
69563	0	0

[82]:
```
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)

print(f"Accuracy: {accuracy}" )
print(f"Precision: {precision}")
print(f"Recall: {recall}")
print(f"F1 Score: {f1}")
```

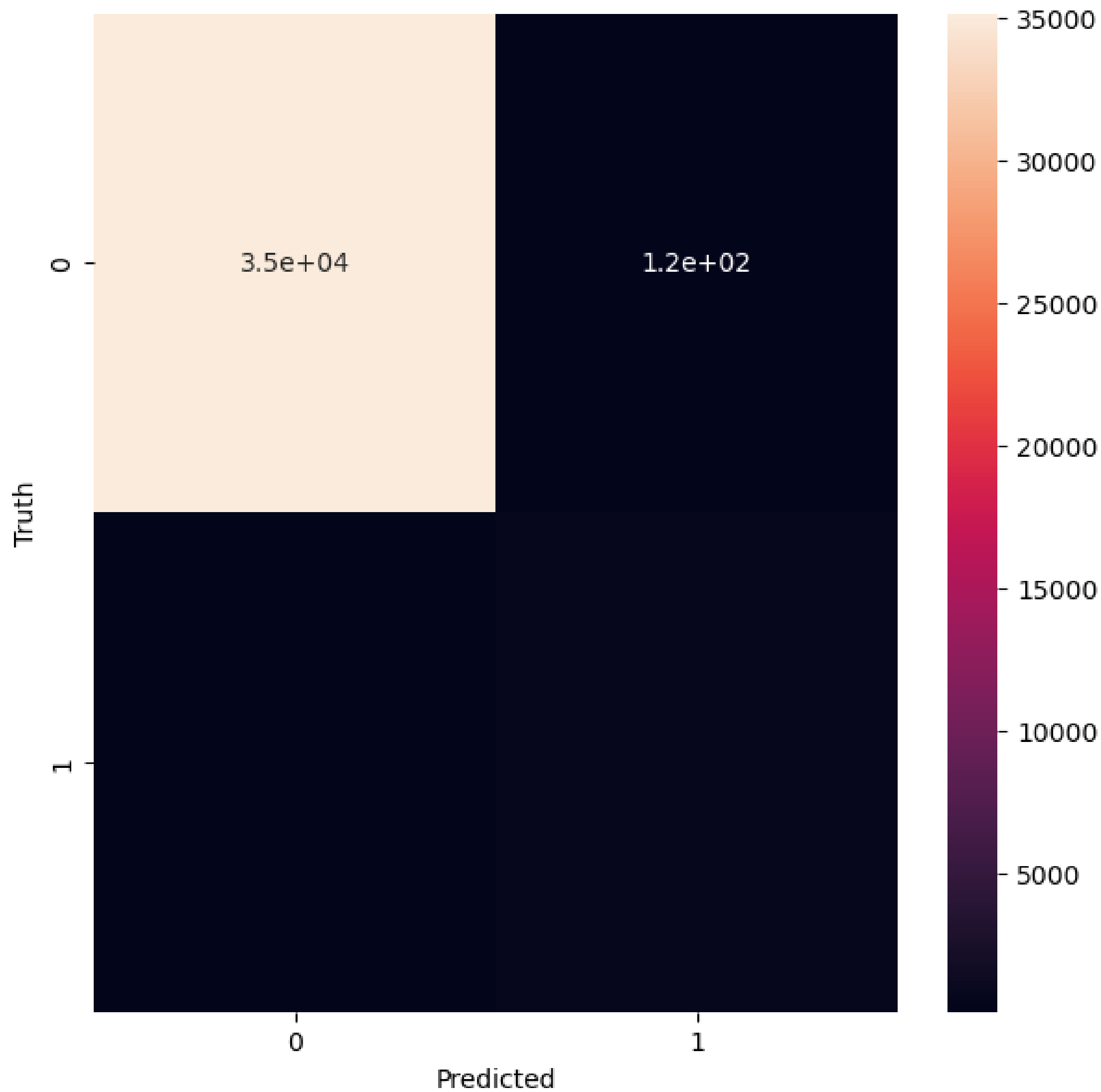
```
Accuracy: 0.9916906713937513
Precision: 0.8435897435897436
Recall: 0.7870813397129187
F1 Score: 0.8143564356435644
```

[83]:
```
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[35146   122]
 [  178   658]]
              precision    recall  f1-score   support
```

```
              0       0.99      1.00      1.00     35268
              1       0.84      0.79      0.81       836

       accuracy                           0.99     36104
      macro avg       0.92      0.89      0.91     36104
   weighted avg       0.99      0.99      0.99     36104
```

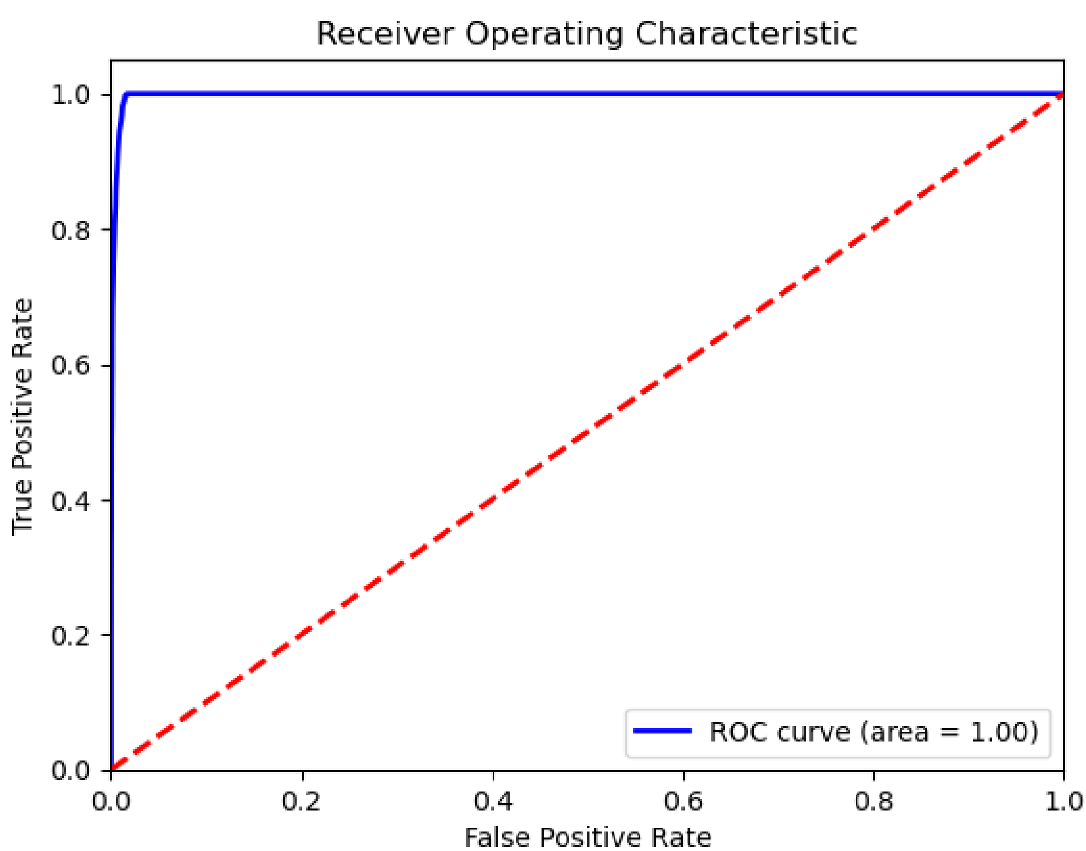
[84]:
```python
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize = (7, 7))
sns.heatmap(cm, annot=True)
plt.xlabel('Predicted')
plt.ylabel('Truth')
plt.show()
```



[85]:

```
fpr, tpr, _ = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

print(f"ROC AUC: {roc_auc}")

# Plot ROC Curve
plt.figure()
plt.plot(fpr, tpr, color='blue', lw=2, label='ROC curve (area = %0.2f)' %␣
↪roc_auc)
plt.plot([0, 1], [0, 1], color='red', lw=2, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic')
plt.legend(loc="lower right")
plt.show()
```

```
ROC AUC: 0.9978674230892584
```



b.

5 5. Forecasting The Sales of Different Products.

For Sales forecasting predictive model, we can use X as above and for target we can take values as below:

[86]:
```
y = df['Sales']
y
```

[86]:
```
33833     299.980011
77011     199.990005
109322    250.000000
87884     129.990005
114915    199.919998
             …
160537    215.820007
93905     215.820007
0         327.750000
52147      11.540000
17863      39.750000
Name: Sales, Length: 180519, dtype: float64
```

[87]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
 ↪random_state=0)
```

Training the model.

[88]:
```
forecast_model = RandomForestRegressor()
```

[89]:
```
forecast_model.fit(X_train, y_train)
```

[89]:
```
RandomForestRegressor()
```

[90]:
```
print('The accuracy of Random Forest Regressor: ', forecast_model.score(X_test,␣
 ↪y_test))
y_pred = forecast_model.predict(X_test)
forecast_predict = pd.DataFrame({'actual' : y_test,
                                  'predicted' : y_pred})
```

```
The accuracy of Random Forest Regressor:  0.9999990493175964
```

[91]:
```
forecast_predict.head()
```

[91]:

	actual	predicted
148115	179.970001	179.970001
133492	149.940002	149.940002

```
90826    99.989998   99.989998
141991  129.990005  129.990005
69563   119.980003  119.980003
```

[92]:
```
forecast_predict.tail()
```

[92]:
```
            actual   predicted
145993  399.980011  399.980011
86750   129.990005  129.990005
9043     79.980003   79.980003
60222    59.990002   59.990002
15861    59.990002   59.990002
```

Let's predict for random value.

[93]:
```
df2[99:100]
```

[93]:
```
       Type  Days for shipping (real)  Delivery Status  Late_delivery_risk  \
18107     0                         3                1                   1

       Category Name  Customer City  Customer Country  Customer Segment  \
18107             12            109                 0                 1

       Department Name  Market  …  Order Item Profit Ratio  \
18107                0       2  …                     0.28

       Order Item Quantity      Sales  Profit Per Order  Order Region  \
18107                    1  59.990002             13.94             3

       Order State  Order Status  Product Name  Product Price  Shipping Mode
18107          292             0            71      59.990002              2

[1 rows x 24 columns]
```

[94]:
```
df2[99:100]['Sales']
```

[94]:
```
18107    59.990002
Name: Sales, dtype: float64
```

[95]:
```
df3 = pd.DataFrame(X, columns=X_nc.columns)
```

[96]:
```
forecast_model.predict(df3[99:100])
```

[96]:
```
array([59.99000168])
```

For another one.

[97]:
```
df2[77487:77488]['Sales']
```

[97]: 80185   199.990005
Name: Sales, dtype: float64

[98]:
```
forecast_model.predict(df3[77487:77488])
```

[98]: array([199.9900055])

[ ]:
```
```