In [13]:
```
import pandas as pd
import numpy as np
import scipy as sp
import matplotlib as mpl
import seaborn as sns
```

In [14]:
```
SGSC = (r'C:\Users\tyler\OneDrive\Desktop\Tyler stuff\Predictive Modeling\Supply Chain Dataset.csv')
```

In [15]:
```
df = pd.read_csv(SGSC, encoding= 'unicode_escape')
```

In [16]:
```
df.head()
```

Out[16]:

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Cate
0	DEBIT	3	4	91.250000	314.640015	Advance shipping	0	
1	TRANSFER	5	4	-249.089996	311.359985	Late delivery	1	
2	CASH	4	4	-247.779999	309.720001	Shipping on time	0	
3	DEBIT	3	4	22.860001	304.809998	Advance shipping	0	
4	PAYMENT	2	4	134.210007	298.250000	Advance shipping	0	

5 rows × 53 columns

In [17]:
```
df.columns
```

Out[17]:
```
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 Statu
s',
       'Order Zipcode', 'Product Card Id', 'Product Category Id',
       'Product Description', 'Product Image', 'Product Name', 'Product Pric
e',
       'Product Status', 'shipping date (DateOrders)', 'Shipping Mode'],
      dtype='object')
```

In [18]:
```
df.shape
```

Out[18]:
```
(180519, 53)
```

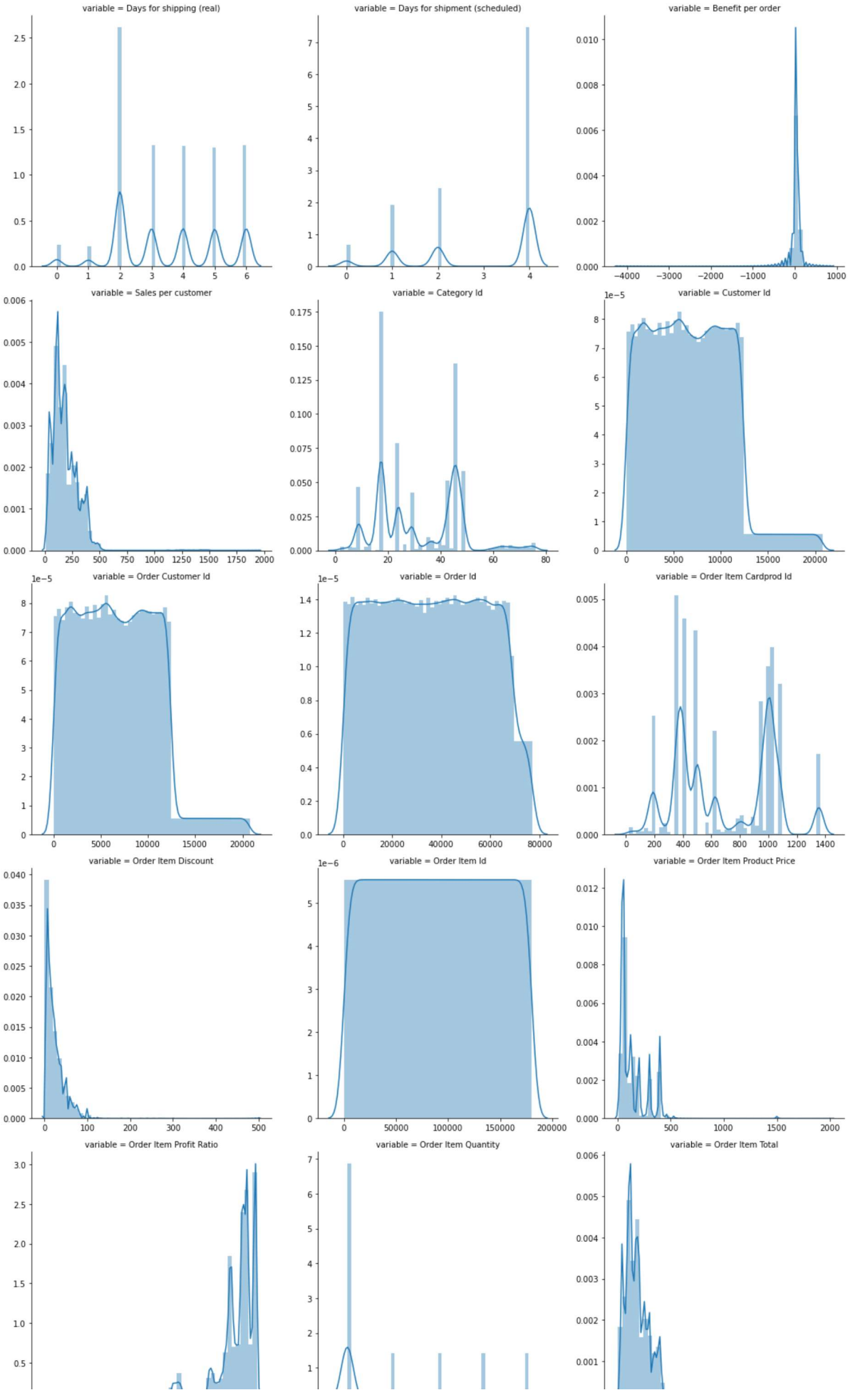
In [19]:
```
df.isnull().sum()
```

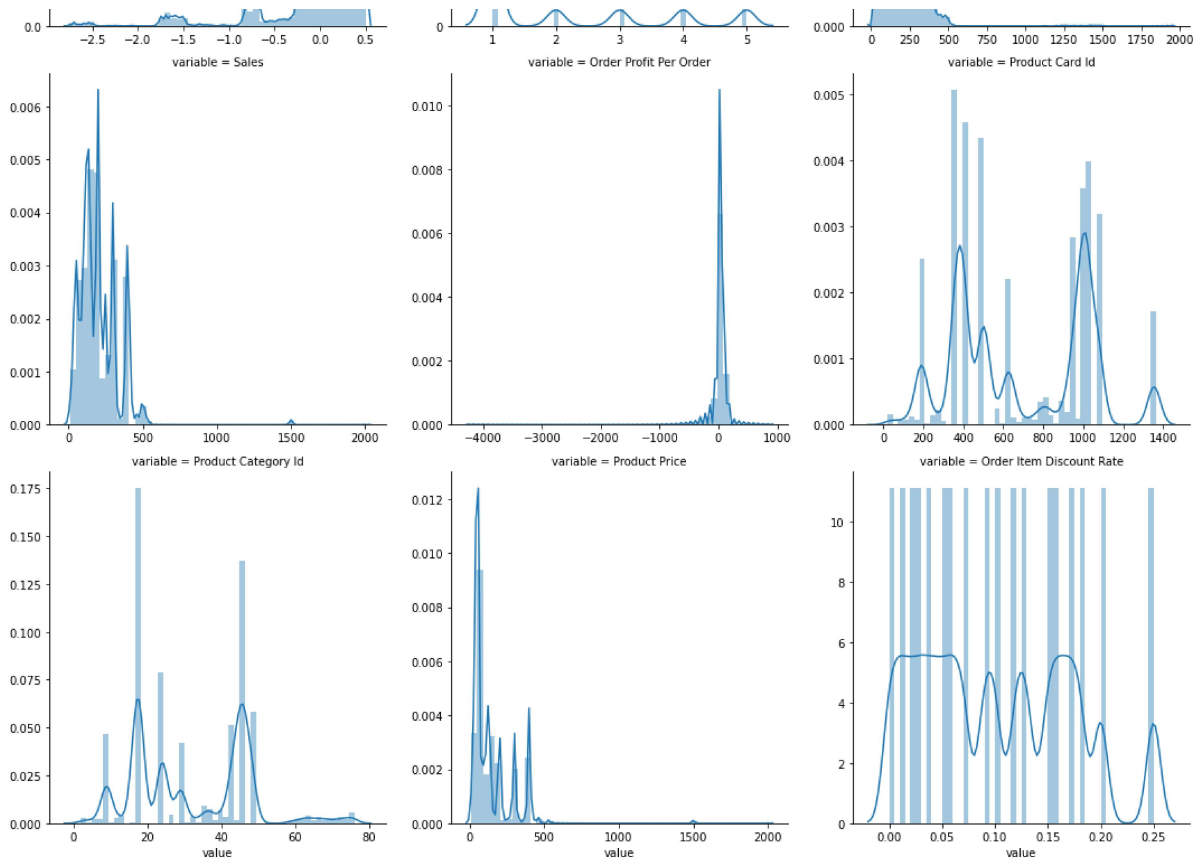
Out[19]:
```
Type                              0
Days for shipping (real)          0
Days for shipment (scheduled)     0
Benefit per order                 0
Sales per customer                0
Delivery Status                   0
Late_delivery_risk                0
Category Id                       0
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
```

In [20]:
```
numeric_cols = ['Days for shipping (real)','Days for shipment (scheduled)','Benefit per order','Sales per customer','Category Id','Customer Id',
                'Order Customer Id','Order Id','Order Item Cardprod Id','Order Item Discount','Order Item Id','Order Item Product Price','Order Item Profit Ratio',
                'Order Item Quantity','Order Item Total','Sales','Order Item Total','Order Profit Per Order','Product Card Id','Product Category Id',
                'Product Price','Order Item Discount Rate']
```

In [21]:
```
categorical_cols = ['Type','Delivery Status','Customer Segment','Market','Department Id','Order Status','Shipping Mode']
```

In [22]:
```
f = pd.melt(df, value_vars=numeric_cols)
g = sns.FacetGrid(f, col="variable",  col_wrap=3, sharex=False, sharey=False,
height = 5)
g = g.map(sns.distplot, "value")
```

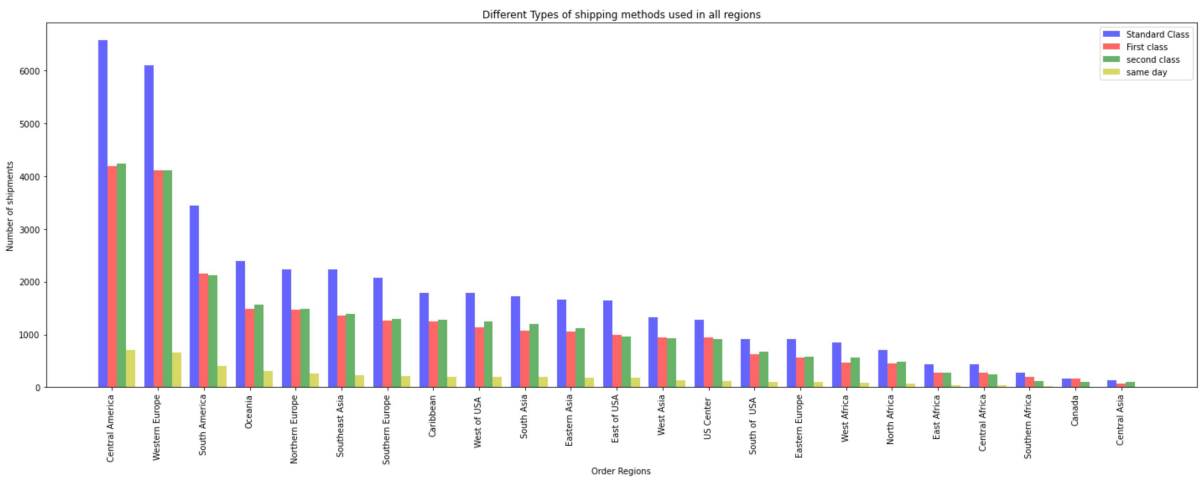

variable = Days for shipping (real)
variable = Days for shipment (scheduled)
variable = Benefit per order
variable = Sales per customer
variable = Category Id
variable = Customer Id
1e−5
variable = Order Customer Id
1e−5
variable = Order Id
1e−5
variable = Order Item Cardprod Id
variable = Order Item Discount
variable = Order Item Id
1e−6
variable = Order Item Product Price
variable = Order Item Profit Ratio
variable = Order Item Quantity
variable = Order Item Total



In [23]:
```
count=df['Delivery Status'].value_counts()  #change categoric variable
print(count / len(df))
```

```
Late delivery        0.548291
Advance shipping     0.230402
Shipping on time     0.178352
Shipping canceled    0.042954
Name: Delivery Status, dtype: float64
```

In [24]:
```
%matplotlib inline
from matplotlib import pyplot as plt
#Filtering late delivery orders with standard class shipping
xyz1 = df[(df['Delivery Status'] == 'Late delivery') & (df['Shipping Mode'] ==
'Standard Class')]
#Filtering late delivery orders with first class shipping
xyz2 = df[(df['Delivery Status'] == 'Late delivery') & (df['Shipping Mode'] ==
'First Class')]
#Filtering late delivery orders with second class shipping
xyz3 = df[(df['Delivery Status'] == 'Late delivery') & (df['Shipping Mode'] ==
'Second Class')]
#Filtering late delivery orders with same day shipping
xyz4 = df[(df['Delivery Status'] == 'Late delivery') & (df['Shipping Mode'] ==
'Same Day')]
#Counting total values
count1=xyz1['Order Region'].value_counts()
count2=xyz2['Order Region'].value_counts()
count3=xyz3['Order Region'].value_counts()
count4=xyz4['Order Region'].value_counts()
#Index names
names=df['Order Region'].value_counts().keys()
n_groups=23
fig,ax = plt.subplots(figsize=(20,8))
index=np.arange(n_groups)
bar_width=0.2
opacity=0.6
type1=plt.bar(index,count1,bar_width,alpha=opacity,color='b',label='Standard C
lass')
type2=plt.bar(index+bar_width,count2,bar_width,alpha=opacity,color='r',label=
'First class')
type3=plt.bar(index+bar_width+bar_width,count3,bar_width,alpha=opacity,color=
'g',label='second class')
type4=plt.bar(index+bar_width+bar_width+bar_width,count4,bar_width,alpha=opaci
ty,color='y',label='same day')
plt.xlabel('Order Regions')
plt.ylabel('Number of shipments')
plt.title('Different Types of shipping methods used in all regions')
plt.legend()
plt.xticks(index+bar_width,names,rotation=90)
plt.tight_layout()
plt.show()
```



In [25]:
```
fig, ax = plt.subplots(figsize=(24,12))          # figsize
sns.heatmap(df.corr(),annot=True,linewidths=.5,fmt='.1g',cmap= sns.diverging_palette(230, 20, as_cmap=True)) # Heatmap for correlation matrix
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x1db296e2370>

In [28]:
```
df.drop(columns=['Benefit per order', 'Sales per customer', 'Category Id', 'Ca
tegory 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', 'Latitude', 'Longitude',
       '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', 'Sales', 'Order
 State',
       'Order Zipcode', 'Product Card Id', 'Product Category Id',
       'Product Description', 'Product Image', 'Product Name',
       'Product Status', 'shipping date (DateOrders)', 'Order Item Total', 'Ma
rket'])
```

Out[28]:

	Type	Days for shipping (real)	Days for shipment (scheduled)	Delivery Status	Late_delivery_risk	Department Name	Order Item Quantity	O
0	DEBIT	3	4	Advance shipping	0	Fitness	1	
1	TRANSFER	5	4	Late delivery	1	Fitness	1	-2
2	CASH	4	4	Shipping on time	0	Fitness	1	-2
3	DEBIT	3	4	Advance shipping	0	Fitness	1	
4	PAYMENT	2	4	Advance shipping	0	Fitness	1	1
...	...	...	...	...	...	...	...	
180514	CASH	4	4	Shipping on time	0	Fan Shop	1	
180515	DEBIT	3	2	Late delivery	1	Fan Shop	1	-6
180516	TRANSFER	5	4	Late delivery	1	Fan Shop	1	1
180517	PAYMENT	3	4	Advance shipping	0	Fan Shop	1	1
180518	PAYMENT	4	4	Shipping on time	0	Fan Shop	1	1

180519 rows × 12 columns

In [26]:
```
df.dtypes
```

Out[26]:
```
Type                           object
Days for shipping (real)        int64
Days for shipment (scheduled)   int64
Benefit per order             float64
Sales per customer            float64
Delivery Status                object
Late_delivery_risk              int64
Category Id                     int64
Category Name                  object
Customer City                  object
Customer Country               object
Customer Email                 object
Customer Fname                 object
Customer Id                     int64
Customer Lname                 object
Customer Password              object
Customer Segment               object
Customer State                 object
Customer Street                object
Customer Zipcode              float64
Department Id                   int64
Department Name                object
Latitude                      float64
Longitude                     float64
Market                         object
Order City                     object
Order Country                  object
Order Customer Id               int64
order date (DateOrders)        object
Order Id                        int64
Order Item Cardprod Id          int64
Order Item Discount           float64
Order Item Discount Rate      float64
Order Item Id                   int64
Order Item Product Price      float64
Order Item Profit Ratio       float64
Order Item Quantity             int64
Sales                         float64
Order Item Total              float64
Order Profit Per Order        float64
Order Region                   object
Order State                    object
Order Status                   object
Order Zipcode                 float64
Product Card Id                 int64
Product Category Id             int64
Product Description           float64
Product Image                  object
Product Name                   object
Product Price                 float64
Product Status                  int64
shipping date (DateOrders)     object
Shipping Mode                  object
dtype: object
```

In [27]:
```
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
```

In [28]:
```
df.describe()
```

Out[28]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Categ
count	180519.000000	180519.000000	180519.000000	180519.000000	180519.000000	180519.0(
mean	3.497654	2.931847	21.974989	183.107609	0.548291	31.8
std	1.623722	1.374449	104.433526	120.043670	0.497664	15.6
min	0.000000	0.000000	-4274.979980	7.490000	0.000000	2.0(
25%	2.000000	2.000000	7.000000	104.379997	0.000000	18.0(
50%	3.000000	4.000000	31.520000	163.990005	1.000000	29.0(
75%	5.000000	4.000000	64.800003	247.399994	1.000000	45.0(
max	6.000000	4.000000	911.799988	1939.989990	1.000000	76.0(

8 rows × 29 columns

In [29]:
```
CorrSGSC = df[['Days for shipping (real)', 'Days for shipment (scheduled)',
       'Benefit per order', 'Sales per customer',
       'Late_delivery_risk', 'Order Item Product Price',
       'Order Item Quantity', 'Sales', 'Order Item Total', 'Order Profit Per O
rder',
       'Product Price', 'Product Status']].corr()
```

In [30]:
```
print(CorrSGSC)
```

	Days for shipping (real)
Days for shipping (real)	1.000000
Days for shipment (scheduled)	0.515880
Benefit per order	-0.005101
Sales per customer	0.001757
Late_delivery_risk	0.401415
Order Item Product Price	0.002185
Order Item Quantity	-0.000811
Sales	0.001962
Order Item Total	0.001757
Order Profit Per Order	-0.005101
Product Price	0.002185
Product Status	NaN

	Days for shipment (scheduled)
Days for shipping (real)	0.515880
Days for shipment (scheduled)	1.000000
Benefit per order	-0.000185
Sales per customer	0.006445
Late_delivery_risk	-0.369352
Order Item Product Price	0.006912
Order Item Quantity	-0.002925
Sales	0.006327
Order Item Total	0.006445
Order Profit Per Order	-0.000185
Product Price	0.006912
Product Status	NaN

	Benefit per order	Sales per customer
Days for shipping (real)	-0.005101	0.001757
Days for shipment (scheduled)	-0.000185	0.006445
Benefit per order	1.000000	0.133484
Sales per customer	0.133484	1.000000
Late_delivery_risk	-0.003727	-0.003791
Order Item Product Price	0.103459	0.781781
Order Item Quantity	0.015696	0.105413
Sales	0.131816	0.989744
Order Item Total	0.133484	1.000000
Order Profit Per Order	1.000000	0.133484
Product Price	0.103459	0.781781
Product Status	NaN	NaN

	Late_delivery_risk	Order Item Product Price
Days for shipping (real)	0.401415	0.002185
Days for shipment (scheduled)	-0.369352	0.006912
Benefit per order	-0.003727	0.103459
Sales per customer	-0.003791	0.781781
Late_delivery_risk	1.000000	-0.002175
Order Item Product Price	-0.002175	1.000000
Order Item Quantity	-0.000139	-0.476232
Sales	-0.003564	0.789948
Order Item Total	-0.003791	0.781781
Order Profit Per Order	-0.003727	0.103459
Product Price	-0.002175	1.000000
Product Status	NaN	NaN

	Order Item Quantity	Sales	\
Days for shipping (real)	-0.000811	0.001962	
Days for shipment (scheduled)	-0.002925	0.006327	
Benefit per order	0.015696	0.131816	
Sales per customer	0.105413	0.989744	
Late_delivery_risk	-0.000139	-0.003564	
Order Item Product Price	-0.476232	0.789948	
Order Item Quantity	1.000000	0.106442	
Sales	0.106442	1.000000	
Order Item Total	0.105413	0.989744	
Order Profit Per Order	0.015696	0.131816	
Product Price	-0.476232	0.789948	
Product Status	NaN	NaN	

	Order Item Total	Order Profit Per Order	\
Days for shipping (real)	0.001757	-0.005101	
Days for shipment (scheduled)	0.006445	-0.000185	
Benefit per order	0.133484	1.000000	
Sales per customer	1.000000	0.133484	
Late_delivery_risk	-0.003791	-0.003727	
Order Item Product Price	0.781781	0.103459	
Order Item Quantity	0.105413	0.015696	
Sales	0.989744	0.131816	
Order Item Total	1.000000	0.133484	
Order Profit Per Order	0.133484	1.000000	
Product Price	0.781781	0.103459	
Product Status	NaN	NaN	

	Product Price	Product Status
Days for shipping (real)	0.002185	NaN
Days for shipment (scheduled)	0.006912	NaN
Benefit per order	0.103459	NaN
Sales per customer	0.781781	NaN
Late_delivery_risk	-0.002175	NaN
Order Item Product Price	1.000000	NaN
Order Item Quantity	-0.476232	NaN
Sales	0.789948	NaN
Order Item Total	0.781781	NaN
Order Profit Per Order	0.103459	NaN
Product Price	1.000000	NaN
Product Status	NaN	NaN

In [65]:
```
CorrSGSC.describe()
```

Out[65]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Order Item Product Price	Order Item Quantity	
count	11.000000	11.000000	11.000000	11.000000	11.000000	11.000000	11.000000	11.
mean	0.174193	0.106934	0.237489	0.448191	0.091725	0.371920	0.035666	0.
std	0.330690	0.357351	0.381352	0.451568	0.347143	0.507255	0.383882	0.
min	-0.005101	-0.369352	-0.005101	-0.003791	-0.369352	-0.476232	-0.476232	-0.
25%	0.000473	-0.000185	0.007756	0.055929	-0.003759	0.004549	-0.001868	0.
50%	0.001962	0.006445	0.103459	0.133484	-0.003564	0.103459	0.015696	0.
75%	0.201800	0.006912	0.133484	0.885763	-0.001157	0.785865	0.105413	0.
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.

In [66]:
```
CorrSGSC.corr()['Sales'].sort_values(ascending=False)
```

Out[66]:
```
Sales                           1.000000
Order Item Total                0.999889
Sales per customer              0.999889
Product Price                   0.898637
Order Item Product Price        0.898637
Order Profit Per Order         -0.201463
Benefit per order              -0.201463
Order Item Quantity            -0.289933
Late_delivery_risk             -0.301798
Days for shipment (scheduled)  -0.303170
Days for shipping (real)       -0.567907
Product Status                       NaN
Name: Sales, dtype: float64
```

In [8]:
```
CorrSGSC.corr()['Late_delivery_risk'].sort_values(ascending=False)
```

Out[8]:
```
Late_delivery_risk              1.000000
Days for shipping (real)        0.380272
Order Item Quantity            -0.026064
Order Profit Per Order         -0.192407
Benefit per order              -0.192407
Product Price                  -0.223128
Order Item Product Price       -0.223128
Sales                          -0.301798
Order Item Total               -0.301865
Sales per customer             -0.301865
Days for shipment (scheduled)  -0.515602
Product Status                       NaN
Name: Late_delivery_risk, dtype: float64
```

In [9]:
```
CorrSGSC.corr()['Days for shipping (real)'].sort_values(ascending=False)
```

Out[9]:
```
Days for shipping (real)         1.000000
Days for shipment (scheduled)    0.574302
Late_delivery_risk               0.380272
Order Item Quantity             -0.057653
Order Profit Per Order          -0.373318
Benefit per order               -0.373318
Product Price                   -0.417164
Order Item Product Price        -0.417164
Order Item Total                -0.567742
Sales per customer              -0.567742
Sales                           -0.567907
Product Status                        NaN
Name: Days for shipping (real), dtype: float64
```

In [10]:
```
CorrSGSC.corr()['Order Item Quantity'].sort_values(ascending=False)
```

Out[10]:
```
Order Item Quantity              1.000000
Late_delivery_risk              -0.026064
Days for shipment (scheduled)   -0.038435
Days for shipping (real)        -0.057653
Order Profit Per Order          -0.069980
Benefit per order               -0.069980
Order Item Total                -0.285375
Sales per customer              -0.285375
Sales                           -0.289933
Product Price                   -0.679148
Order Item Product Price        -0.679148
Product Status                        NaN
Name: Order Item Quantity, dtype: float64
```

In [67]:
```
CorrSGSC.corr().style.background_gradient(cmap='coolwarm')
```

```
C:\Users\tyler\anaconda3\lib\site-packages\pandas\io\formats\style.py:1089: R
untimeWarning: All-NaN slice encountered
  smin = np.nanmin(s.to_numpy()) if vmin is None else vmin
C:\Users\tyler\anaconda3\lib\site-packages\pandas\io\formats\style.py:1090: R
untimeWarning: All-NaN slice encountered
  smax = np.nanmax(s.to_numpy()) if vmax is None else vmax
```

Out[67]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Order Item Product Price	Q
Days for shipping (real)	1.000000	0.574302	-0.373318	-0.567742	0.380272	-0.417164	-0.(
Days for shipment (scheduled)	0.574302	1.000000	-0.203428	-0.302936	-0.515602	-0.219776	-0.(
Benefit per order	-0.373318	-0.203428	1.000000	-0.199193	-0.192407	-0.130480	-0.(
Sales per customer	-0.567742	-0.302936	-0.199193	1.000000	-0.301865	0.896204	-0.:
Late_delivery_risk	0.380272	-0.515602	-0.192407	-0.301865	1.000000	-0.223128	-0.(
Order Item Product Price	-0.417164	-0.219776	-0.130480	0.896204	-0.223128	1.000000	-0.(
Order Item Quantity	-0.057653	-0.038435	-0.069980	-0.285375	-0.026064	-0.679148	1.(
Sales	-0.567907	-0.303170	-0.201463	0.999889	-0.301798	0.898637	-0.:
Order Item Total	-0.567742	-0.302936	-0.199193	1.000000	-0.301865	0.896204	-0.:
Order Profit Per Order	-0.373318	-0.203428	1.000000	-0.199193	-0.192407	-0.130480	-0.(
Product Price	-0.417164	-0.219776	-0.130480	0.896204	-0.223128	1.000000	-0.(
Product Status	nan	nan	nan	nan	nan	nan	

In [76]:
```
import statsmodels.formula.api as smf
model1=smf.ols(formula='Sales~Late_delivery_risk',data=CorrSGSC).fit()
model1.params
```

Out[76]:
```
Intercept            0.484580
Late_delivery_risk   -0.392672
dtype: float64
```

In [77]:
```
model1.summary()
```

```
C:\Users\tyler\anaconda3\lib\site-packages\scipy\stats\stats.py:1603: UserWar
ning: kurtosistest only valid for n>=20 ... continuing anyway, n=11
  warnings.warn("kurtosistest only valid for n>=20 ... continuing "
```

Out[77]:

OLS Regression Results

Dep. Variable:	Sales	**R-squared:**	0.091
Model:	OLS	**Adj. R-squared:**	-0.010
Method:	Least Squares	**F-statistic:**	0.9019
Date:	Tue, 03 Aug 2021	**Prob (F-statistic):**	0.367
Time:	17:38:08	**Log-Likelihood:**	-5.8161
No. Observations:	11	**AIC:**	15.63
Df Residuals:	9	**BIC:**	16.43
Df Model:	1		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
Intercept	0.4846	0.142	3.412	0.008	0.163	0.806
Late_delivery_risk	-0.3927	0.413	-0.950	0.367	-1.328	0.543

Omnibus:	4.771	**Durbin-Watson:**	2.075
Prob(Omnibus):	0.092	**Jarque-Bera (JB):**	1.212
Skew:	0.036	**Prob(JB):**	0.545
Kurtosis:	1.375	**Cond. No.**	3.05

Warnings:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [ ]:
```
LogSGSC = df.drop(['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 Zip
code','Department Id', 'Department Name',
        'Latitude', 'Longitude', 'Order City', 'Order Country', 'Order Custome
r Id', 'Order Id', 'Order Item Cardprod Id',
       'Order Item Discount', 'Order Item Discount Rate', 'Order Item Id', 'Or
der Item Profit Ratio', 'Order State',
       'Order Zipcode', 'Product Card Id', 'Product Category Id', 'Product Des
cription', 'Product Image', 'Product Name',
        'shipping date (DateOrders)', 'Order Item Product Price', 'Order Item
 Quantity', 'Order Profit Per Order',
       'Product Price', 'Product Status', 'order date (DateOrders)'], axis=1,
inplace=True)
```

In [90]:
```
df.head()
```

Out[90]:

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Mark
0	DEBIT	3	4	91.250000	314.640015	Advance shipping	0	Pacif As
1	TRANSFER	5	4	-249.089996	311.359985	Late delivery	1	Pacif As
2	CASH	4	4	-247.779999	309.720001	Shipping on time	0	Pacif As
3	DEBIT	3	4	22.860001	304.809998	Advance shipping	0	Pacif As
4	PAYMENT	2	4	134.210007	298.250000	Advance shipping	0	Pacif As

In [98]:
```
df.drop(['Order Profit Per Order', 'order date (DateOrders)', 'Order Item Quantity', 'Product Price', 'Product Status'],
        axis=1, inplace=True)
```

In [99]:
```
df.head()
```

Out[99]:

	Type	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Delivery Status	Late_delivery_risk	Mark
0	DEBIT	3	4	91.250000	314.640015	Advance shipping	0	Pacif As
1	TRANSFER	5	4	-249.089996	311.359985	Late delivery	1	Pacif As
2	CASH	4	4	-247.779999	309.720001	Shipping on time	0	Pacif As
3	DEBIT	3	4	22.860001	304.809998	Advance shipping	0	Pacif As
4	PAYMENT	2	4	134.210007	298.250000	Advance shipping	0	Pacif As

In [31]:
```
df_vars=['Type', 'Days for shipping (real)', 'Days for shipment (scheduled)',
       'Benefit per order', 'Sales per customer', 'Delivery Status', 'Late_delivery_risk', 'Market',
       'Order Item Product Price', 'Sales', 'Order Item Total', 'Order Region', 'Order Status', 'Shipping Mode']
```

In [32]:
```
test_data = df.copy()
test_data["Late_delivery_risk"].fillna(df["Late_delivery_risk"].median(skipna=
True), inplace=True)

#test_data['TravelAlone']=np.where((test_data["SibSp"]+test_data["Parch"])>0,
 0, 1)

testing = pd.get_dummies(test_data, columns=["Type", "Delivery Status", "Marke
t", "Order Region",
                                             "Order Status", "Shipping Mode"])

final_test = testing
final_test.head()
```

Out[32]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Category Id	Category Name	Custo
0	3	4	91.250000	314.640015	0	73	Sporting Goods	Cag
1	5	4	-249.089996	311.359985	1	73	Sporting Goods	Cag
2	4	4	-247.779999	309.720001	0	73	Sporting Goods	San
3	3	4	22.860001	304.809998	0	73	Sporting Goods	Ang
4	2	4	134.210007	298.250000	0	73	Sporting Goods	Cag

5 rows × 96 columns

In [29]:
```
final_test.columns.values
```

Out[29]:
```
array(['Days for shipping (real)', 'Days for shipment (scheduled)',
       'Benefit per order', 'Sales per customer', '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', '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 State', 'Order Zipcode',
       'Product Card Id', 'Product Category Id', 'Product Description',
       'Product Image', 'Product Name', 'Product Price', 'Product Status',
       'shipping date (DateOrders)', 'Type_CASH', 'Type_DEBIT',
       'Type_PAYMENT', 'Type_TRANSFER',
       'Delivery Status_Advance shipping',
       'Delivery Status_Late delivery',
       'Delivery Status_Shipping canceled',
       'Delivery Status_Shipping on time', 'Market_Africa',
       'Market_Europe', 'Market_LATAM', 'Market_Pacific Asia',
       'Market_USCA', 'Order Region_Canada', 'Order Region_Caribbean',
       'Order Region_Central Africa', 'Order Region_Central America',
       'Order Region_Central Asia', 'Order Region_East Africa',
       'Order Region_East of USA', 'Order Region_Eastern Asia',
       'Order Region_Eastern Europe', 'Order Region_North Africa',
       'Order Region_Northern Europe', 'Order Region_Oceania',
       'Order Region_South America', 'Order Region_South Asia',
       'Order Region_South of  USA ', 'Order Region_Southeast Asia',
       'Order Region_Southern Africa', 'Order Region_Southern Europe',
       'Order Region_US Center ', 'Order Region_West Africa',
       'Order Region_West Asia', 'Order Region_West of USA ',
       'Order Region_Western Europe', 'Order Status_CANCELED',
       'Order Status_CLOSED', 'Order Status_COMPLETE',
       'Order Status_ON_HOLD', 'Order Status_PAYMENT_REVIEW',
       'Order Status_PENDING', 'Order Status_PENDING_PAYMENT',
       'Order Status_PROCESSING', 'Order Status_SUSPECTED_FRAUD',
       'Shipping Mode_First Class', 'Shipping Mode_Same Day',
       'Shipping Mode_Second Class', 'Shipping Mode_Standard Class'],
      dtype=object)
```

In [16]:
```
final_test.head(10)
```

Out[16]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Category Id	Category Name	Cust
0	3	4	91.250000	314.640015	0	73	Sporting Goods	Ca
1	5	4	-249.089996	311.359985	1	73	Sporting Goods	Ca
2	4	4	-247.779999	309.720001	0	73	Sporting Goods	San
3	3	4	22.860001	304.809998	0	73	Sporting Goods	An
4	2	4	134.210007	298.250000	0	73	Sporting Goods	Ca
5	6	4	18.580000	294.980011	0	73	Sporting Goods	Tonaw
6	2	1	95.180000	288.420013	1	73	Sporting Goods	Ca
7	2	1	68.430000	285.140015	1	73	Sporting Goods	M
8	3	2	133.720001	278.589996	1	73	Sporting Goods	Ca
9	2	1	132.149994	275.309998	1	73	Sporting Goods	Ra

10 rows × 96 columns

In [17]:
```
final_test.drop(['Sales per customer', 'Order Item Product Price', 'Order Item Total', 'Type_CASH', 'Type_DEBIT', 'Type_PAYMENT', 'Type_TRANSFER','Order Region_Canada', 'Order Region_Caribbean', 'Order Region_Central Africa', 'Order Region_Central America', 'Order Region_Central Asia', 'Order Region_East Africa', 'Order Region_East of USA', 'Order Region_Eastern Asia', 'Order Region_Eastern Europe', 'Order Region_North Africa', 'Order Region_Northern Europe', 'Order Region_Oceania', 'Order Region_South America', 'Order Region_South Asia', 'Order Region_South of  USA ', 'Order Region_Southeast Asia', 'Order Region_Southern Africa', 'Order Region_Southern Europe', 'Order Region_US Center ', 'Order Region_West Africa', 'Order Region_West Asia', 'Order Region_West of USA ', 'Order Region_Western Europe'], axis=1, inplace=True)
```

In [18]:
```
final_test.head(10)
```

Out[18]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Late_delivery_risk	Category Id	Category Name	Customer City	Custo Cou
0	3	4	91.250000	0	73	Sporting Goods	Caguas	Pu F
1	5	4	-249.089996	1	73	Sporting Goods	Caguas	Pu F
2	4	4	-247.779999	0	73	Sporting Goods	San Jose	EE.
3	3	4	22.860001	0	73	Sporting Goods	Los Angeles	EE.
4	2	4	134.210007	0	73	Sporting Goods	Caguas	Pu F
5	6	4	18.580000	0	73	Sporting Goods	Tonawanda	EE.
6	2	1	95.180000	1	73	Sporting Goods	Caguas	Pu F
7	2	1	68.430000	1	73	Sporting Goods	Miami	EE.
8	3	2	133.720001	1	73	Sporting Goods	Caguas	Pu F
9	2	1	132.149994	1	73	Sporting Goods	San Ramon	EE.

10 rows × 66 columns

In [22]:
```
final_test.head(10)
```

Out[22]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Late_delivery_risk	Customer Id	Customer Lname	Customer Password	Cu S
0	3	4	91.250000	0	20755	Holloway	XXXXXXXXX	Cc
1	5	4	-249.089996	1	19492	Luna	XXXXXXXXX	Cc
2	4	4	-247.779999	0	19491	Maldonado	XXXXXXXXX	Cc
3	3	4	22.860001	0	19490	Tate	XXXXXXXXX	
4	2	4	134.210007	0	19489	Hendricks	XXXXXXXXX	Cc
5	6	4	18.580000	0	19488	Flowers	XXXXXXXXX	Cc
6	2	1	95.180000	1	19487	Terrell	XXXXXXXXX	
7	2	1	68.430000	1	19486	Stevens	XXXXXXXXX	Cc
8	3	2	133.720001	1	19485	Olsen	XXXXXXXXX	Cc
9	2	1	132.149994	1	19484	Delacruz	XXXXXXXXX	Cc

10 rows × 60 columns

In [23]:
```
final_test.drop(['Customer Id', 'Customer Lname', 'Customer Password', 'Custom
er State', 'Customer Street'], axis=1, inplace=True)
```

In [24]:
```
final_test.head(10)
```

Out[24]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Late_delivery_risk	Customer Segment	Customer Zipcode	Department Id	Dep
0	3	4	91.250000	0	Consumer	725.0	2	
1	5	4	-249.089996	1	Consumer	725.0	2	
2	4	4	-247.779999	0	Consumer	95125.0	2	
3	3	4	22.860001	0	Home Office	90027.0	2	
4	2	4	134.210007	0	Corporate	725.0	2	
5	6	4	18.580000	0	Consumer	14150.0	2	
6	2	1	95.180000	1	Home Office	725.0	2	
7	2	1	68.430000	1	Corporate	33162.0	2	
8	3	2	133.720001	1	Corporate	725.0	2	
9	2	1	132.149994	1	Corporate	94583.0	2	

10 rows × 55 columns

In [33]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.feature_selection import RFE

cols = ['Days for shipping (real)', 'Days for shipment (scheduled)', 'Late_del
ivery_risk',
       'Delivery Status_Advance shipping',
       'Delivery Status_Late delivery', 'Delivery Status_Shipping canceled',
'Delivery Status_Shipping on time',
        'Market_Africa', 'Market_Europe', 'Market_LATAM', 'Market_Pacific Asi
a',
       'Market_USCA', 'Order Status_CANCELED', 'Order Status_CLOSED', 'Order S
tatus_COMPLETE',
       'Order Status_ON_HOLD', 'Order Status_PAYMENT_REVIEW', 'Order Status_PE
NDING', 'Order Status_PENDING_PAYMENT',
       'Order Status_PROCESSING', 'Order Status_SUSPECTED_FRAUD']
X = final_test[cols]
y = final_test['Late_delivery_risk']
# Build a logreg and compute the feature importances
model = LogisticRegression()
# create the RFE model and select 8 attributes
rfe = RFE(model, 57)
rfe = rfe.fit(X, y)
# summarize the selection of the attributes
print('Selected features: %s' % list(X.columns[rfe.support_]))
print(rfe.ranking_)
```

```
Selected features: ['Days for shipping (real)', 'Days for shipment (schedule
d)', 'Late_delivery_risk', 'Delivery Status_Advance shipping', 'Delivery Stat
us_Late delivery', 'Delivery Status_Shipping canceled', 'Delivery Status_Ship
ping on time', 'Market_Africa', 'Market_Europe', 'Market_LATAM', 'Market_Paci
fic Asia', 'Market_USCA', 'Order Status_CANCELED', 'Order Status_CLOSED', 'Or
der Status_COMPLETE', 'Order Status_ON_HOLD', 'Order Status_PAYMENT_REVIEW',
'Order Status_PENDING', 'Order Status_PENDING_PAYMENT', 'Order Status_PROCESS
ING', 'Order Status_SUSPECTED_FRAUD']
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [33]:
```
final_test.head(10)
```

Out[33]:

	Days for shipping (real)	Days for shipment (scheduled)	Benefit per order	Sales per customer	Late_delivery_risk	Category Id	Category Name	Custo
0	3	4	91.250000	314.640015	0	73	Sporting Goods	Ca
1	5	4	-249.089996	311.359985	1	73	Sporting Goods	Ca
2	4	4	-247.779999	309.720001	0	73	Sporting Goods	San
3	3	4	22.860001	304.809998	0	73	Sporting Goods	An
4	2	4	134.210007	298.250000	0	73	Sporting Goods	Ca
5	6	4	18.580000	294.980011	0	73	Sporting Goods	Tonaw
6	2	1	95.180000	288.420013	1	73	Sporting Goods	Ca
7	2	1	68.430000	285.140015	1	73	Sporting Goods	M
8	3	2	133.720001	278.589996	1	73	Sporting Goods	Ca
9	2	1	132.149994	275.309998	1	73	Sporting Goods	Ra

10 rows × 96 columns

In [34]:
```
final_test.columns.values
```

Out[34]:
```
array(['Days for shipping (real)', 'Days for shipment (scheduled)',
       'Benefit per order', 'Sales per customer', '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', '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 State', 'Order Zipcode',
       'Product Card Id', 'Product Category Id', 'Product Description',
       'Product Image', 'Product Name', 'Product Price', 'Product Status',
       'shipping date (DateOrders)', 'Type_CASH', 'Type_DEBIT',
       'Type_PAYMENT', 'Type_TRANSFER',
       'Delivery Status_Advance shipping',
       'Delivery Status_Late delivery',
       'Delivery Status_Shipping canceled',
       'Delivery Status_Shipping on time', 'Market_Africa',
       'Market_Europe', 'Market_LATAM', 'Market_Pacific Asia',
       'Market_USCA', 'Order Region_Canada', 'Order Region_Caribbean',
       'Order Region_Central Africa', 'Order Region_Central America',
       'Order Region_Central Asia', 'Order Region_East Africa',
       'Order Region_East of USA', 'Order Region_Eastern Asia',
       'Order Region_Eastern Europe', 'Order Region_North Africa',
       'Order Region_Northern Europe', 'Order Region_Oceania',
       'Order Region_South America', 'Order Region_South Asia',
       'Order Region_South of  USA ', 'Order Region_Southeast Asia',
       'Order Region_Southern Africa', 'Order Region_Southern Europe',
       'Order Region_US Center ', 'Order Region_West Africa',
       'Order Region_West Asia', 'Order Region_West of USA ',
       'Order Region_Western Europe', 'Order Status_CANCELED',
       'Order Status_CLOSED', 'Order Status_COMPLETE',
       'Order Status_ON_HOLD', 'Order Status_PAYMENT_REVIEW',
       'Order Status_PENDING', 'Order Status_PENDING_PAYMENT',
       'Order Status_PROCESSING', 'Order Status_SUSPECTED_FRAUD',
       'Shipping Mode_First Class', 'Shipping Mode_Same Day',
       'Shipping Mode_Second Class', 'Shipping Mode_Standard Class'],
      dtype=object)
```

In [37]:
```
final_test.head(10)
```

Out[37]:

	Days for shipping (real)	Days for shipment (scheduled)	Late_delivery_risk	Product Status	Delivery Status_Advance shipping	Delivery Status_Late delivery	Deliv Status_Shipp cance
0	3	4	0	0	1	0	
1	5	4	1	0	0	1	
2	4	4	0	0	0	0	
3	3	4	0	0	1	0	
4	2	4	0	0	1	0	
5	6	4	0	0	0	0	
6	2	1	1	0	0	1	
7	2	1	1	0	0	1	
8	3	2	1	0	0	1	
9	2	1	1	0	0	1	

10 rows × 22 columns

In [39]:
```
final_test.drop(['Product Status'], axis=1, inplace=True)
```

In [40]:
```
final_test.head(10)
```

Out[40]:

	Days for shipping (real)	Days for shipment (scheduled)	Late_delivery_risk	Delivery Status_Advance shipping	Delivery Status_Late delivery	Delivery Status_Shipping canceled	Statu
0	3	4	0	1	0	0	
1	5	4	1	0	1	0	
2	4	4	0	0	0	0	
3	3	4	0	1	0	0	
4	2	4	0	1	0	0	
5	6	4	0	0	0	1	
6	2	1	1	0	1	0	
7	2	1	1	0	1	0	
8	3	2	1	0	1	0	
9	2	1	1	0	1	0	

10 rows × 21 columns

In [41]:
```
final_test.columns.values
```

Out[41]:
```
array(['Days for shipping (real)', 'Days for shipment (scheduled)',
       'Late_delivery_risk', 'Delivery Status_Advance shipping',
       'Delivery Status_Late delivery',
       'Delivery Status_Shipping canceled',
       'Delivery Status_Shipping on time', 'Market_Africa',
       'Market_Europe', 'Market_LATAM', 'Market_Pacific Asia',
       'Market_USCA', 'Order Status_CANCELED', 'Order Status_CLOSED',
       'Order Status_COMPLETE', 'Order Status_ON_HOLD',
       'Order Status_PAYMENT_REVIEW', 'Order Status_PENDING',
       'Order Status_PENDING_PAYMENT', 'Order Status_PROCESSING',
       'Order Status_SUSPECTED_FRAUD'], dtype=object)
```

In [34]:
```
SGSCcols=['Order Status_CLOSED', 'Order Status_COMPLETE', 'Order Status_ON_HOLD','Order Status_PAYMENT_REVIEW',
          'Order Status_PENDING', 'Order Status_PENDING_PAYMENT', 'Order Status_PROCESSING', 'Order Status_CANCELED',
          'Order Status_SUSPECTED_FRAUD']
X=final_test[SGSCcols]
y=final_test['Late_delivery_risk']
```

In [35]:
```
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.653108
         Iterations: 35

C:\Users\tyler\anaconda3\lib\site-packages\statsmodels\base\model.py:567: Con
vergenceWarning: Maximum Likelihood optimization failed to converge. Check ml
e_retvals
  warn("Maximum Likelihood optimization failed to converge. "

                           Logit Regression Results
==============================================================================
=
Dep. Variable:     Late_delivery_risk   No. Observations:               18051
9
Model:                          Logit   Df Residuals:                   18051
0
Method:                           MLE   Df Model:
8
Date:                 Wed, 11 Aug 2021   Pseudo R-squ.:                 0.0513
7
Time:                         10:58:48   Log-Likelihood:            -1.1790e+0
5
converged:                       False   LL-Null:                   -1.2428e+0
5
Covariance Type:             nonrobust   LLR p-value:                     0.00
0
==============================================================================
====================
                                   coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
------------------
Order Status_CLOSED              0.2669      0.014     18.523      0.000
0.239       0.295
Order Status_COMPLETE            0.3017      0.008     36.380      0.000
0.285       0.318
Order Status_ON_HOLD             0.2245      0.020     11.046      0.000
0.185       0.264
Order Status_PAYMENT_REVIEW      0.2883      0.046      6.207      0.000
0.197       0.379
Order Status_PENDING             0.3188      0.014     22.384      0.000
0.291       0.347
Order Status_PENDING_PAYMENT     0.3042      0.010     30.007      0.000
0.284       0.324
Order Status_PROCESSING          0.2854      0.014     20.903      0.000
0.259       0.312
Order Status_CANCELED          -20.8600    557.259     -0.037      0.970   -1
113.068    1071.348
Order Status_SUSPECTED_FRAUD   -17.3669     92.641     -0.187      0.851    -
198.939     164.206
==============================================================================
====================
```

In [57]:
```
SGSCcols=['Market_Africa', 'Market_Europe', 'Market_LATAM', 'Market_Pacific As
ia', 'Market_USCA']
X=final_test[SGSCcols]
y=final_test['Late_delivery_risk']
```

In [58]:
```
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

```
Optimization terminated successfully.
         Current function value: 0.688452
         Iterations 4
                          Logit Regression Results
==============================================================================
=
Dep. Variable:     Late_delivery_risk   No. Observations:                18051
9
Model:                          Logit   Df Residuals:                    18051
4
Method:                           MLE   Df Model:
4
Date:                Sun, 08 Aug 2021   Pseudo R-squ.:                3.480e-0
5
Time:                        16:16:29   Log-Likelihood:            -1.2428e+0
5
converged:                       True   LL-Null:                   -1.2428e+0
5
Covariance Type:            nonrobust   LLR p-value:                    0.0704
8
==============================================================================
==========
                           coef    std err          z      P>|z|      [0.025
0.975]
------------------------------------------------------------------------------
----------
Market_Africa            0.1841      0.019      9.878      0.000       0.148
0.221
Market_Europe            0.2091      0.009     23.306      0.000       0.191
0.227
Market_LATAM             0.1746      0.009     19.760      0.000       0.157
0.192
Market_Pacific Asia      0.2025      0.010     20.465      0.000       0.183
0.222
Market_USCA              0.1926      0.013     15.398      0.000       0.168
0.217
==============================================================================
==========
```

In [63]:
```
SGSCcols=['Delivery Status_Advance shipping', 'Delivery Status_Shipping cancel
ed',
          'Delivery Status_Shipping on time']
X=final_test[SGSCcols]
y=final_test['Late_delivery_risk']
```

In [64]:
```
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

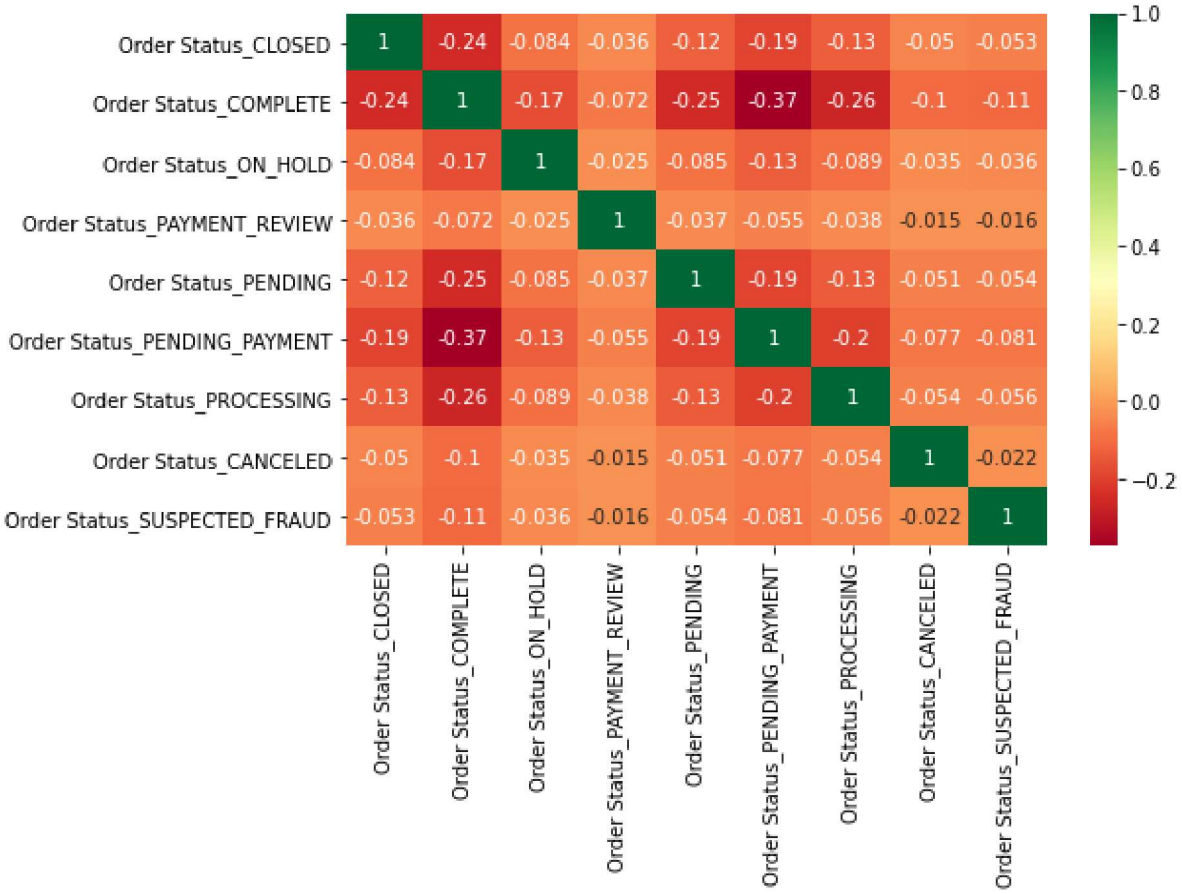
```
Warning: Maximum number of iterations has been exceeded.
         Current function value: 0.380047
         Iterations: 35

C:\Users\tyler\anaconda3\lib\site-packages\statsmodels\base\model.py:567: Con
vergenceWarning: Maximum Likelihood optimization failed to converge. Check ml
e_retvals
  warn("Maximum Likelihood optimization failed to converge. "

                           Logit Regression Results
==============================================================================
=
Dep. Variable:     Late_delivery_risk   No. Observations:                18051
9
Model:                          Logit   Df Residuals:                    18051
6
Method:                           MLE   Df Model:
2
Date:                Sun, 08 Aug 2021   Pseudo R-squ.:                   0.448
0
Time:                        16:20:54   Log-Likelihood:                -6860
6.
converged:                      False   LL-Null:                    -1.2428e+0
5
Covariance Type:            nonrobust   LLR p-value:                     0.00
0
==============================================================================
========================
                                        coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
------------------------
Delivery Status_Advance shipping    -49.8971   3.35e+08  -1.49e-07      1.000
-6.57e+08    6.57e+08
Delivery Status_Shipping canceled   -23.0452   1146.661     -0.020      0.984
-2270.459    2224.368
Delivery Status_Shipping on time    -77.6382   4.03e+14  -1.93e-13      1.000
-7.89e+14    7.89e+14
==============================================================================
========================

Possibly complete quasi-separation: A fraction 0.45 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

In [61]:
```
SGSCcols=['Days for shipping (real)', 'Days for shipment (scheduled)']
X=final_test[SGSCcols]
y=final_test['Late_delivery_risk']
```

In [62]:
```
import statsmodels.api as sm
logit_model=sm.Logit(y,X)
result=logit_model.fit()
print(result.summary())
```

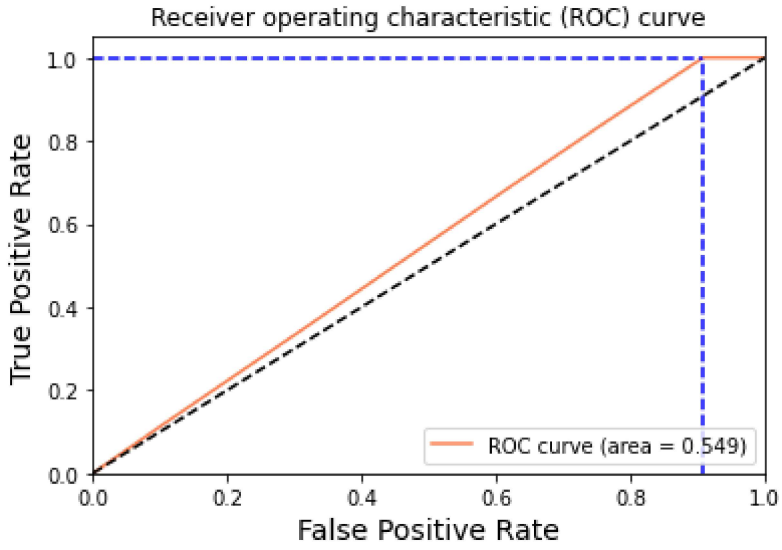
```
Optimization terminated successfully.
         Current function value: 0.187854
         Iterations 9
                           Logit Regression Results
==============================================================================
=
Dep. Variable:     Late_delivery_risk   No. Observations:                18051
9
Model:                          Logit   Df Residuals:                    18051
7
Method:                           MLE   Df Model:
1
Date:                Sun, 08 Aug 2021   Pseudo R-squ.:                   0.727
1
Time:                        16:18:52   Log-Likelihood:                -3391
1.
converged:                       True   LL-Null:                      -1.2428e+0
5
Covariance Type:            nonrobust   LLR p-value:                     0.00
0
==============================================================================
====================
                                   coef    std err          z      P>|z|
[0.025      0.975]
------------------------------------------------------------------------------
--------------------
Days for shipping (real)          3.6384      0.018    196.909      0.000
3.602       3.675
Days for shipment (scheduled)    -4.2322      0.022   -191.281      0.000
-4.276      -4.189
==============================================================================
====================

Possibly complete quasi-separation: A fraction 0.19 of observations can be
perfectly predicted. This might indicate that there is complete
quasi-separation. In this case some parameters will not be identified.
```

In [36]:
```
Selected_features = ['Order Status_CLOSED', 'Order Status_COMPLETE', 'Order St
atus_ON_HOLD','Order Status_PAYMENT_REVIEW',
          'Order Status_PENDING', 'Order Status_PENDING_PAYMENT', 'Order Statu
s_PROCESSING', 'Order Status_CANCELED',
          'Order Status_SUSPECTED_FRAUD']
X = final_test[Selected_features]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



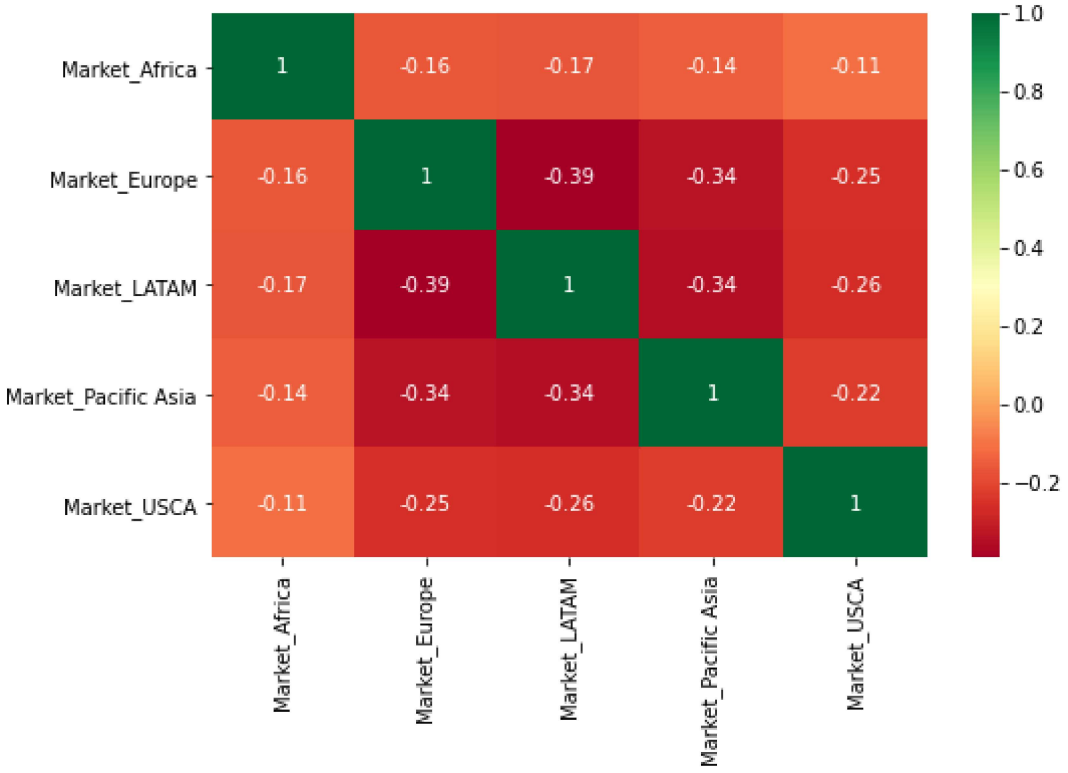
In [46]:
```
from sklearn.model_selection import train_test_split, cross_val_score
from sklearn.metrics import accuracy_score, classification_report, precision_s
core, recall_score
from sklearn.metrics import confusion_matrix, precision_recall_curve, roc_curv
e, auc, log_loss


# create X (features) and y (response)
X = final_test[Selected_features]
y = final_test['Late_delivery_risk']

# use train/test split with different random_state values
# we can change the random_state values that changes the accuracy scores
# the scores change a lot, this is why testing scores is a high-variance estim
ate
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=2)

# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test,
y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred
_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))

idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which th
e sensibility > 0.95

plt.figure()
plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % auc(fpr,
tpr))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--', color='blue')
plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--', color='blue')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate', fontsize=14)
plt.ylabel('True Positive Rate', fontsize=14)
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()

print("Using a threshold of %.3f " % thr[idx] + "guarantees a sensitivity of
%.3f " % tpr[idx] +
      "and a specificity of %.3f" % (1-fpr[idx]) +
      ", i.e. a false positive rate of %.2f%%." % (np.array(fpr[idx])*100))
```

```
Train/Test split results:
LogisticRegression accuracy is 0.588
LogisticRegression log_loss is 0.655
LogisticRegression auc is 0.549
```



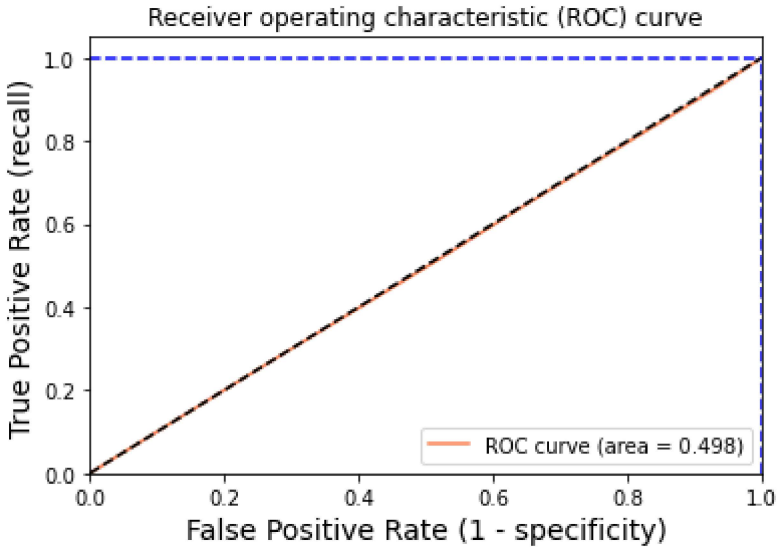

```
Using a threshold of 0.554 guarantees a sensitivity of 1.000 and a specificit
y of 0.092, i.e. a false positive rate of 90.79%.
```

In [38]:
```
Selected_features1 = ['Market_Africa', 'Market_Europe', 'Market_LATAM', 'Marke
t_Pacific Asia', 'Market_USCA']
X = final_test[Selected_features1]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



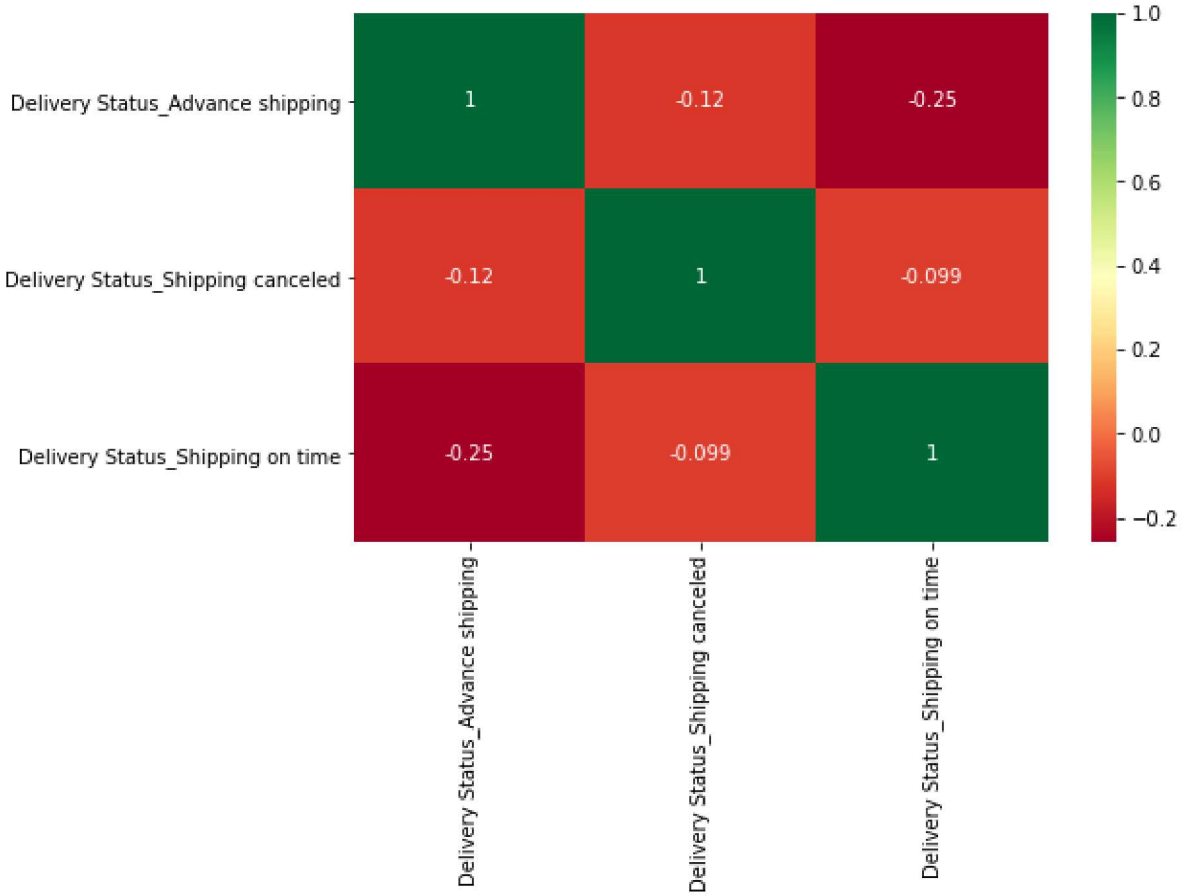
In [39]:
```
# create X (features) and y (response)
X = final_test[Selected_features1]
y = final_test['Late_delivery_risk']

# use train/test split with different random_state values
# we can change the random_state values that changes the accuracy scores
# the scores change a lot, this is why testing scores is a high-variance estim
ate
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=2)

# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test,
y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred
_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))

idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which th
e sensibility > 0.95

plt.figure()
plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % auc(fpr,
tpr))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--', color='blue')
plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--', color='blue')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - specificity)', fontsize=14)
plt.ylabel('True Positive Rate (recall)', fontsize=14)
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()

print("Using a threshold of %.3f " % thr[idx] + "guarantees a sensitivity of
%.3f " % tpr[idx] +
      "and a specificity of %.3f" % (1-fpr[idx]) +
      ", i.e. a false positive rate of %.2f%%." % (np.array(fpr[idx])*100))
```

```
Train/Test split results:
LogisticRegression accuracy is 0.546
LogisticRegression log_loss is 0.689
LogisticRegression auc is 0.498
```




```
Using a threshold of 0.541 guarantees a sensitivity of 1.000 and a specificit
y of 0.000, i.e. a false positive rate of 100.00%.
```

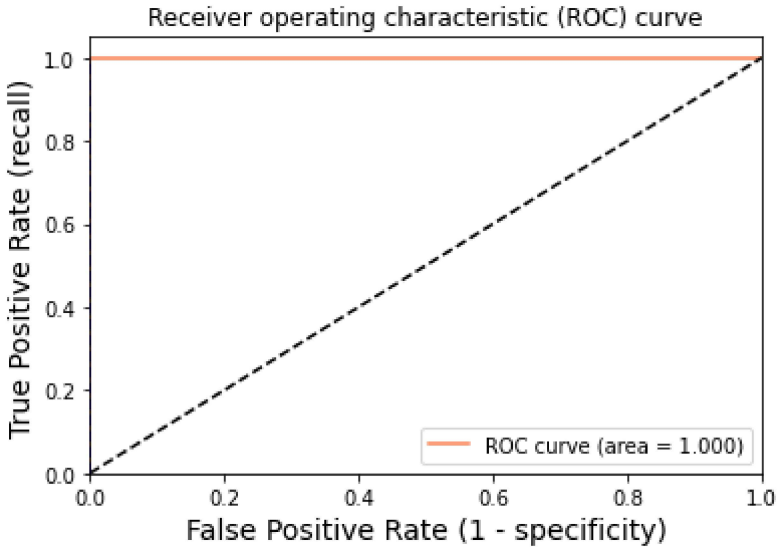
In [40]:
```
Selected_features2 = ['Delivery Status_Advance shipping', 'Delivery Status_Shi
pping canceled',
                      'Delivery Status_Shipping on time']
X = final_test[Selected_features2]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```



In [41]:

```
X = final_test[Selected_features2]
y = final_test['Late_delivery_risk']

# use train/test split with different random_state values
# we can change the random_state values that changes the accuracy scores
# the scores change a lot, this is why testing scores is a high-variance estim
ate
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=2)

# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test,
y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred
_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))

idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which th
e sensibility > 0.95

plt.figure()
plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % auc(fpr,
tpr))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--', color='blue')
plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--', color='blue')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - specificity)', fontsize=14)
plt.ylabel('True Positive Rate (recall)', fontsize=14)
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()

print("Using a threshold of %.3f " % thr[idx] + "guarantees a sensitivity of
%.3f " % tpr[idx] +
      "and a specificity of %.3f" % (1-fpr[idx]) +
      ", i.e. a false positive rate of %.2f%%." % (np.array(fpr[idx])*100))
```

```
Train/Test split results:
LogisticRegression accuracy is 1.000
LogisticRegression log_loss is 0.001
LogisticRegression auc is 1.000
```



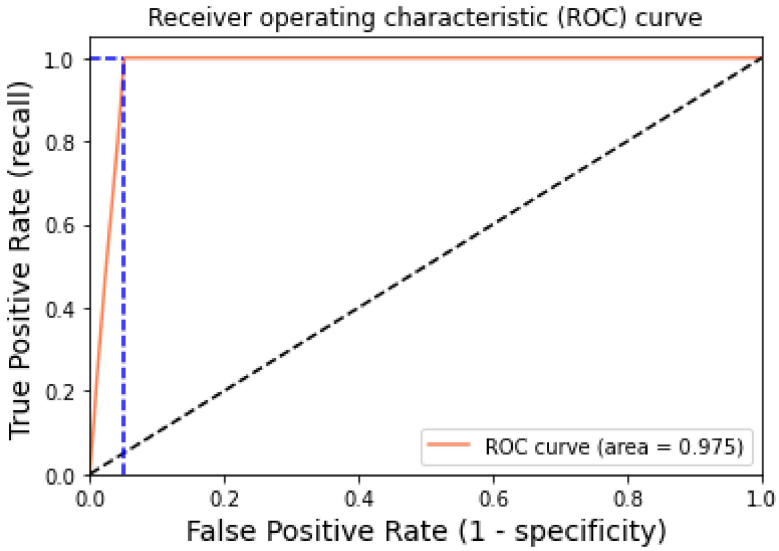

```
Using a threshold of 0.999 guarantees a sensitivity of 1.000 and a specificit
y of 1.000, i.e. a false positive rate of 0.00%.
```

In [42]:
```
Selected_features3 = ['Days for shipping (real)', 'Days for shipment (schedule
d)']
X = final_test[Selected_features3]

plt.subplots(figsize=(8, 5))
sns.heatmap(X.corr(), annot=True, cmap="RdYlGn")
plt.show()
```

In [43]:

```
X = final_test[Selected_features3]
y = final_test['Late_delivery_risk']

# use train/test split with different random_state values
# we can change the random_state values that changes the accuracy scores
# the scores change a lot, this is why testing scores is a high-variance estim
ate
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, rando
m_state=2)

# check classification scores of logistic regression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
y_pred_proba = logreg.predict_proba(X_test)[:, 1]
[fpr, tpr, thr] = roc_curve(y_test, y_pred_proba)
print('Train/Test split results:')
print(logreg.__class__.__name__+" accuracy is %2.3f" % accuracy_score(y_test,
y_pred))
print(logreg.__class__.__name__+" log_loss is %2.3f" % log_loss(y_test, y_pred
_proba))
print(logreg.__class__.__name__+" auc is %2.3f" % auc(fpr, tpr))

idx = np.min(np.where(tpr > 0.95)) # index of the first threshold for which th
e sensibility > 0.95

plt.figure()
plt.plot(fpr, tpr, color='coral', label='ROC curve (area = %0.3f)' % auc(fpr,
tpr))
plt.plot([0, 1], [0, 1], 'k--')
plt.plot([0,fpr[idx]], [tpr[idx],tpr[idx]], 'k--', color='blue')
plt.plot([fpr[idx],fpr[idx]], [0,tpr[idx]], 'k--', color='blue')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate (1 - specificity)', fontsize=14)
plt.ylabel('True Positive Rate (recall)', fontsize=14)
plt.title('Receiver operating characteristic (ROC) curve')
plt.legend(loc="lower right")
plt.show()

print("Using a threshold of %.3f " % thr[idx] + "guarantees a sensitivity of
%.3f " % tpr[idx] +
      "and a specificity of %.3f" % (1-fpr[idx]) +
      ", i.e. a false positive rate of %.2f%%." % (np.array(fpr[idx])*100))
```

```
Train/Test split results:
LogisticRegression accuracy is 0.977
LogisticRegression log_loss is 0.162
LogisticRegression auc is 0.975
```




```
Using a threshold of 0.845 guarantees a sensitivity of 1.000 and a specificit
y of 0.949, i.e. a false positive rate of 5.13%.
```

In [ ]: