amazon-sales-data-analysis

Use the "Run" button to execute the code.

```
!pip install jovian --upgrade --quiet
```

```
import jovian
```

```
# Execute this to save new versions of the notebook
jovian.commit(project="amazon-sales-data-analysis")
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis
```

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
data = pd.read_csv("SALESDATA.csv")
```

```
/opt/conda/lib/python3.9/site-packages/IPython/core/interactiveshell.py:3441:
DtypeWarning: Columns (21) have mixed types.Specify dtype option on import or set
low_memory=False.
  exec(code_obj, self.user_global_ns, self.user_ns)
```

```
data.shape
```

```
(65282, 22)
```

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65282 entries, 0 to 65281
Data columns (total 22 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   CustKey                         65282 non-null  int64
 1   DateKey                         65282 non-null  object
 2   Discount Amount                 65280 non-null  float64
 3   Invoice Date                    65282 non-null  object
```

```
 4   Invoice Number                     65282 non-null  int64
 5   Item Class                         56993 non-null  object
 6   Item Number                        65241 non-null  object
 7   Item                               65282 non-null  object
 8   Line Number                        65282 non-null  int64
 9   List Price                         65282 non-null  float64
 10  Order Number                       65282 non-null  int64
 11  Promised Delivery Date             65282 non-null  object
 12  Sales Amount                       65282 non-null  float64
 13  Sales Amount Based on List Price   65282 non-null  float64
 14  Sales Cost Amount                  65282 non-null  float64
 15  Sales Margin Amount                65282 non-null  float64
 16  Sales Price                        65281 non-null  float64
 17  Sales Quantity                     65282 non-null  int64
 18  Sales Rep                          65282 non-null  int64
 19  U/M                                65282 non-null  object
 20  Unnamed: 20                        0 non-null      float64
 21  Unnamed: 21                        4 non-null      object
dtypes: float64(8), int64(6), object(8)
memory usage: 11.0+ MB
```

```
data['Invoice Date'] = pd.to_datetime(data['Invoice Date'])
```

```
data['DateKey'] = pd.to_datetime(data['DateKey'])
```

```
# Checking null values
data.isnull().sum()
```

```
CustKey                             0
DateKey                             0
Discount Amount                     2
Invoice Date                        0
Invoice Number                      0
Item Class                       8289
Item Number                        41
Item                                0
Line Number                         0
List Price                          0
Order Number                        0
Promised Delivery Date              0
Sales Amount                        0
Sales Amount Based on List Price    0
Sales Cost Amount                   0
Sales Margin Amount                 0
Sales Price                         1
Sales Quantity                      0
```

```
Sales Rep                          0
U/M                                0
Unnamed: 20                    65282
Unnamed: 21                    65278
dtype: int64
```

```
data01=data.copy()
```

```
# removing Null values
data01.dropna(subset=['Discount Amount','Sales Price','Item Number'],inplace=True)
```

```
data01.describe()
```

	CustKey	Discount Amount	Invoice Number	Line Number	List Price	Order Number	Sales Amc
count	6.524100e+04	65241.000000	65241.000000	65241.000000	65241.000000	65241.000000	65241.000
mean	1.001770e+07	1857.310923	216292.785242	23725.043178	515.016834	180567.610122	2853.121
std	7.175846e+03	9039.535784	94982.018695	32669.565014	449.144896	67612.238675	15169.020
min	1.000045e+07	-255820.800000	100034.000000	1000.000000	0.000000	100838.000000	200.010
25%	1.001272e+07	246.280000	117969.000000	3000.000000	181.560000	115281.000000	308.310
50%	1.001966e+07	442.200000	222904.000000	12000.000000	325.190000	203695.000000	553.940
75%	1.002351e+07	1001.500000	314325.000000	32000.000000	803.860000	218576.000000	1279.750
max	1.002758e+07	343532.660000	332842.000000	344000.000000	2760.700000	321532.000000	555376.000

```
# Creating Year, Month, Quarter, Day Columns in Sales_data01

data01['Invoice_Year'] = data['Invoice Date'].dt.year
data01['Invoice_Month'] = data['Invoice Date'].dt.month
data01['Invoice_Quarter'] = data['Invoice Date'].dt.quarter
data01['Invoice_Day'] = data['Invoice Date'].dt.day
```

```
data01.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 65241 entries, 1 to 65281
Data columns (total 26 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   CustKey                         65241 non-null  int64
 1   DateKey                         65241 non-null  datetime64[ns]
 2   Discount Amount                 65241 non-null  float64
 3   Invoice Date                    65241 non-null  datetime64[ns]
 4   Invoice Number                  65241 non-null  int64
 5   Item Class                      56993 non-null  object
```

```
 6   Item Number                          65241 non-null  object
 7   Item                                 65241 non-null  object
 8   Line Number                          65241 non-null  int64
 9   List Price                           65241 non-null  float64
 10  Order Number                         65241 non-null  int64
 11  Promised Delivery Date               65241 non-null  object
 12  Sales Amount                         65241 non-null  float64
 13  Sales Amount Based on List Price     65241 non-null  float64
 14  Sales Cost Amount                    65241 non-null  float64
 15  Sales Margin Amount                  65241 non-null  float64
 16  Sales Price                          65241 non-null  float64
 17  Sales Quantity                       65241 non-null  int64
 18  Sales Rep                            65241 non-null  int64
 19  U/M                                  65241 non-null  object
 20  Unnamed: 20                          0 non-null      float64
 21  Unnamed: 21                          3 non-null      object
 22  Invoice_Year                         65241 non-null  int64
 23  Invoice_Month                        65241 non-null  int64
 24  Invoice_Quarter                      65241 non-null  int64
 25  Invoice_Day                          65241 non-null  int64
dtypes: datetime64[ns](2), float64(8), int64(10), object(6)
memory usage: 13.4+ MB
```

```
data01.head()
```

	CustKey	DateKey	Discount Amount	Invoice Date	Invoice Number	Item Class	Item Number	Item	Line Number	List Price	...	Sales Price	Qu
1	10002220	2017-07-14	368.790	2017-07-14	100233	P01	20910	Moms Sliced Turkey	1000	824.960	...	456.170000	
2	10002220	2017-10-17	109.730	2017-10-17	116165	P01	38076	Cutting Edge Foot-Long Hot Dogs	1000	548.660	...	438.930000	
4	10004516	2017-05-27	96627.940	2017-05-27	103341	P01	60776	High Top Sweet Onion	1000	408.520	...	196.150901	
6	10007866	2017-03-09	371.014	2017-03-09	100403	P01	20910	Moms Sliced Turkey	2000	795.314	...	424.300000	
7	10009356	2017-06-18	608.080	2017-06-18	105481	P01	62550	Tell Tale Garlic	29000	575.000	...	270.960000	

5 rows × 26 columns

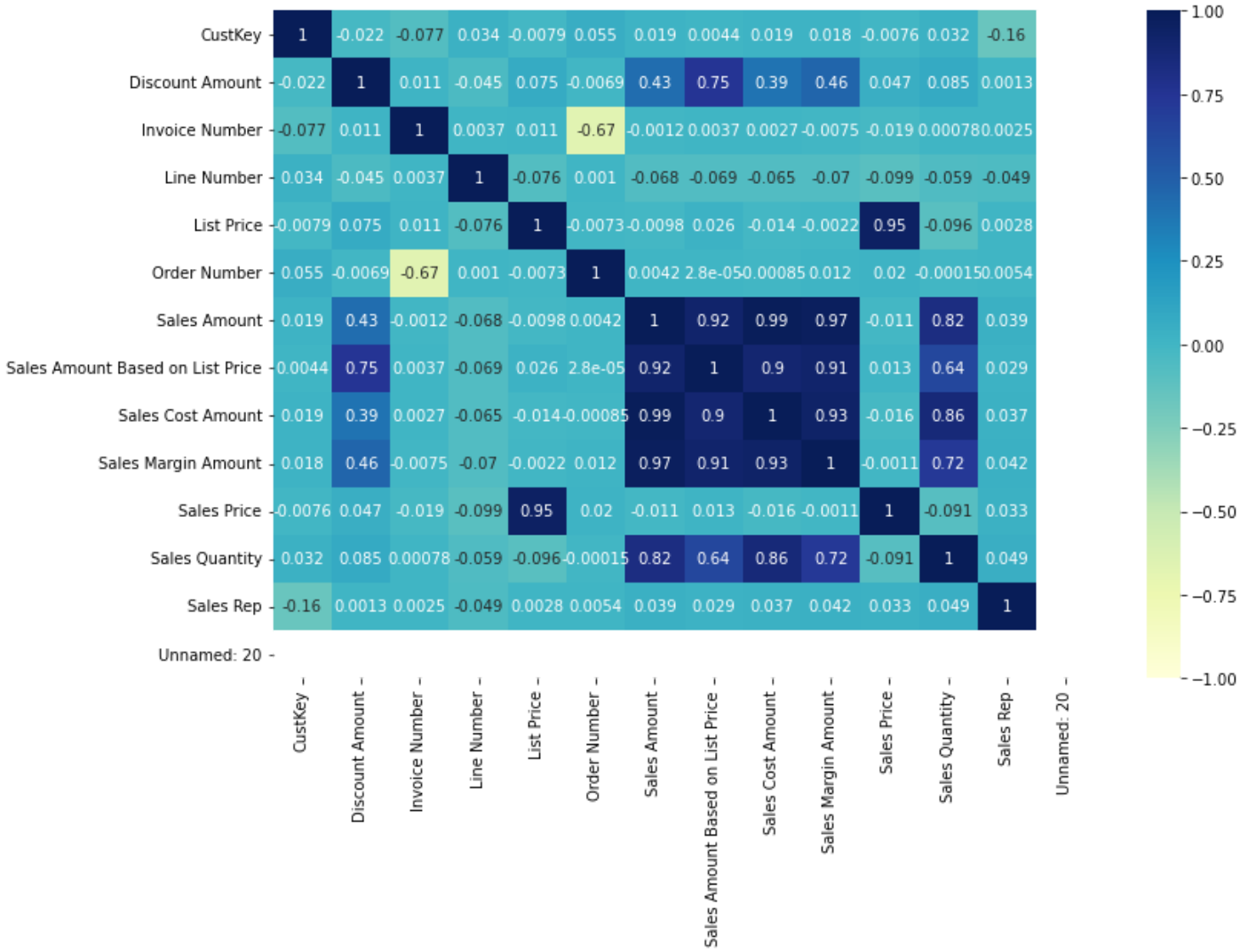
```
# Creating DataFrame only with neccessary values.
data02 = data01[['CustKey','Item','Invoice Date','Invoice_Year','Invoice_Quarter','Invo
                 'Sales Quantity', 'Sales Amount Based on List Price','Disco
                'Sales Amount', 'Sales Margin Amount','Sales Cost Amount','S
                'Sales Price']]
```

```
data02.isnull().sum()
```

```
CustKey                             0
Item                                0
Invoice Date                        0
Invoice_Year                        0
Invoice_Quarter                     0
Invoice_Day                         0
Invoice_Month                       0
Sales Quantity                      0
Sales Amount Based on List Price    0
Discount Amount                     0
Sales Amount                        0
Sales Margin Amount                 0
Sales Cost Amount                   0
Sales Rep                           0
U/M                                 0
List Price                          0
Sales Price                         0
dtype: int64
```

```
#Checking the correlation
plt.figure(figsize=(12,8))
sns.heatmap(data.corr(method='pearson'), annot=True, vmin=-1, vmax=1, cmap='YlGnBu')
```

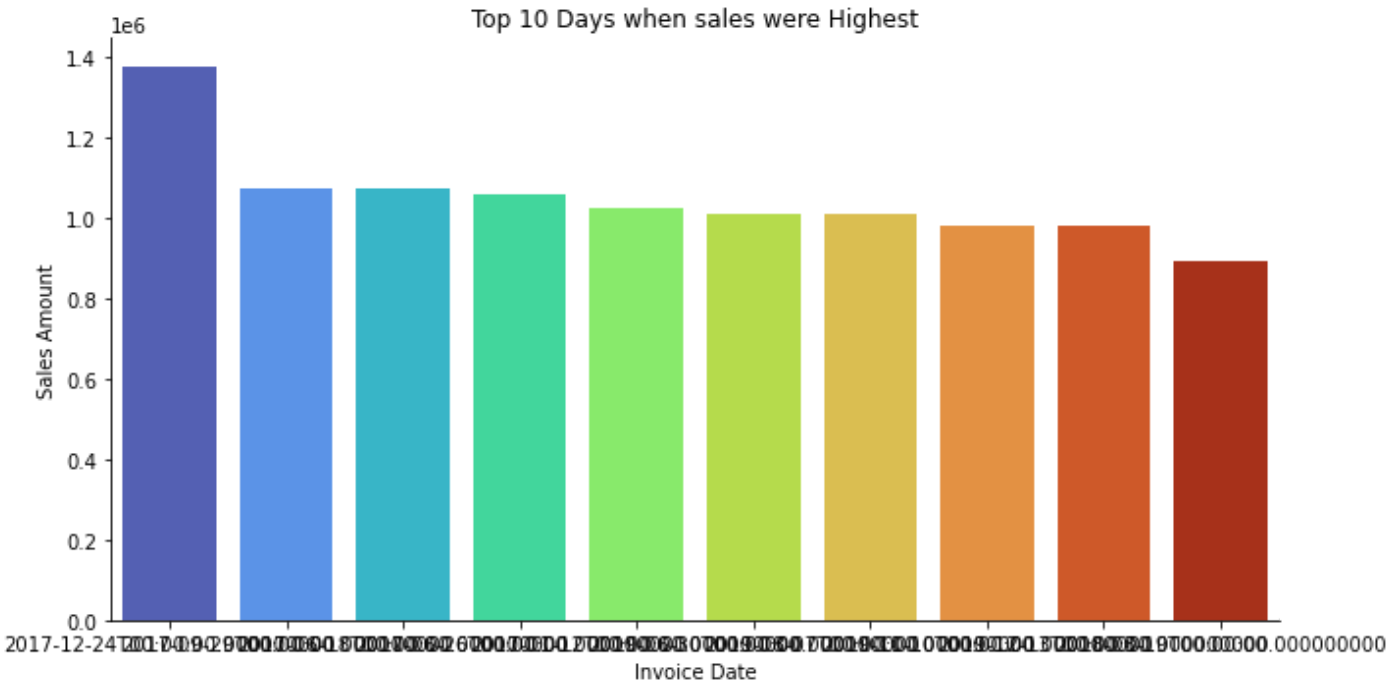
```
<AxesSubplot:>
```




```
daysalesinsights = data01.copy()
daysalesinsights['Invoice_Date'] = pd.to_datetime(data01['Invoice Date']).dt.date
```

```
top10sales = data.groupby('Invoice Date').sum().sort_values('Sales Amount', ascending =
top10sales = top10sales.reset_index().head(10)
```

```
sns.catplot(y='Sales Amount', x = 'Invoice Date', data = top10sales, aspect = 2,palette
plt.title('Top 10 Days when sales were Highest')
top10sales[['Sales Amount']]
plt.show()
sns.set(style='darkgrid')
```

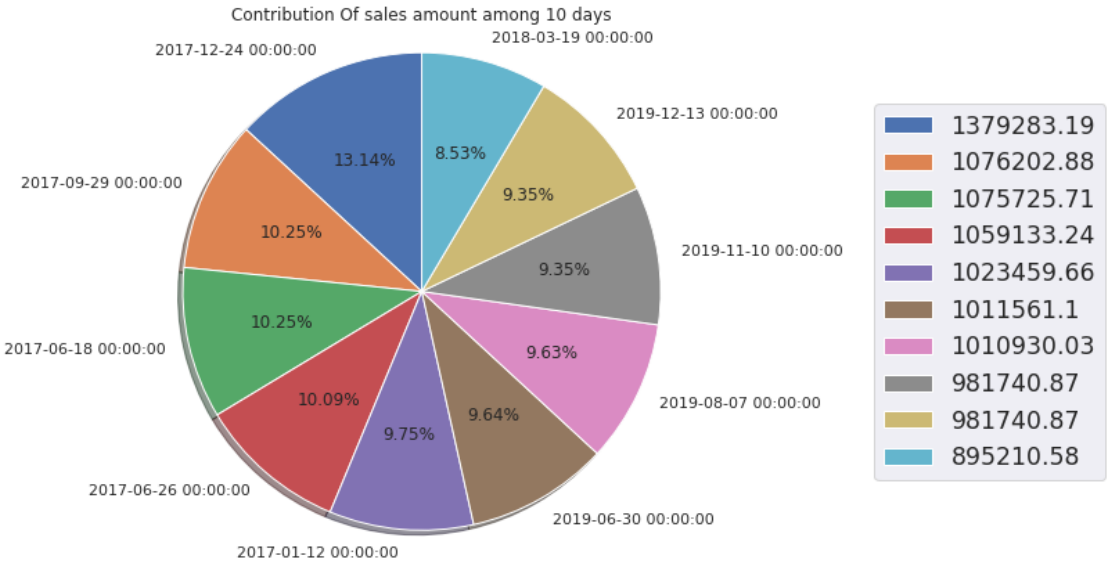



```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis
```

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

```
plt.figure(figsize=(18,7))
plt.pie('Sales Amount',labels='Invoice Date',data = top10sales,
        autopct ='%1.2f%%', shadow = True, startangle = 90)
plt.axis('equal')
plt.title('Contribution Of sales amount among 10 days')
plt.legend(round(top10sales['Sales Amount'],2),loc=7, fontsize ='x-large')
plt.show()
```




```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis
```

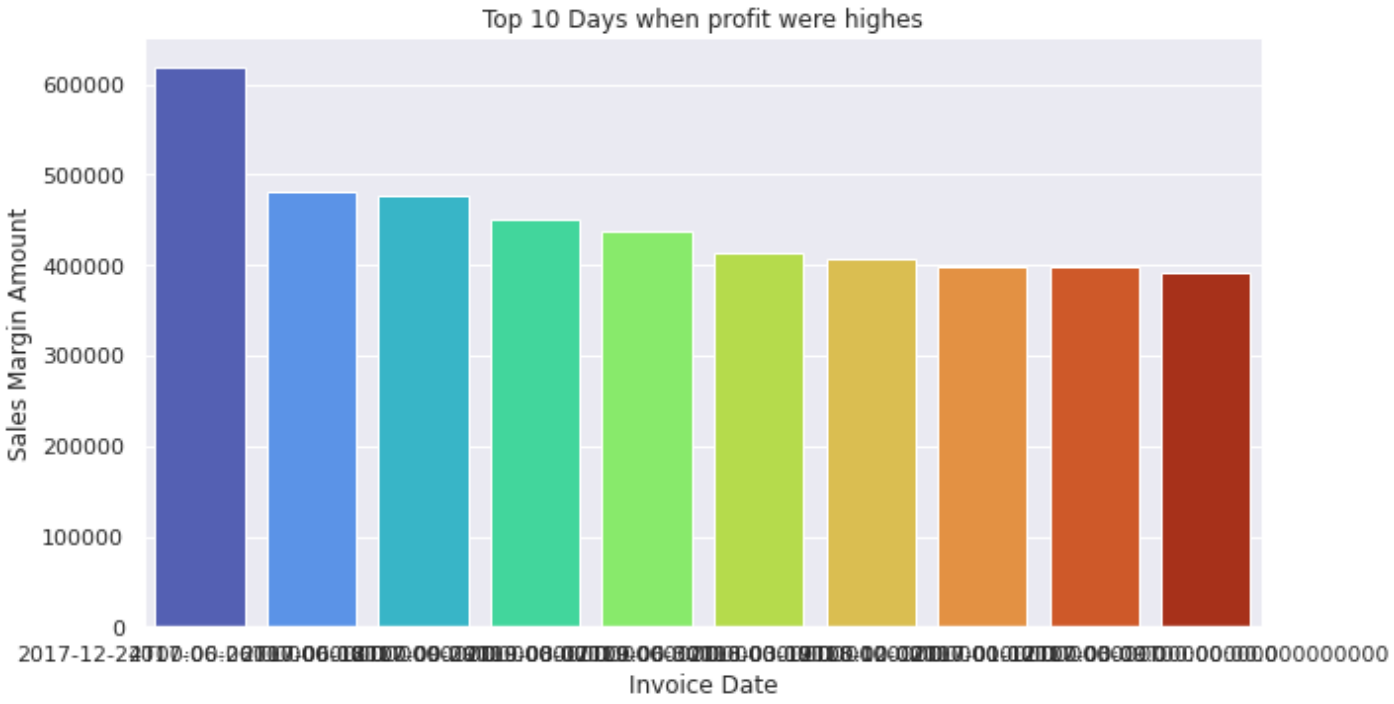
'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

Listing out top 10 days when profit were highest

```
top10profit =daysalesinsights.groupby('Invoice Date').sum().sort_values('Sales Margin A
top10profit = top10profit.reset_index().head(10)
```

```
sns.catplot(y='Sales Margin Amount', x = 'Invoice Date', data= top10profit, aspect = 2,
plt.title("Top 10 Days when profit were highes")
top10profit[['Sales Margin Amount']]
```

	Sales Margin Amount
0	619085.33
1	480769.87
2	477173.34
3	450965.94
4	436621.64
5	413530.29
6	406118.86
7	398430.22
8	398222.72
9	392326.13



```
jovian.commit()
```

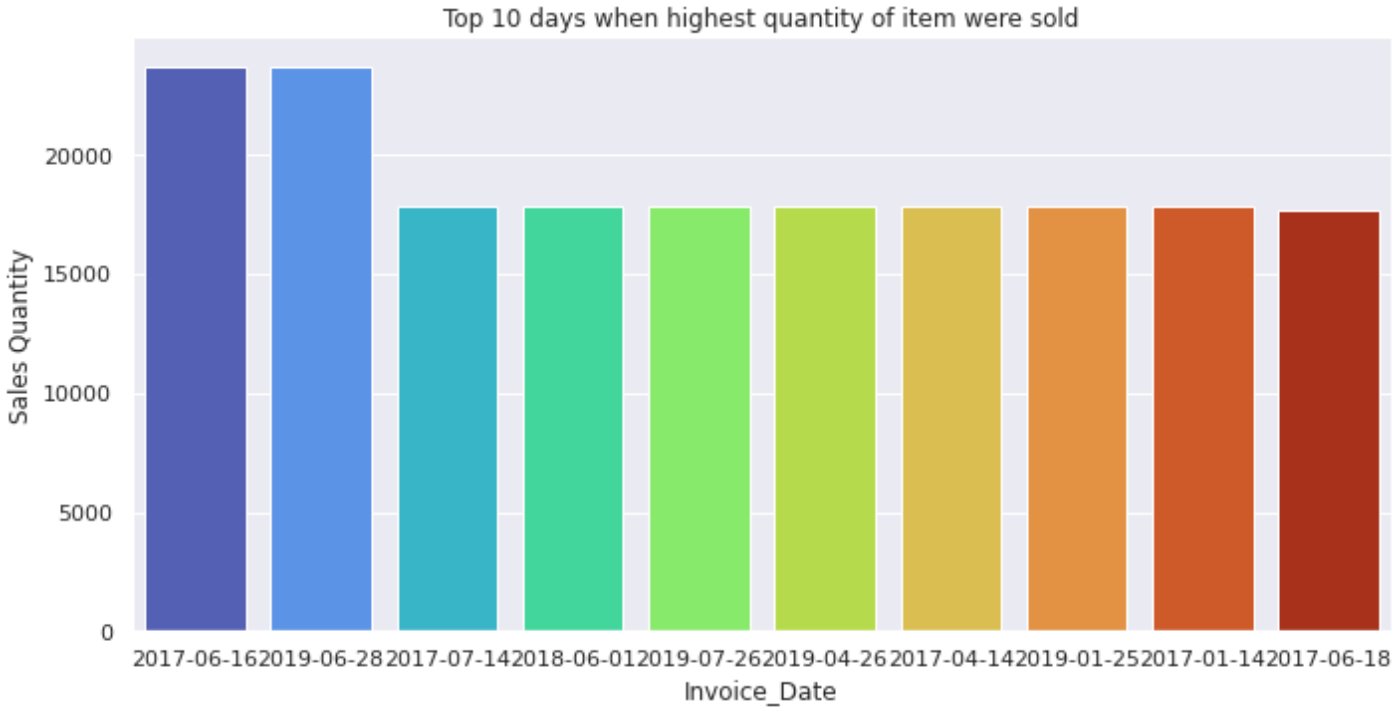
```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis
```

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

```
highqty = daysalesinsights.groupby('Invoice_Date').sum().sort_values('Sales Quantity',
highqty = highqty.reset_index().head(10)
```

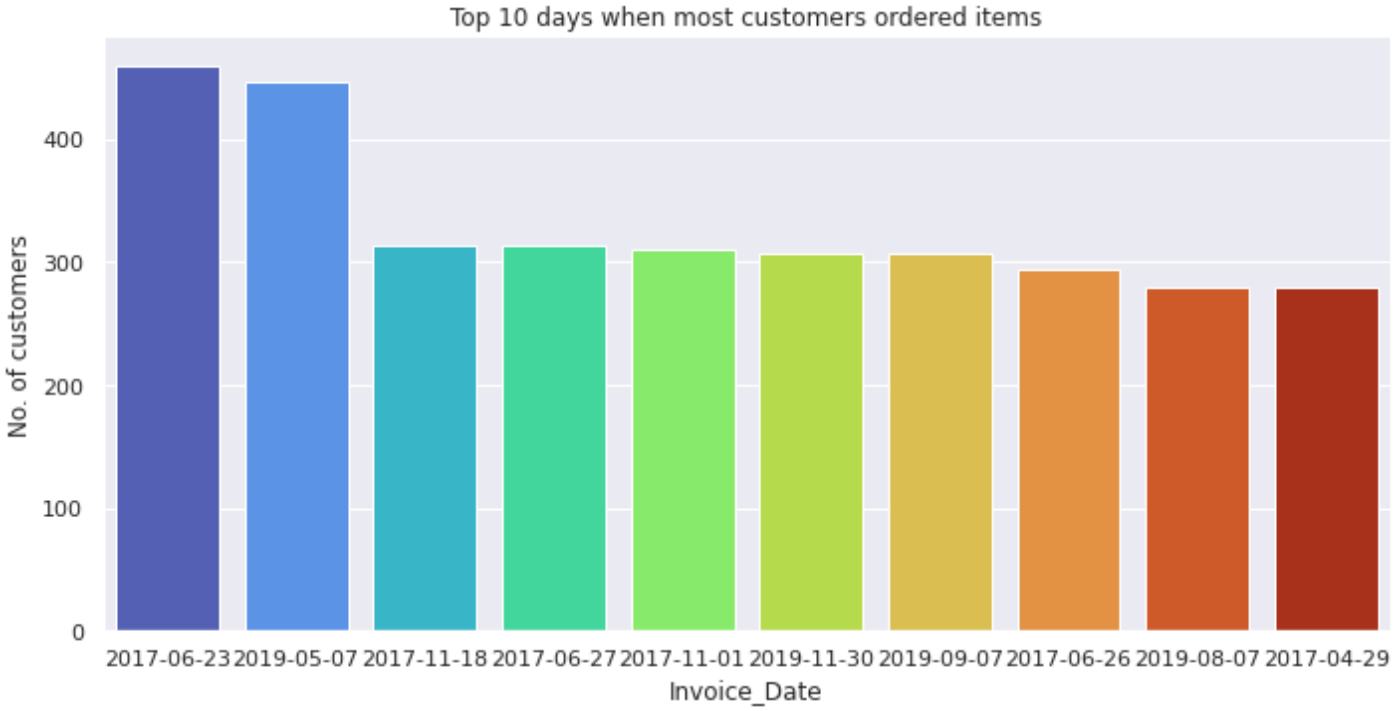
```
sns.catplot(y='Sales Quantity', x = 'Invoice_Date', data = highqty, aspect =2, palette=
plt.title('Top 10 days when highest quantity of item were sold')
highqty[['Sales Quantity']]
```

	Sales Quantity
0	23710
1	23708
2	17822
3	17820
4	17819
5	17819
6	17819
7	17819
8	17819
9	17649



```
mostcust = daysalesinsights.groupby(['Invoice_Date']).count().sort_values('CustKey',asc
mostcust = mostcust.reset_index().head(10)
```

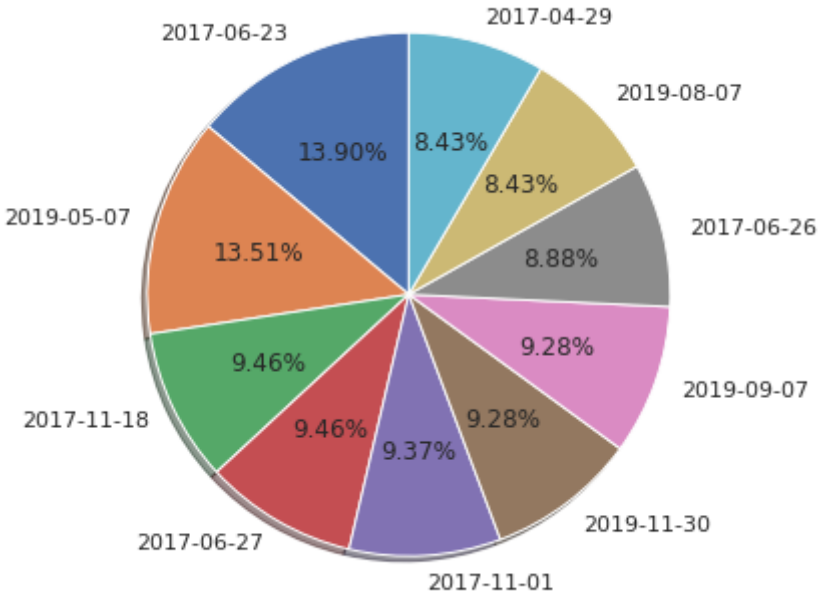
```
sns.catplot(y='CustKey',x='Invoice_Date',data = mostcust,aspect = 2,palette ='turbo',ki
plt.title('Top 10 days when most customers ordered items')
plt.ylabel('No. of customers')
plt.show()
```




```
plt.figure(figsize=(12,6))
plt.pie(x='CustKey',labels = 'Invoice_Date',data = mostcust,
       autopct = '%1.2f%%',shadow = True,startangle = 90)
```

```
([<matplotlib.patches.Wedge at 0x7fc360d66190>,
  <matplotlib.patches.Wedge at 0x7fc360d66a90>,
  <matplotlib.patches.Wedge at 0x7fc360cf4460>,
  <matplotlib.patches.Wedge at 0x7fc360cf4df0>,
  <matplotlib.patches.Wedge at 0x7fc360cff7c0>,
  <matplotlib.patches.Wedge at 0x7fc360d0f190>,
  <matplotlib.patches.Wedge at 0x7fc360d0fb20>,
  <matplotlib.patches.Wedge at 0x7fc360d1c4f0>,
  <matplotlib.patches.Wedge at 0x7fc360d1ce80>,
  <matplotlib.patches.Wedge at 0x7fc360d2a850>],
 [Text(-0.4652744342988959, 0.9967545840315171, '2017-06-23'),
  Text(-1.059276375788081, 0.2965359332311492, '2019-05-07'),
  Text(-0.9911628149042261, -0.47707051297594466, '2017-11-18'),
  Text(-0.5540649010327483, -0.9502694804336141, '2017-06-27'),
  Text(0.06992422907964263, -1.097775296765061, '2017-11-01'),
  Text(0.6651754791813568, -0.8760945051168011, '2019-11-30'),
  Text(1.0375923242028662, -0.36524261629127336, '2019-09-07'),
  Text(1.0704963560403848, 0.25305642000205747, '2017-06-26'),
  Text(0.7849840658683533, 0.770584204570006, '2019-08-07'),
```

```
 Text(0.2879780653392332, 1.0616348872769172, '2017-04-29')],
[Text(-0.25378605507212504, 0.5436843185626457, '13.90%'),
 Text(-0.5777871140662258, 0.1617468726715359, '13.51%'),
 Text(-0.5406342626750323, -0.2602202798050607, '9.46%'),
 Text(-0.3022172187451354, -0.518328807509244, '9.46%'),
 Text(0.03814048858889598, -0.598786525508215, '9.37%'),
 Text(0.3628229886443764, -0.47786973006370964, '9.28%'),
 Text(0.5659594495651997, -0.19922324524978544, '9.28%'),
 Text(0.5839071032947554, 0.13803077454657678, '8.88%'),
 Text(0.42817312683728365, 0.42031865703818505, '8.43%'),
 Text(0.1570789447304908, 0.5790735748783183, '8.43%')])
```

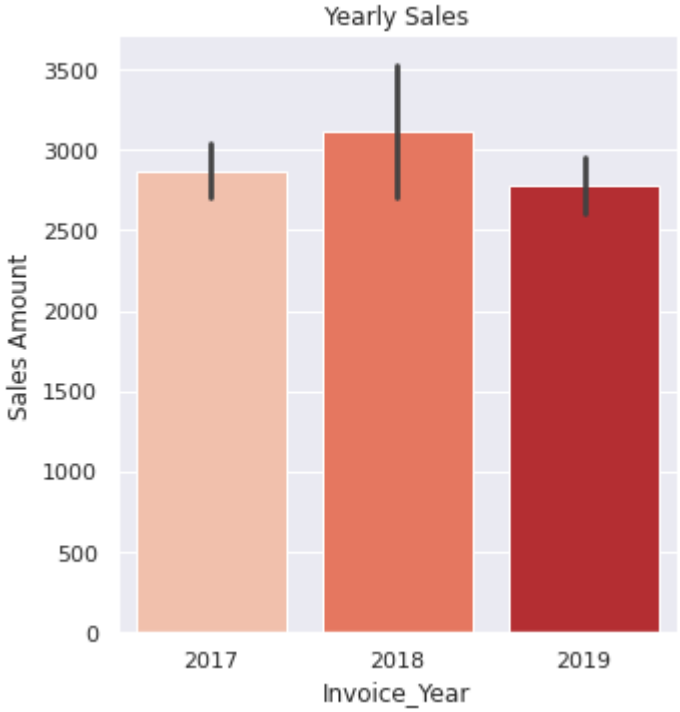


Yearly Sales Record

```
Yearly_Sales =  data02[['CustKey','Item','Invoice Date','Invoice_Year','Invoice_Month',
                        'Sales Quantity', 'Sales Amount Based on List Price','Disco
                       'Sales Amount', 'Sales Margin Amount','Sales Cost Amount','S
                       'Sales Price']]
```

```
yearly_sales01 = Yearly_Sales.groupby('Invoice_Year').sum().reset_index()
sns.catplot(y = 'Sales Amount', x = 'Invoice_Year', data = Yearly_Sales,palette='Reds',
plt.xlabel('Invoice_Year')
plt.ylabel('Sales Amount')
plt.title('Yearly Sales')
yearly_sales01[['Invoice_Year','Sales Amount']]
```

	Invoice_Year	Sales Amount
0	2017	87416407.83
1	2018	20817471.00
2	2019	77906591.65



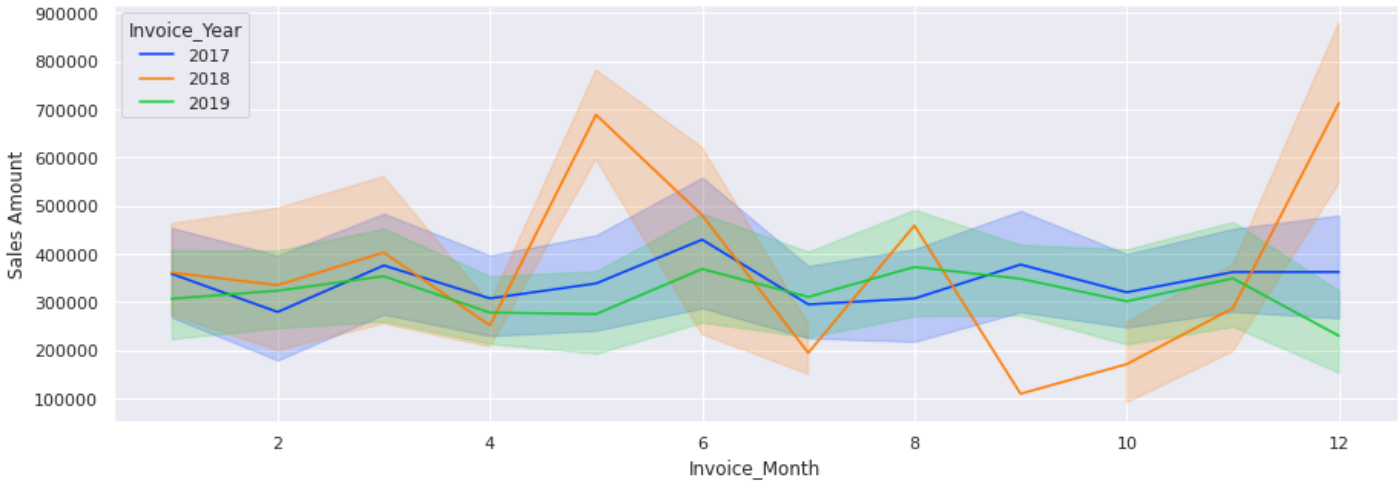

```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis
```

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

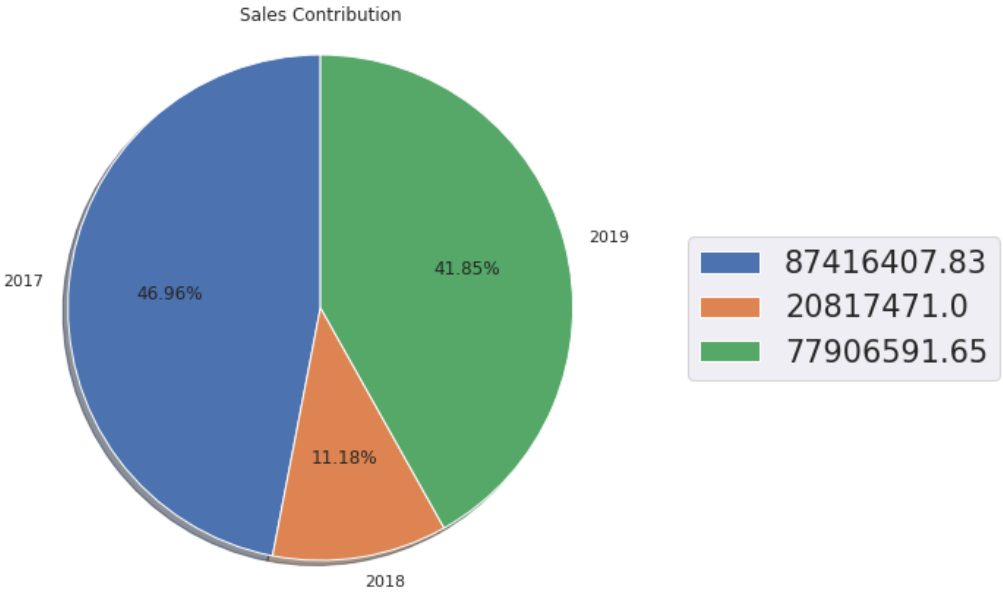
```
plt.figure(figsize=(15,5))
sns.lineplot(y = 'Sales Amount', x = 'Invoice_Month',
             data = data02.groupby(['Invoice Date', 'Invoice_Year','Invoice_Month']).sum
             hue = 'Invoice_Year', palette = 'bright')
```

```
<AxesSubplot:xlabel='Invoice_Month', ylabel='Sales Amount'>
```




```
# Plotting pie chart to know sales sahre among 3 years
plt.figure(figsize=(17,7))
plt.pie('Sales Amount', labels = 'Invoice_Year', data = yearly_sales01,
        autopct = '%1.2f%%', shadow= True, startangle = 90)
```

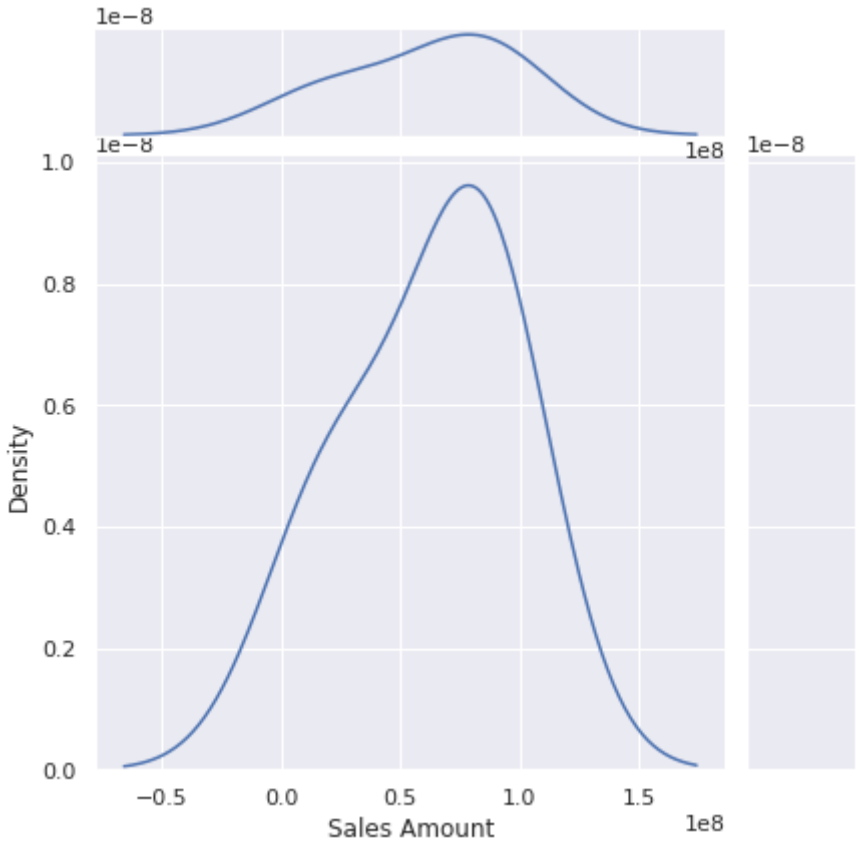
```
plt.axis('equal')
plt.title('Sales Contribution')
plt.legend(round(yearly_sales01['Sales Amount'],2),loc=7, fontsize = 'xx-large')
plt.show()
```



Note: Highest sales amount 2017>2019>2018

```
sns.jointplot(x='Sales Amount', data= yearly_sales01,kind = 'kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc360eb7dc0>
```



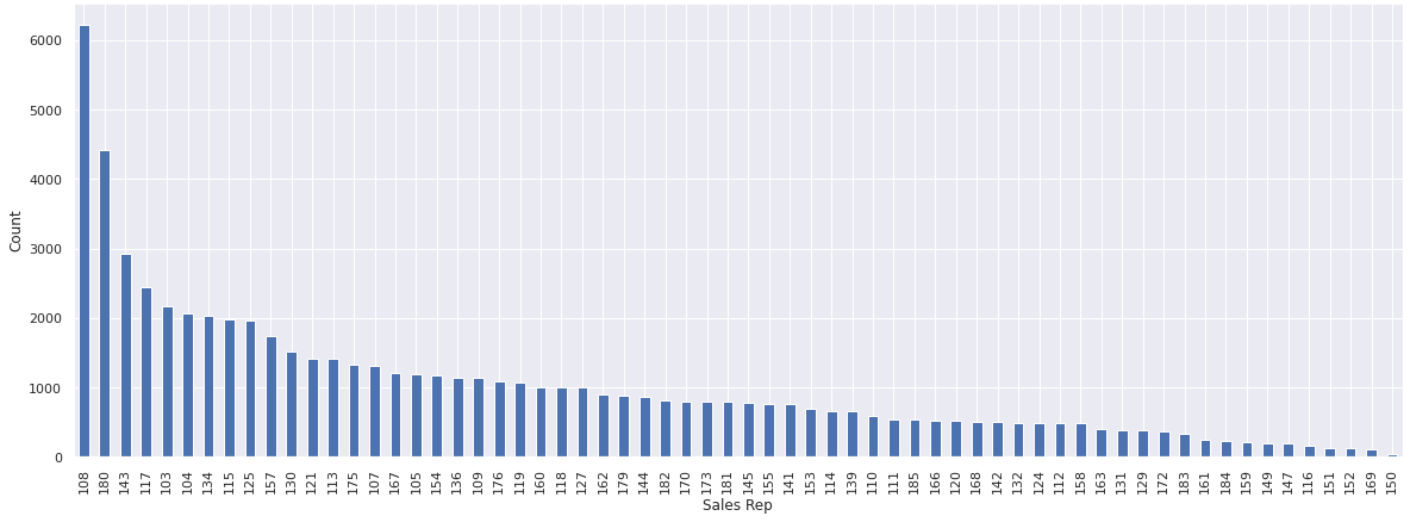

```
sns.relplot(x ='Sales Quantity',y = 'Sales Amount',
            data = data02.groupby(['Invoice_Year','Invoice_Month','Invoice_Day']).sum()
            hue='Invoice_Year',height=8,aspect= 2, size='Invoice_Year',palette='muted',
            style= 'Invoice_Year', sizes = (400,150))
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc360e714c0>
```

Note: From 2017-2019 total quantity of item on any particular day was in the range of 0 to 4000 while the item amount was between $1 and 200000$

```
plt.figure(figsize=(20,7))
data01['Sales Rep'].value_counts().plot.bar()
plt.xlabel('Sales Rep')
plt.ylabel('Count')
```

```
Text(0, 0.5, 'Count')
```



Note: Sales Rep 108 was use the most and sales rep 150 was least used

Yearly_Monthwise Record

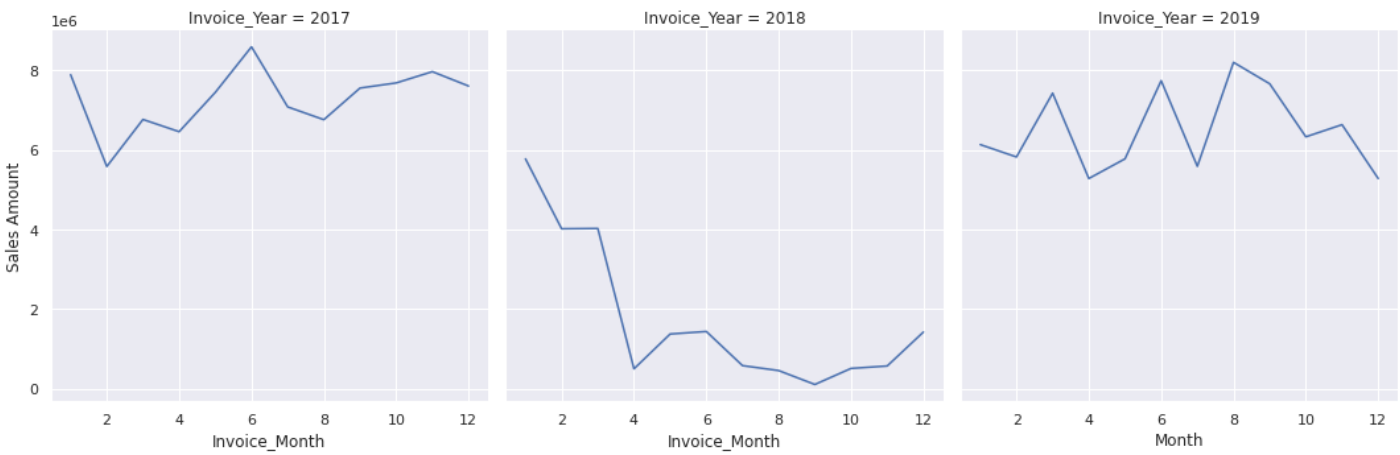
```
yearly_monthwise = data02.groupby(['Invoice_Year','Invoice_Month']).sum().reset_index()
yearly_monthwise.iloc[:,6:].describe()
```

	Sales Amount Based on List Price	Discount Amount	Sales Amount	Sales Margin Amount	Sales Cost Amount	Sales Rep	List Price

	Sales Amount Based on List Price	Discount Amount	Sales Amount	Sales Margin Amount	Sales Cost Amount	Sales Rep	List Price	
count	3.600000e+01	3.600000e+01	3.600000e+01	3.600000e+01	3.600000e+01	36.000000	3.600000e+01	
mean	8.536480e+06	3.365912e+06	5.170569e+06	2.158567e+06	3.012001e+06	249041.972222	9.333393e+05	5
std	4.718837e+06	2.005110e+06	2.787809e+06	1.161547e+06	1.633682e+06	139524.994252	5.313228e+05	2
min	1.823928e+05	7.284534e+04	1.095475e+05	5.082622e+04	5.872126e+04	7381.000000	2.851060e+04	
25%	5.499474e+06	2.120824e+06	3.376615e+06	1.515036e+06	1.862708e+06	161417.750000	6.147058e+05	3
50%	9.370350e+06	3.536524e+06	5.981103e+06	2.509936e+06	3.560870e+06	293311.000000	1.066564e+06	5
75%	1.258521e+07	5.206829e+06	7.476334e+06	3.066029e+06	4.313140e+06	356601.750000	1.321216e+06	7
max	1.396024e+07	6.295524e+06	8.592380e+06	3.678925e+06	4.913455e+06	440248.000000	1.658983e+06	9

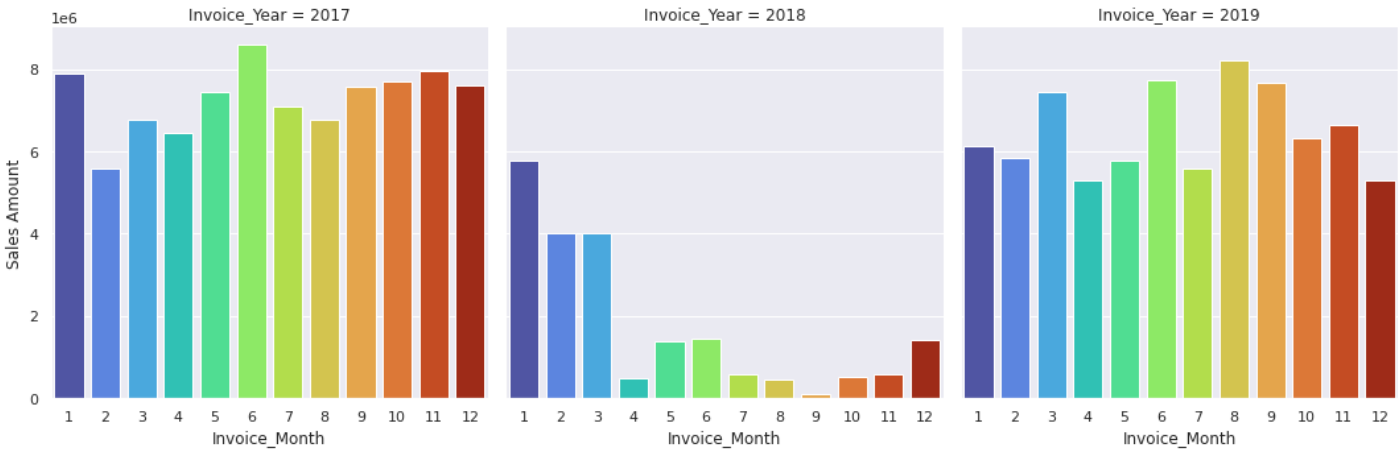
```
sns.relplot(x='Invoice_Month',y= 'Sales Amount', data= yearly_monthwise, height = 5,
            kind = 'line',aspect = 1, col ='Invoice_Year')
plt.xlabel('Month')
plt.ylabel('Sales Amount')
print('yearly_monthwise Sales trend ')
```

```
yearly_monthwise Sales trend
```



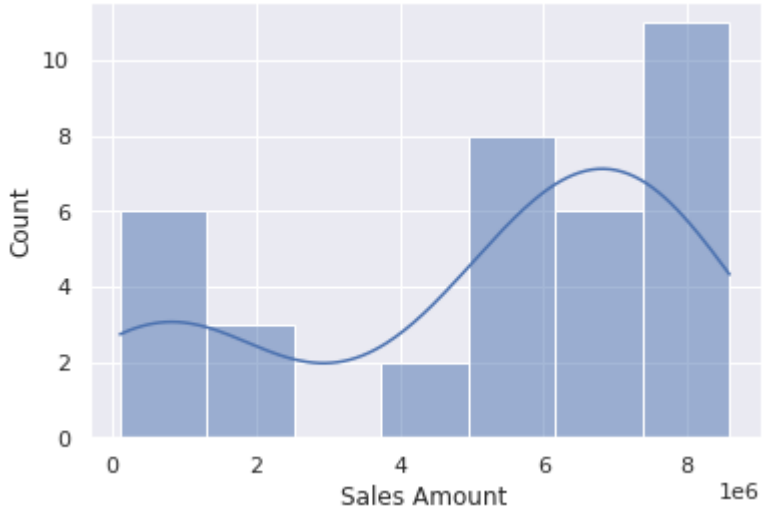

```
sns.catplot(y='Sales Amount', x = 'Invoice_Month', data = yearly_monthwise,palette='tur
            col = 'Invoice_Year',col_wrap=3)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc360be9250>
```




```
sns.histplot(yearly_monthwise['Sales Amount'],kde = True)
```

```
<AxesSubplot:xlabel='Sales Amount', ylabel='Count'>
```



Monthly Record

```
monthly_sales = data02.groupby(['Invoice_Year','Invoice_Month','Invoice_Day']).sum().re
monthly_sales.iloc[:,5:].describe()
```

	Sales Quantity	Sales Amount Based on List Price	Discount Amount	Sales Amount	Sales Margin Amount	Sales Cost Amount	Sales Rep
count	558.00000	5.580000e+02	5.580000e+02	5.580000e+02	558.000000	558.000000	558.000000
mean	5273.84767	5.507407e+05	2.171556e+05	3.335851e+05	139262.409570	194322.662975	16067.224014
std	4780.99002	3.885589e+05	1.819447e+05	2.299789e+05	96929.658407	135882.365942	8029.838008
min	56.00000	1.084864e+04	-2.555238e+05	5.433650e+03	2349.360000	3084.290000	345.000000
25%	2118.00000	2.585411e+05	8.911556e+04	1.573360e+05	63142.357500	92438.910000	10068.000000
50%	3118.00000	4.537218e+05	1.634601e+05	2.688299e+05	113446.860000	150548.000000	14681.500000
75%	6581.50000	7.178838e+05	2.896384e+05	4.696801e+05	195474.320000	267013.720000	19665.750000
max	23710.00000	2.408920e+06	1.029636e+06	1.379283e+06	619085.330000	760197.860000	55866.000000

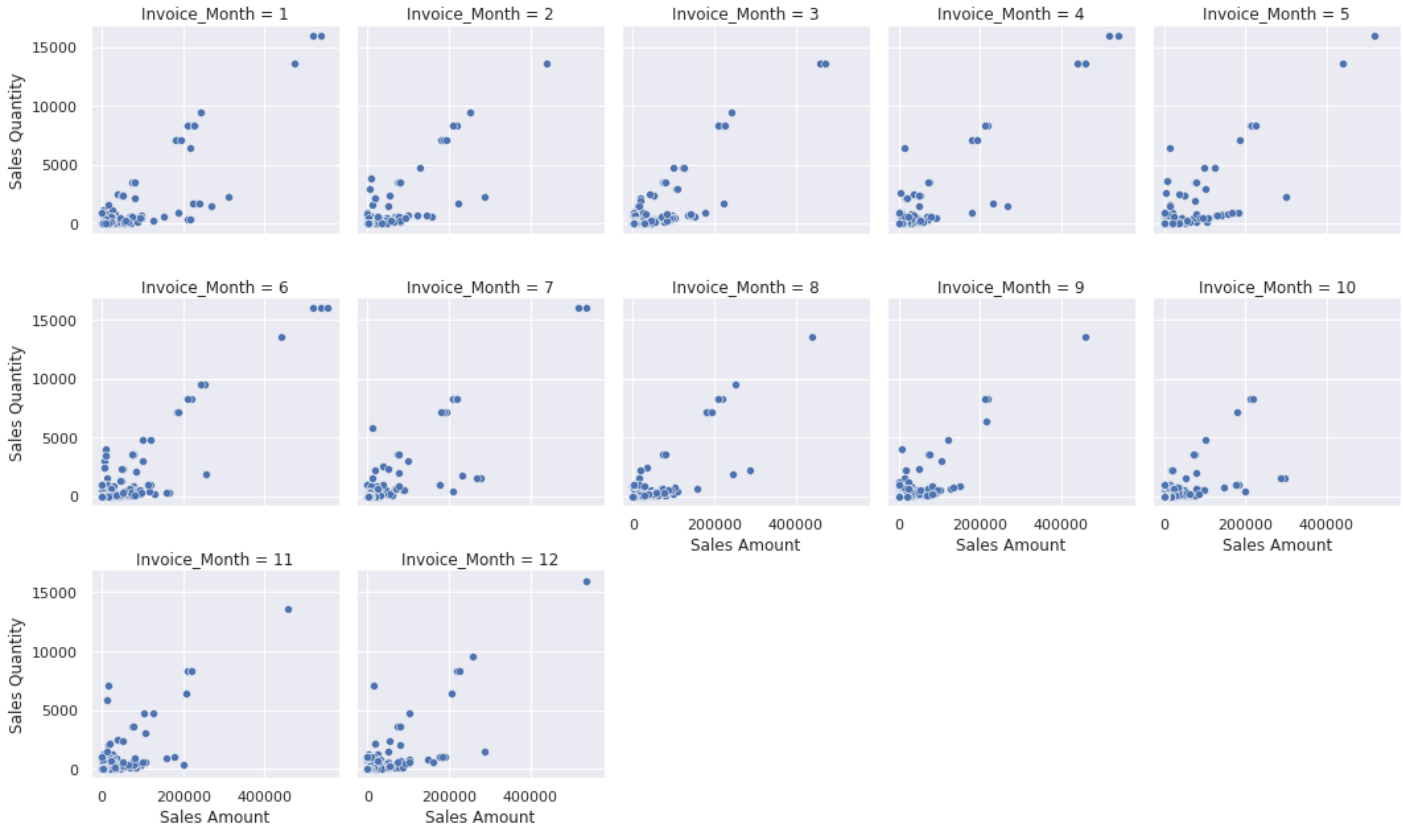
```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-
analysis
```

```
'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'
```

```
sns.relplot(y='Sales Quantity', x ='Sales Amount', data = data02, height = 3, aspect =
            col = 'Invoice_Month', col_wrap = 5, palette = 'muted')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc360a95640>
```




```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y ='Sales Amount', data = data02.query('Invoice_Year == 20
            col = 'Invoice_Month', col_wrap = 4)
plt.ylabel('Sales Amount')
print('Monthly sales trend  in 2017')
```

```
Monthly sales trend  in 2017
```

```
<Figure size 576x1440 with 0 Axes>
```


```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on
```

https://jovian.ai

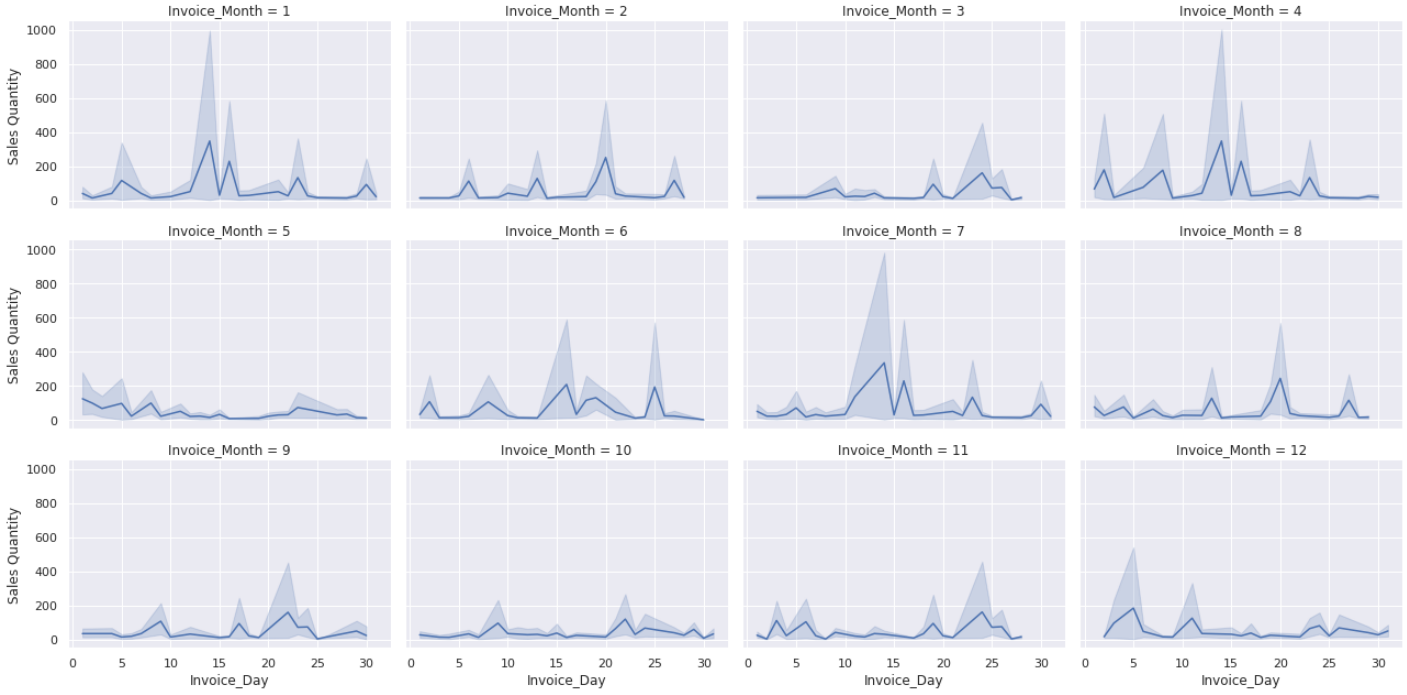
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y ='Sales Quantity', data = data02.query('Invoice_Year ==2
            col = 'Invoice_Month', col_wrap = 4)
plt.ylabel('Sales Quantity')
print('Monthly sales quantity trend in 2017 ')
```

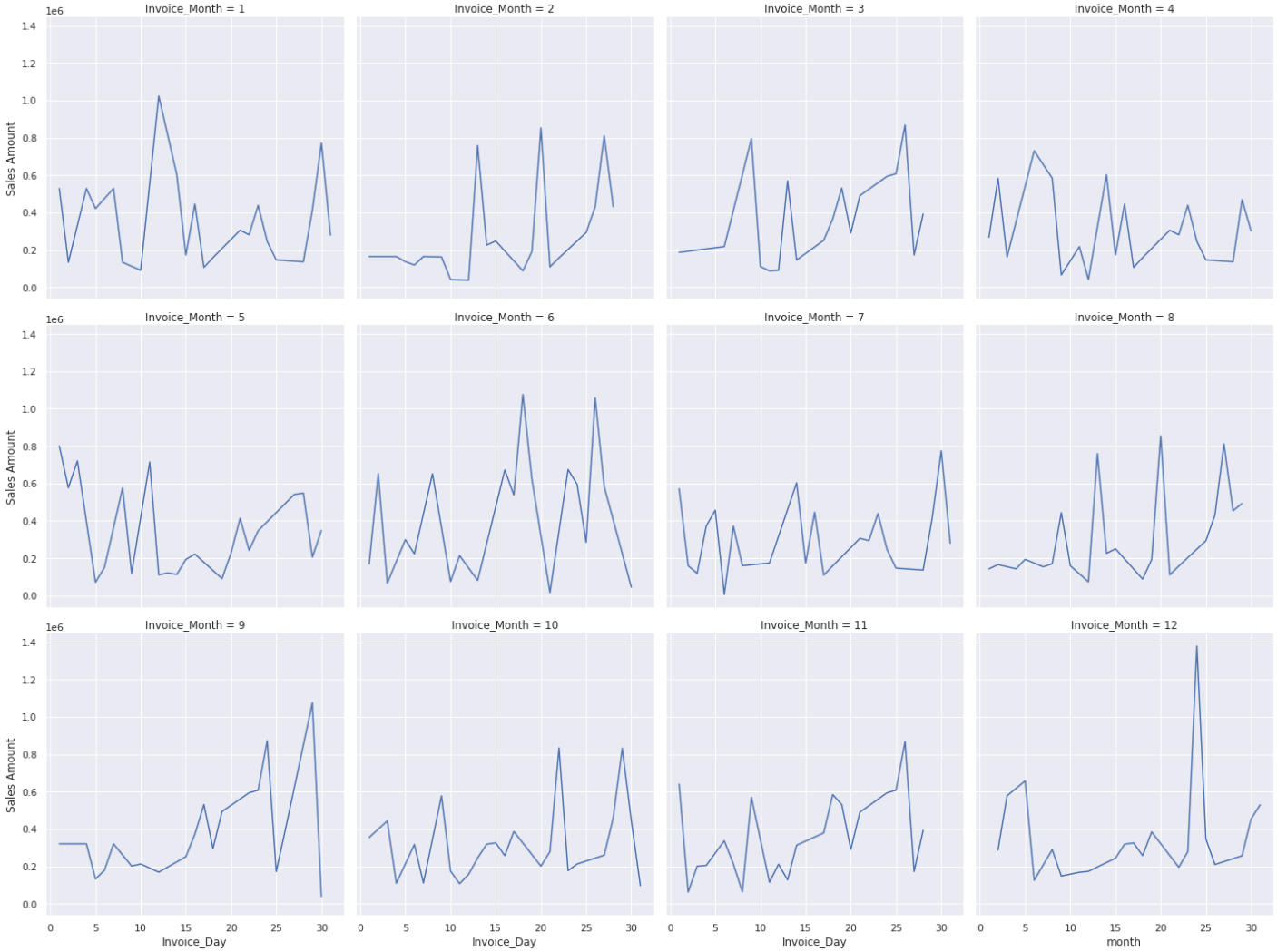
Monthly sales quantity trend in 2017

<Figure size 576x1440 with 0 Axes>



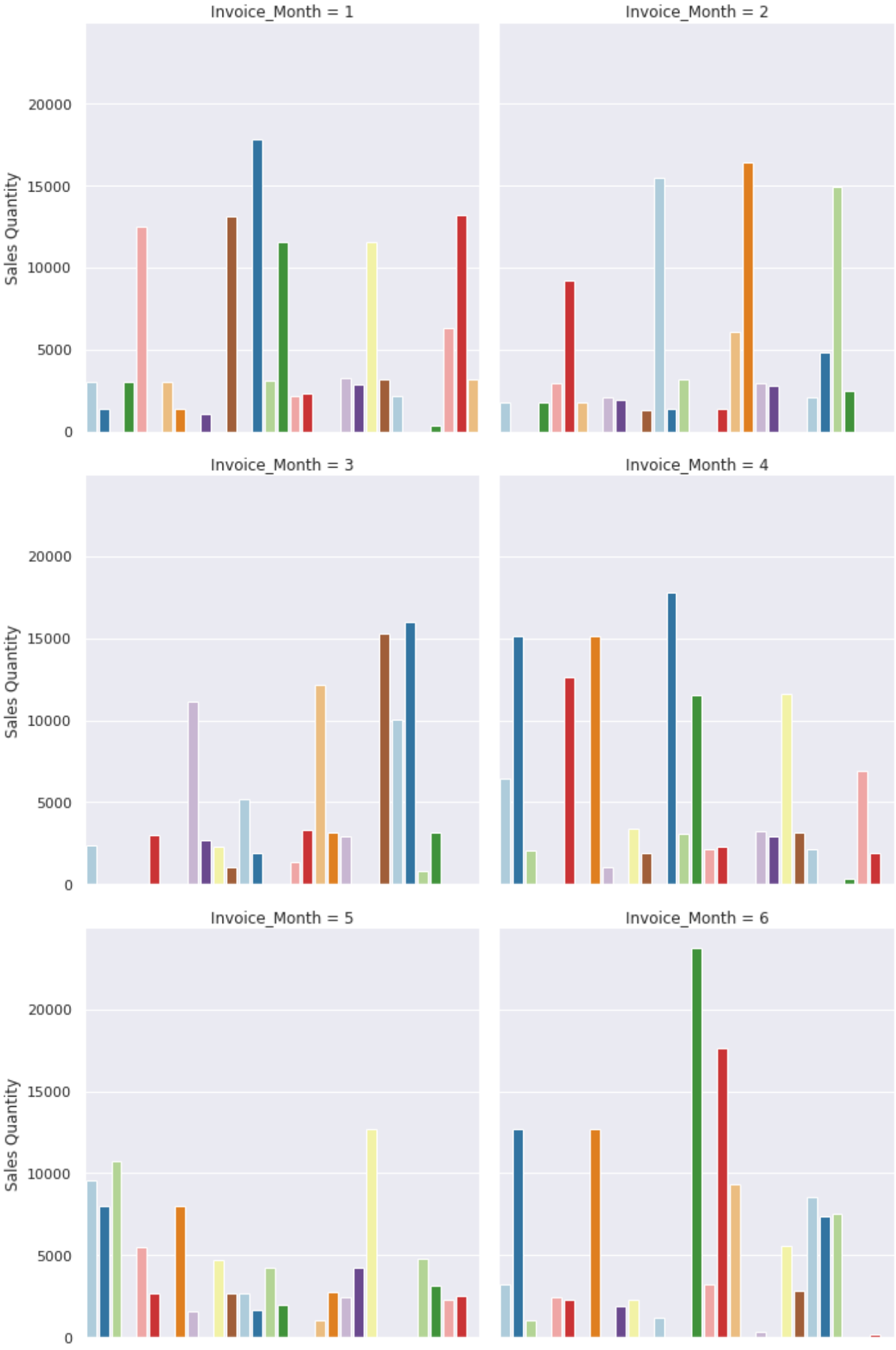

```
sns.relplot(x='Invoice_Day', y ='Sales Amount', data = monthly_sales.query('Invoice_Yea
            kind = 'line', aspect = 1, col='Invoice_Month', col_wrap = 4)
plt.xlabel('month')
plt.ylabel('Sales Amount')
print('Daily total sales trend per month in 2017')
```

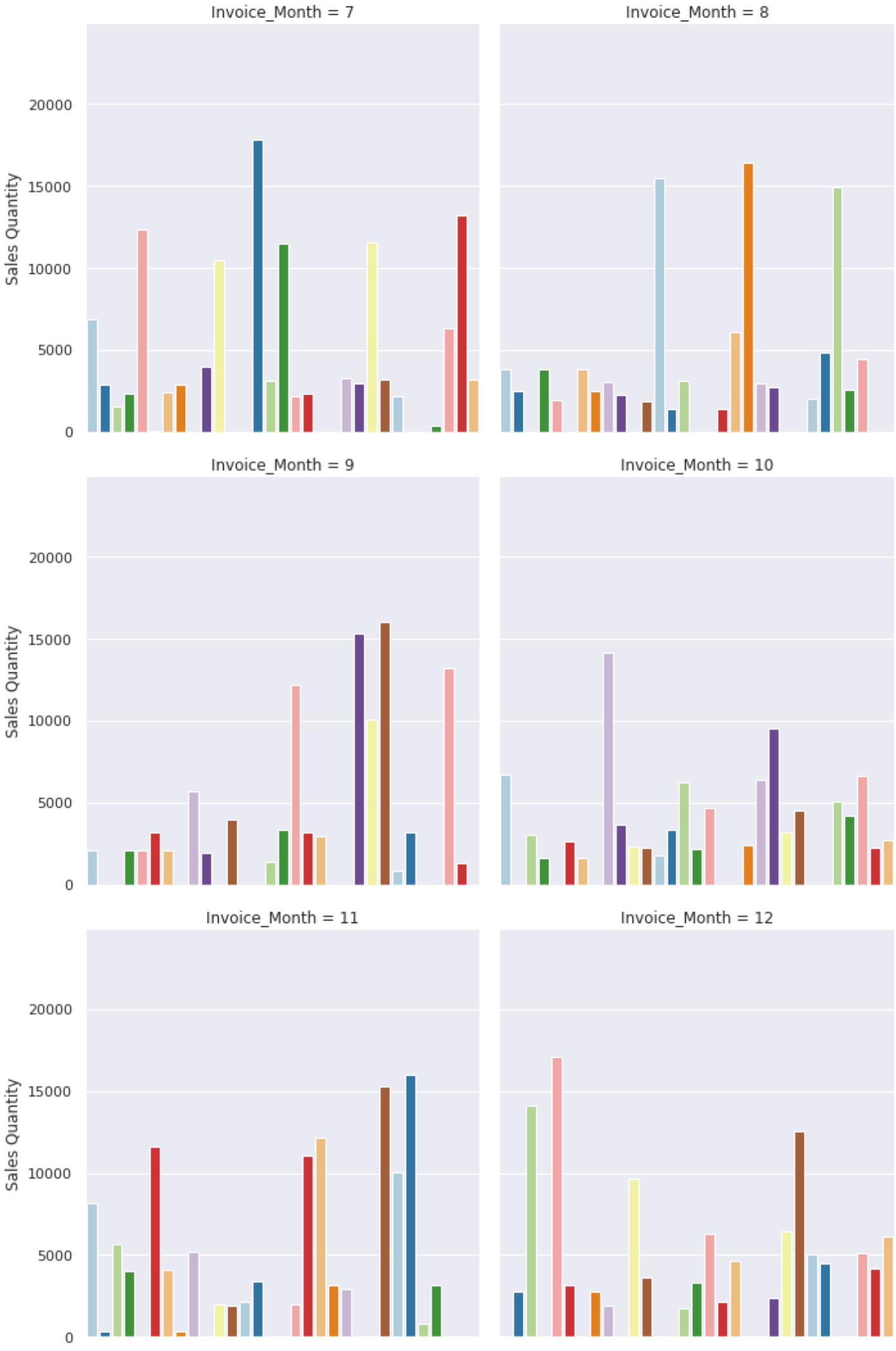
Daily total sales trend per month in 2017

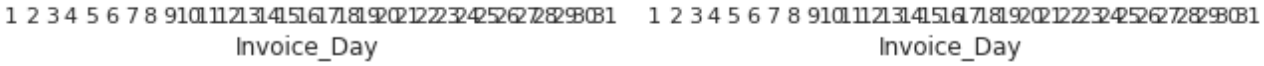



```
sns.catplot(y = 'Sales Quantity', x = 'Invoice_Day', data = monthly_sales[monthly_sales
           palette = 'Paired',kind = "bar", col = 'Invoice_Month', col_wrap = 2)
print('Monthly quantity purachased trend in 2017')
```

```
Monthly quantity purachased trend in 2017
```


Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Sales Quantity
0
5000
10000
15000
20000

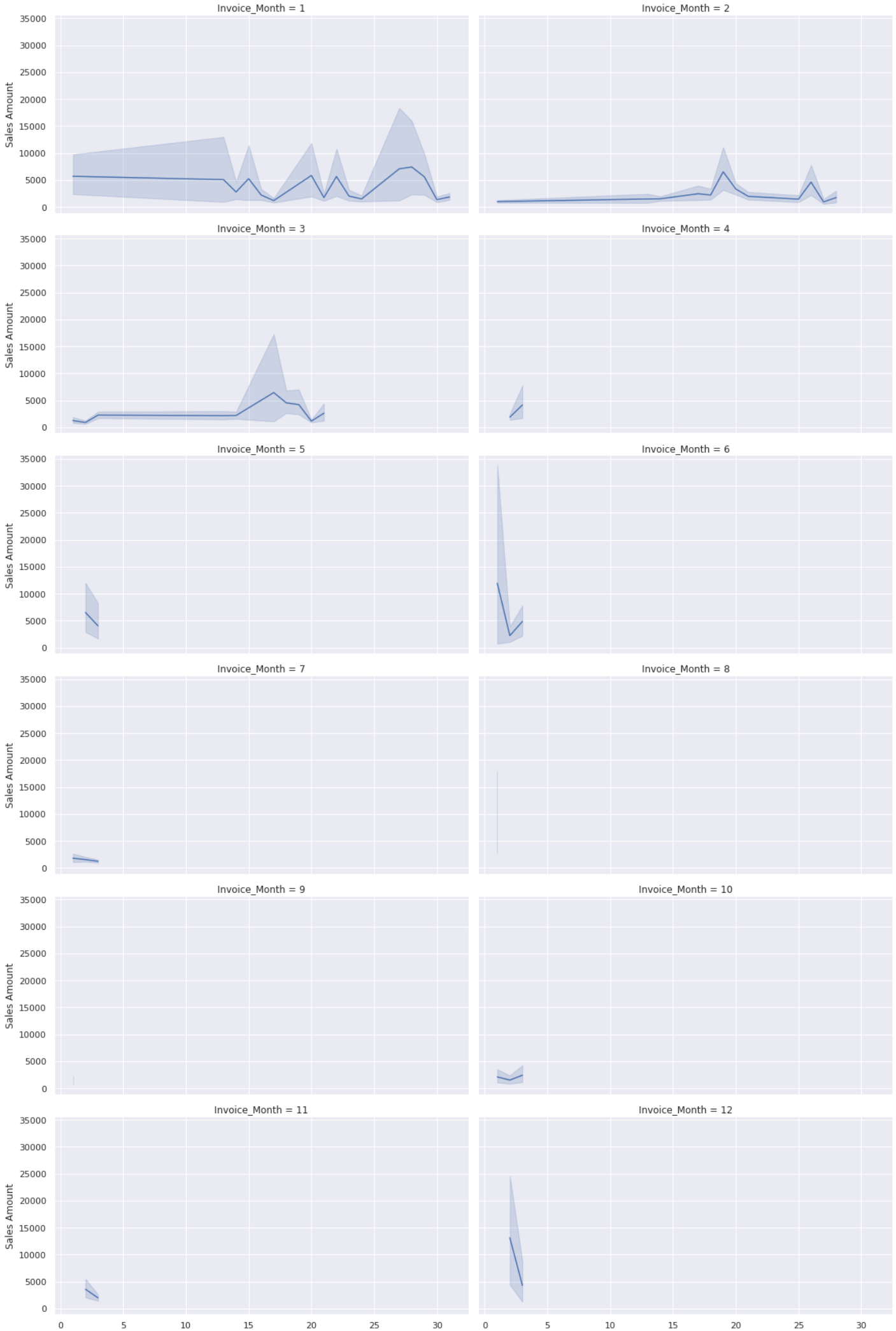

Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Quantity
0
5000
10000
15000
20000




```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Amount', data = data02.query('Invoice_Year == 2
            kind = 'line', col = 'Invoice_Month', col_wrap = 2, height = 4, aspect = 2)
plt.ylabel('Sales Amount')
print('Monthly Sales Trend in 2018')
```

```
Monthly Sales Trend in 2018
```

```
<Figure size 576x1440 with 0 Axes>
```

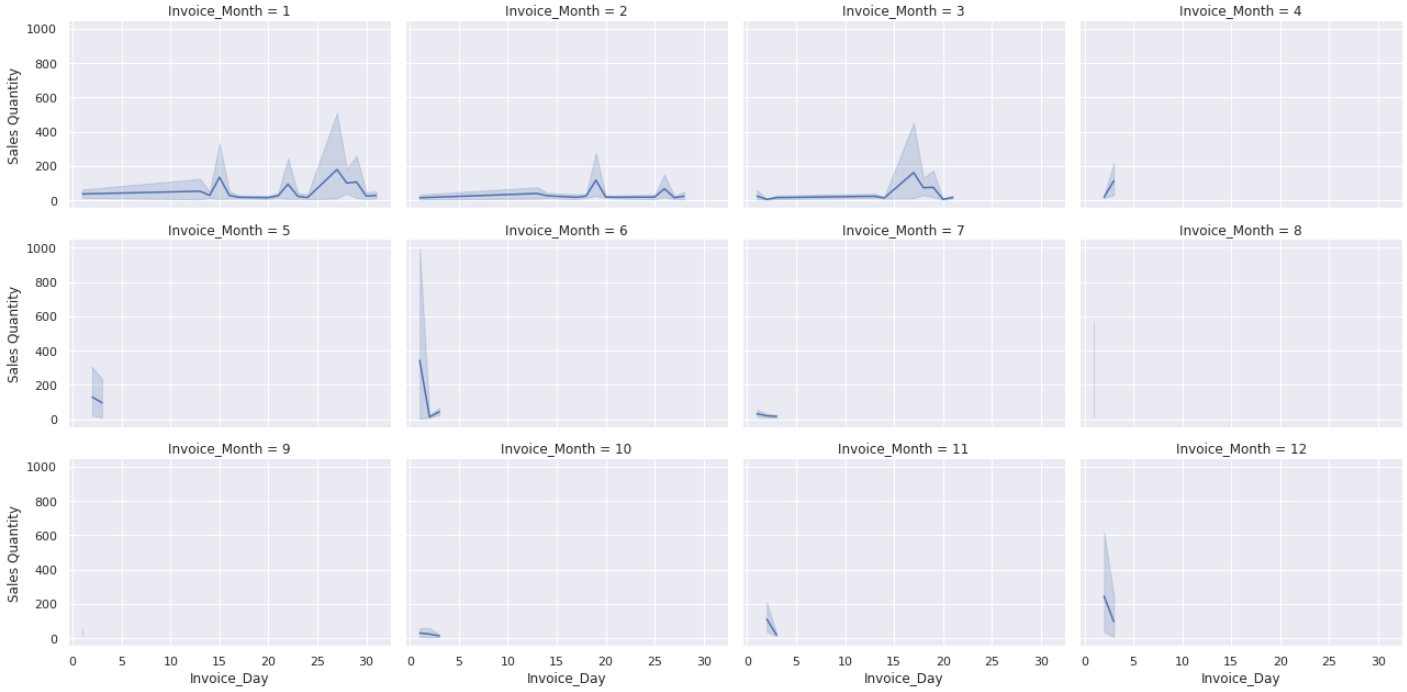

Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Amount
0
5000
10000
15000
20000
25000
30000
35000
0
5
10
15
20
25
30

Invoice_Day Invoice_Day

```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Quantity', data = data02.query('Invoice_Year ==
            kind = 'line', col = 'Invoice_Month', col_wrap = 4, height = 3, aspect = 1.5
plt.ylabel('Sales Amount')
print('Monthly sales Quantity Trend in 2018')
```

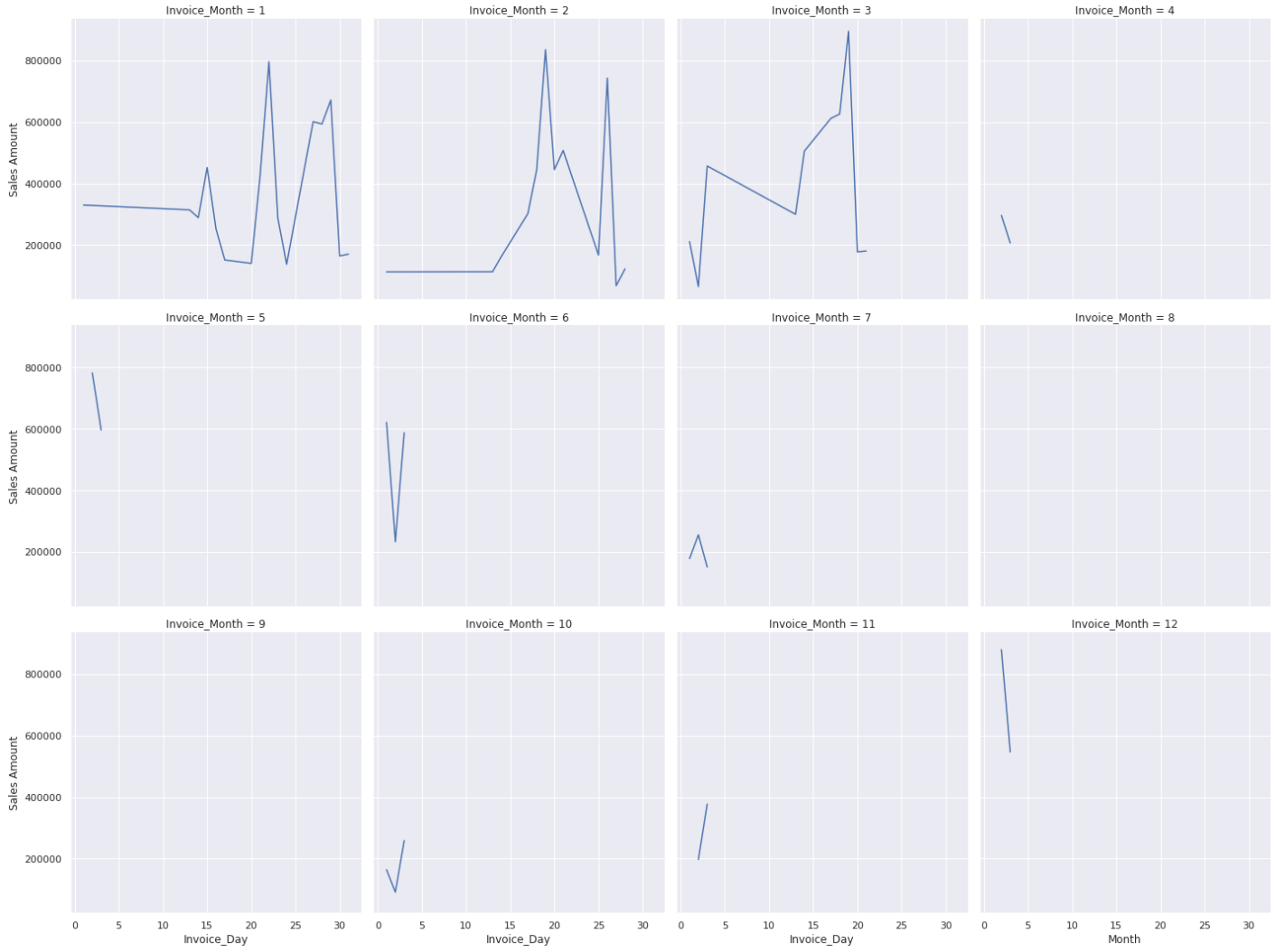
```
Monthly sales Quantity Trend in 2018
```

```
<Figure size 576x1440 with 0 Axes>
```



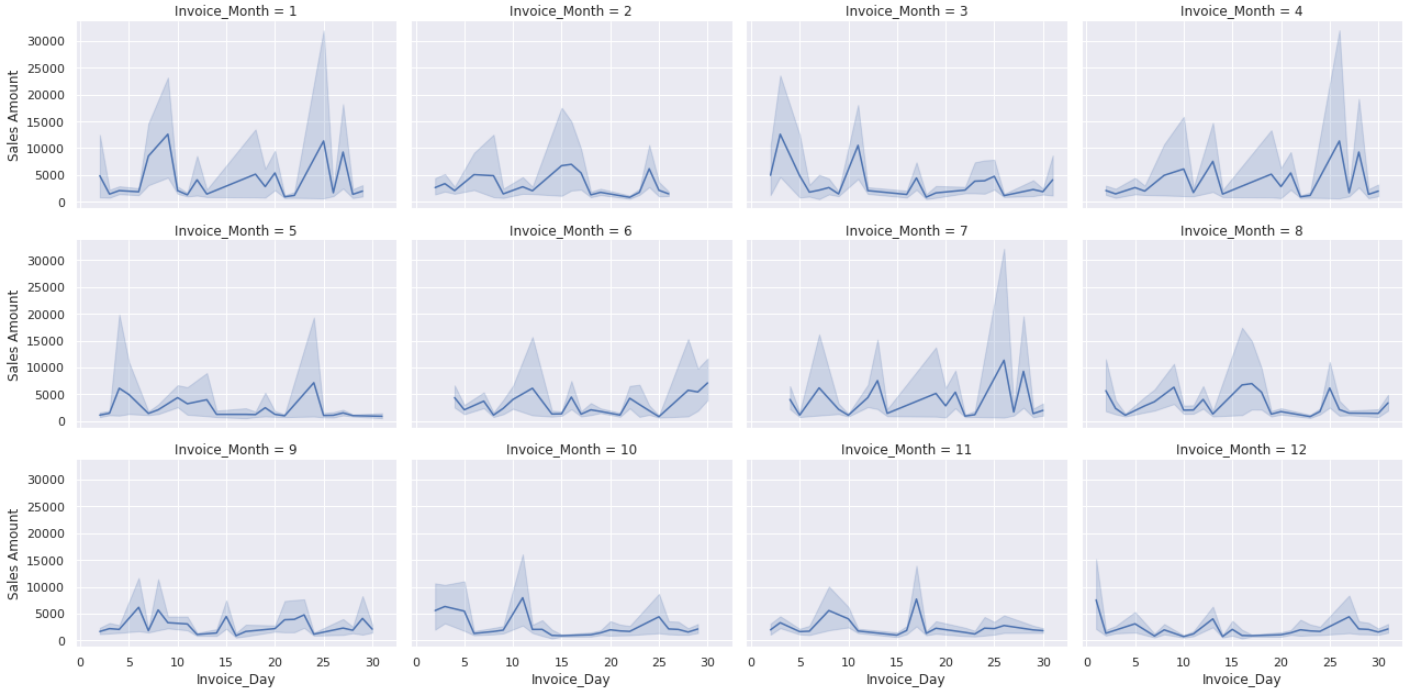

```
sns.relplot(x='Invoice_Day', y = 'Sales Amount', data = monthly_sales.query('Invoice_Ye
            kind = 'line', aspect = 1, col = 'Invoice_Month', col_wrap = 4)
plt.xlabel('Month')
plt.ylabel('Sales Amount')
print('Daily Total Sales Trend per Month in 2018')
```

```
Daily Total Sales Trend per Month in 2018
```



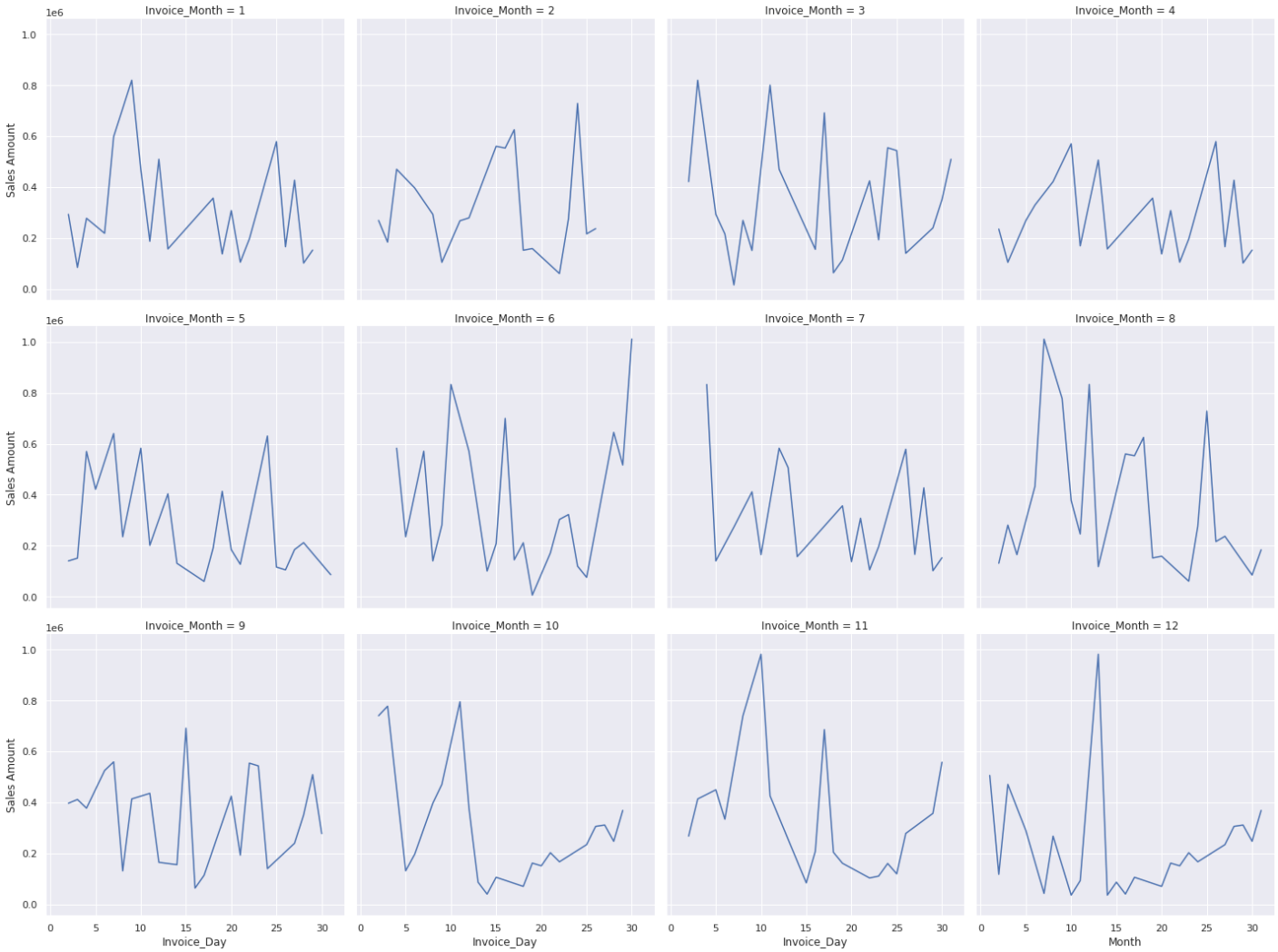

```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Amount', data = data02.query('Invoice_Year ==20
            kind= 'line', col= 'Invoice_Month', col_wrap = 4, height = 3, aspect = 1.5)
plt.ylabel('Sales Amount')
print('Monthly sales trend in 2019')
```

```
Monthly sales trend in 2019
<Figure size 576x1440 with 0 Axes>
```




```
sns.relplot(x='Invoice_Day', y = 'Sales Amount',data = monthly_sales.query('Invoice_Yea
            kind = 'line',aspect = 1, col = 'Invoice_Month', col_wrap = 4)
plt.xlabel('Month')
plt.ylabel('Sales Amount')
print('Daily total sales trend per month in 2019')
```

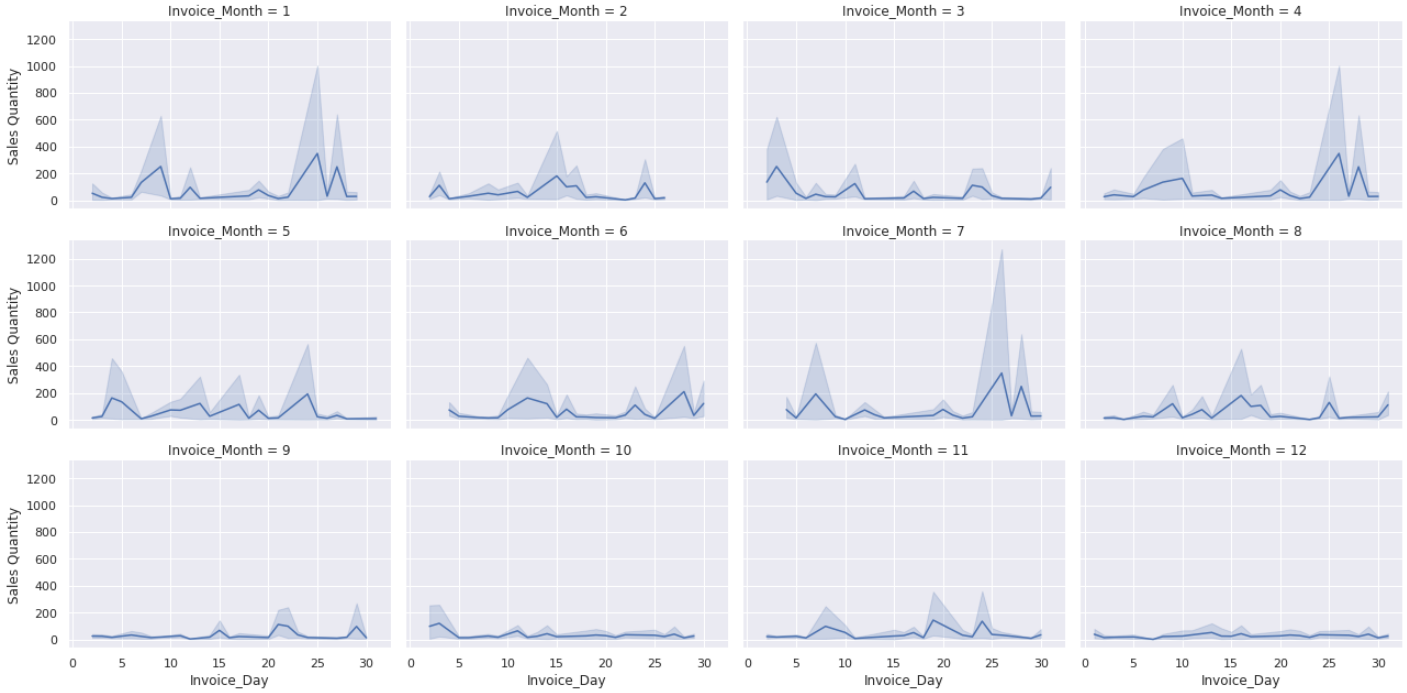
Daily total sales trend per month in 2019



```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Quantity', data = data02.query('Invoice_Year==2
           kind ='line',col = 'Invoice_Month', col_wrap = 4, height = 3, aspect = 1.5)
plt.ylabel('Sales Amount')
print('Monthly sales quantity trend in 2019')
```

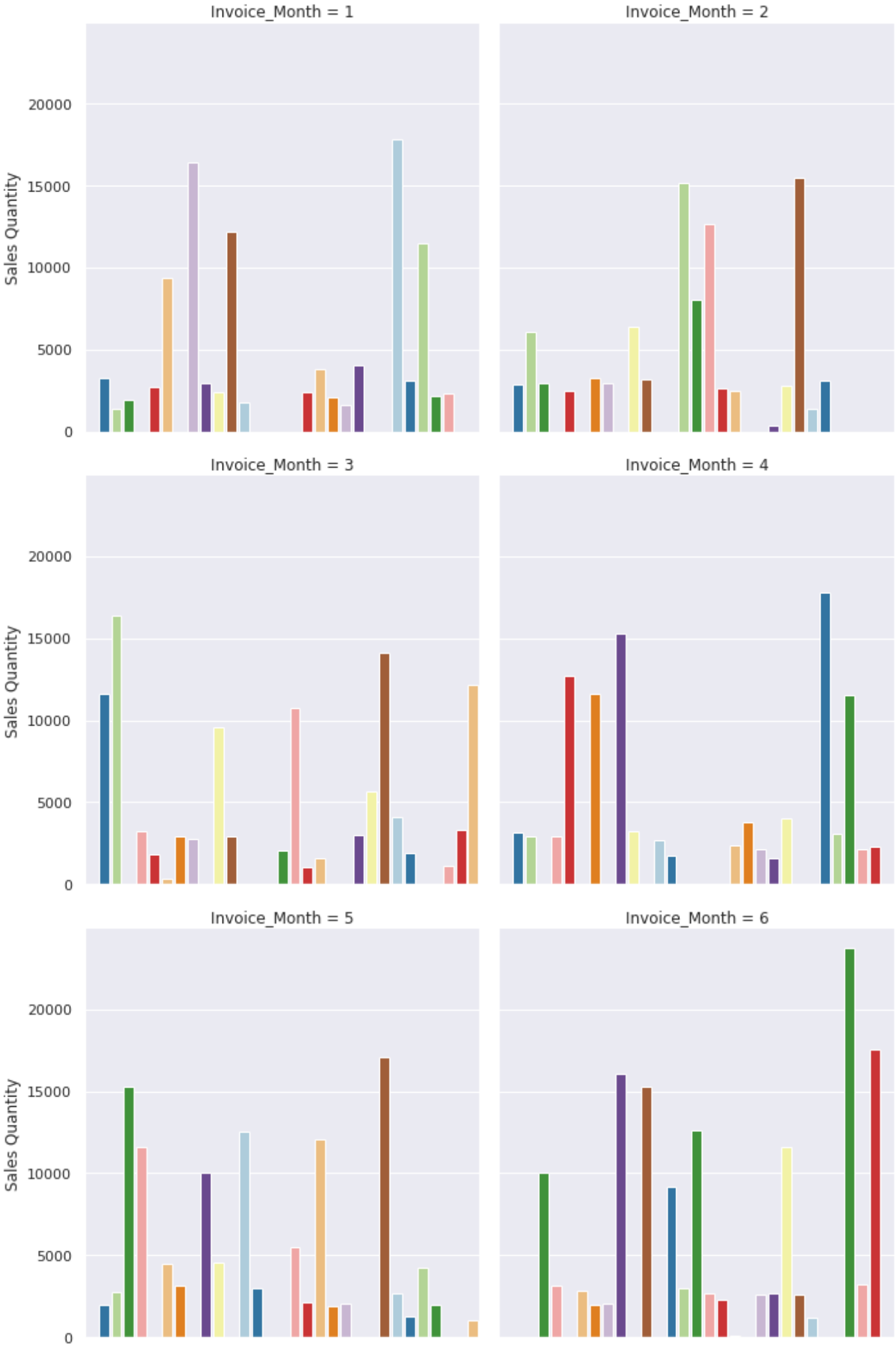
```
Monthly sales quantity trend in 2019
```

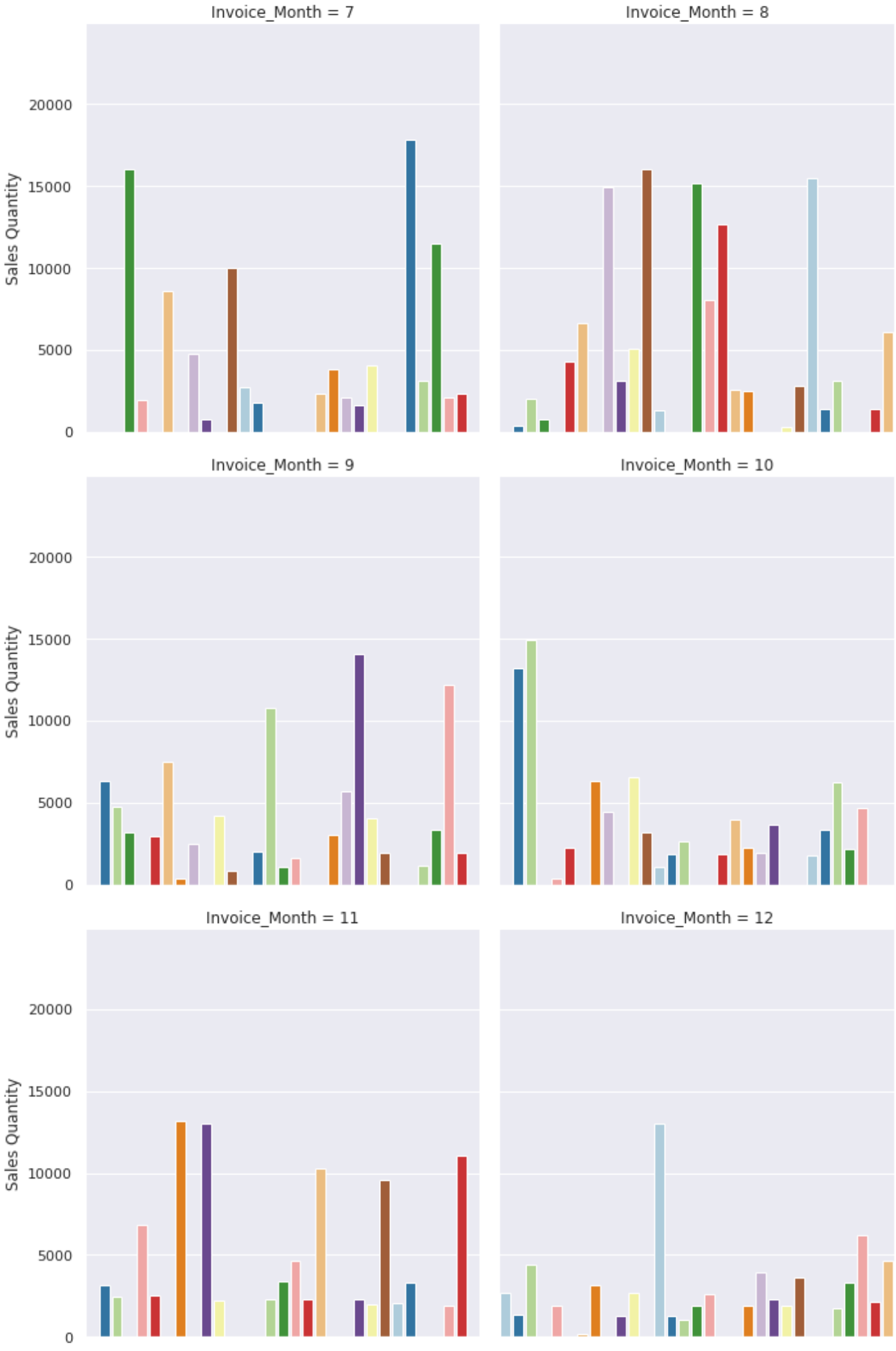
```
<Figure size 576x1440 with 0 Axes>
```

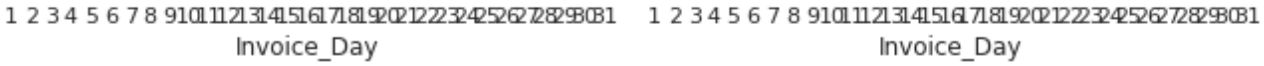



```
sns.catplot(y= 'Sales Quantity', x = 'Invoice_Day', data= monthly_sales[monthly_sales['
            palette ='Paired', kind = 'bar', col = 'Invoice_Month', col_wrap = 2)
print('Monthly quantity purchased trend in 2019')
```

```
Monthly quantity purchased trend in 2019
```


Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Sales Quantity
0
5000
10000
15000
20000


Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Quantity
0
5000
10000
15000
20000



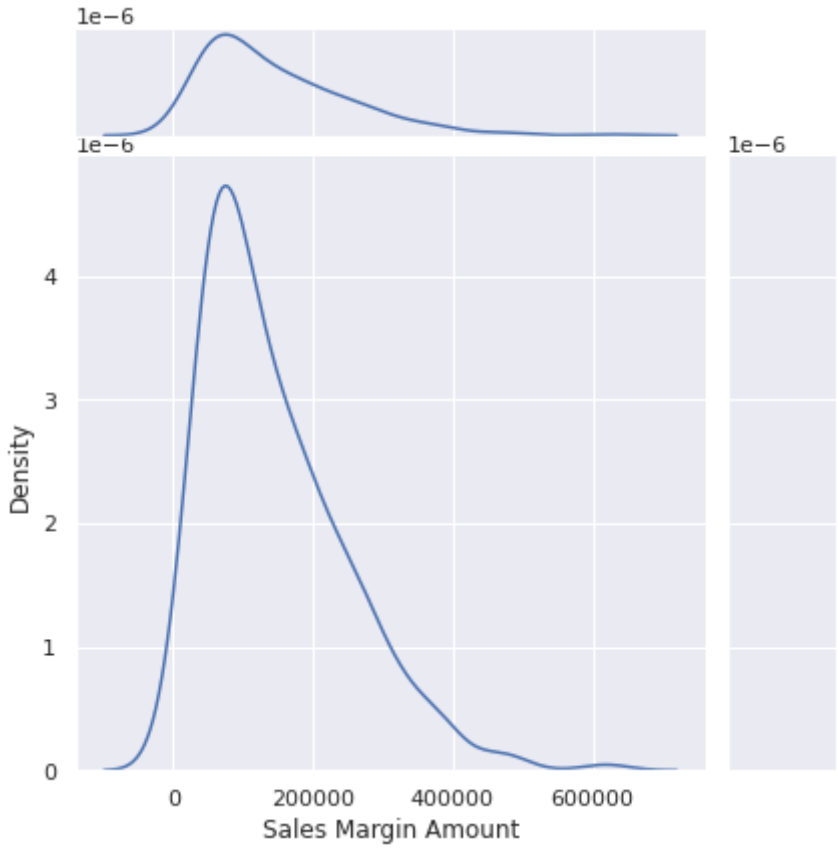

```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-
analysis
```

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

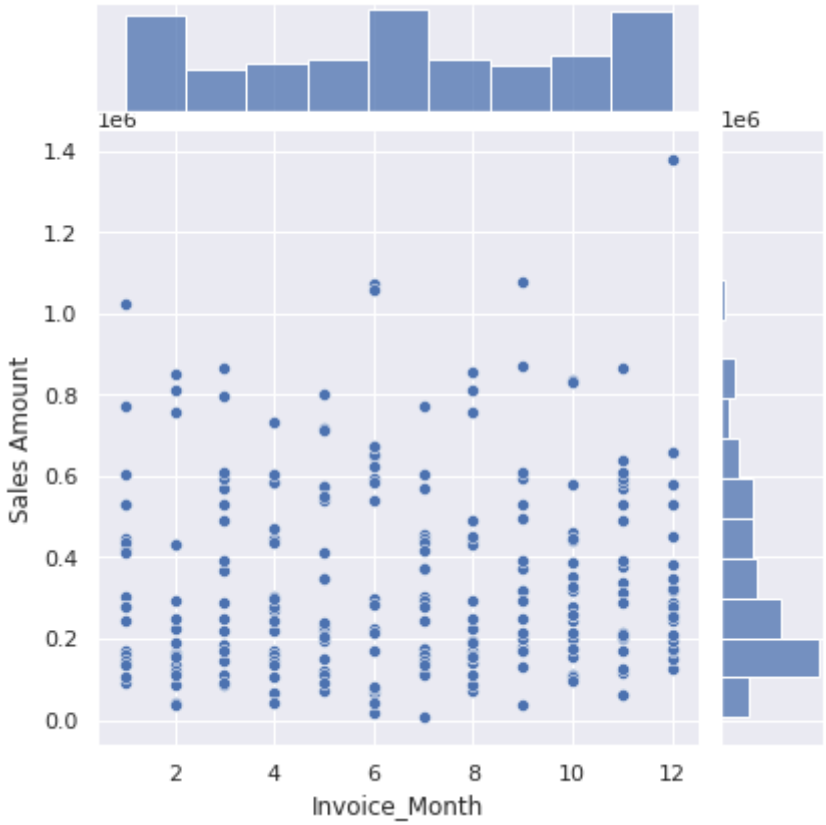
```
 sns.jointplot(x='Sales Margin Amount', data = monthly_sales[monthly_sales['Invoice_Yea
```

<seaborn.axisgrid.JointGrid at 0x7fc35b181c70>



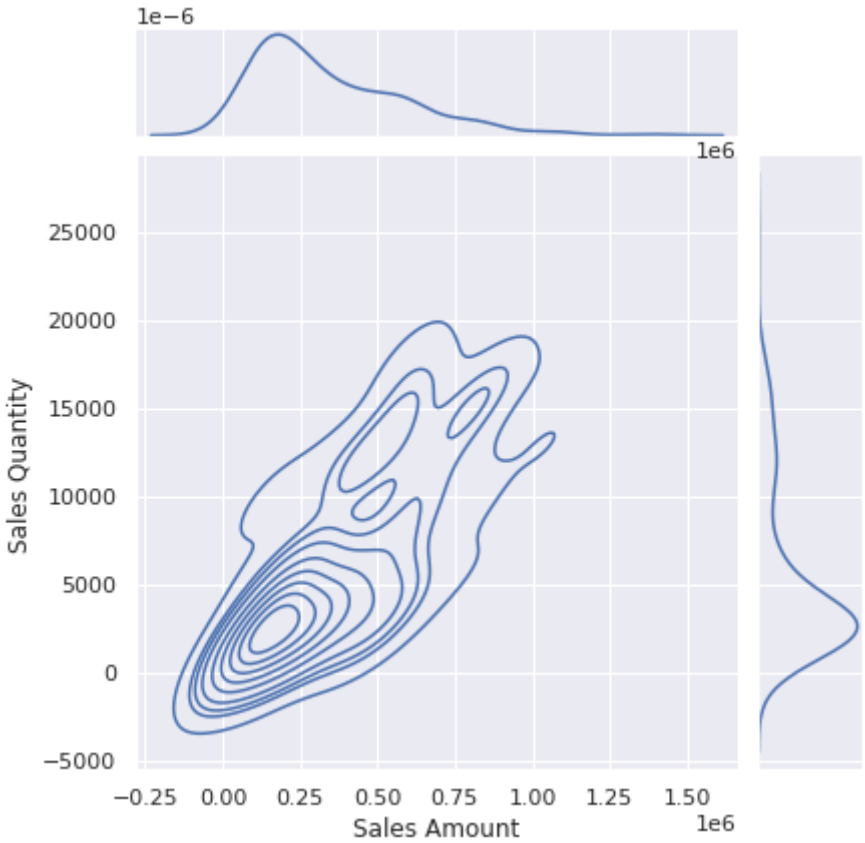

```
sns.jointplot(x='Invoice_Month', y = 'Sales Amount', data = monthly_sales[monthly_sales
```

<seaborn.axisgrid.JointGrid at 0x7fc35bbe6910>



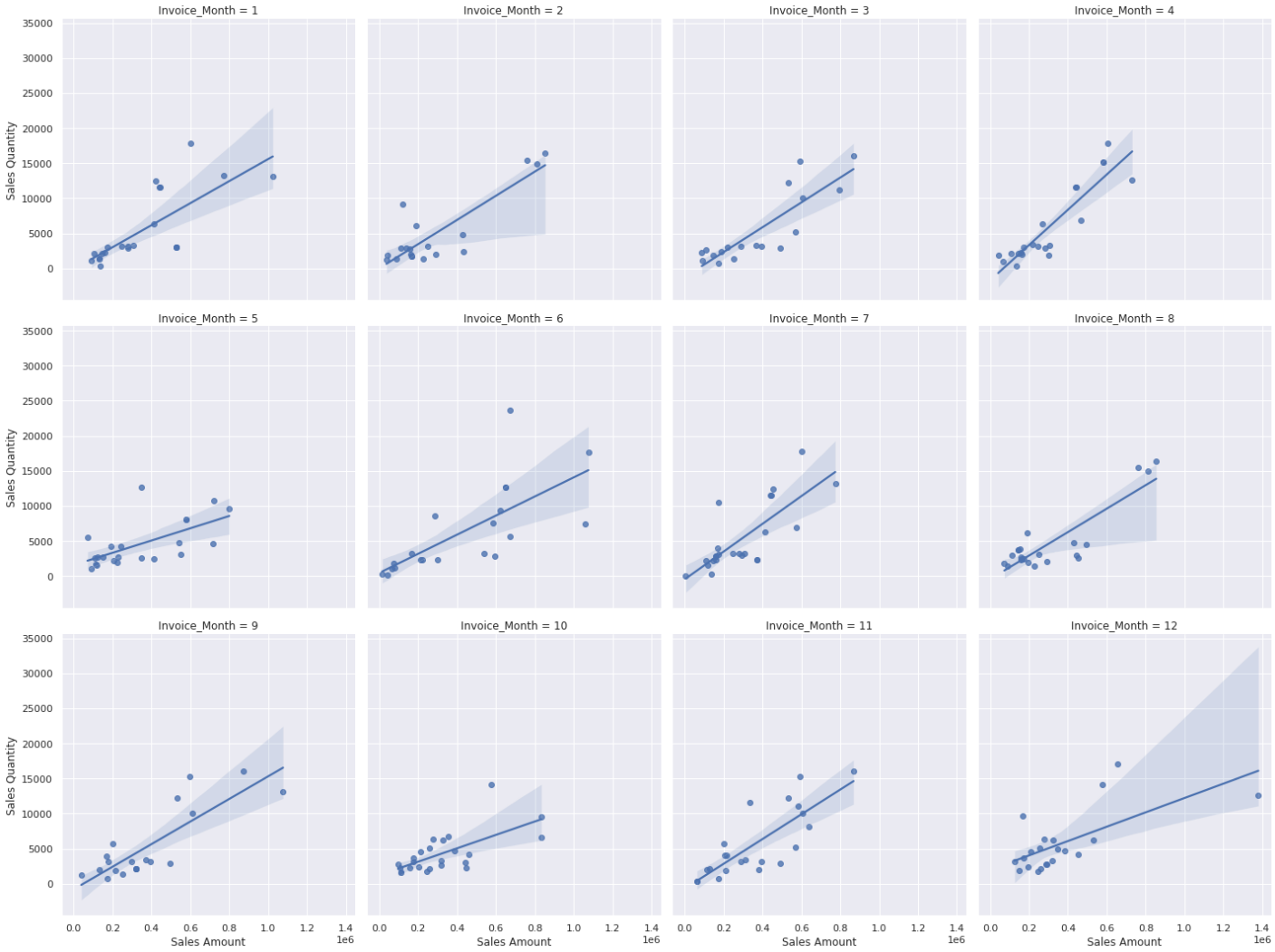

```
sns.jointplot(y='Sales Quantity', x = 'Sales Amount',
              data = monthly_sales[monthly_sales['Invoice_Year']== 2017], kind = 'kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc35ba978e0>
```



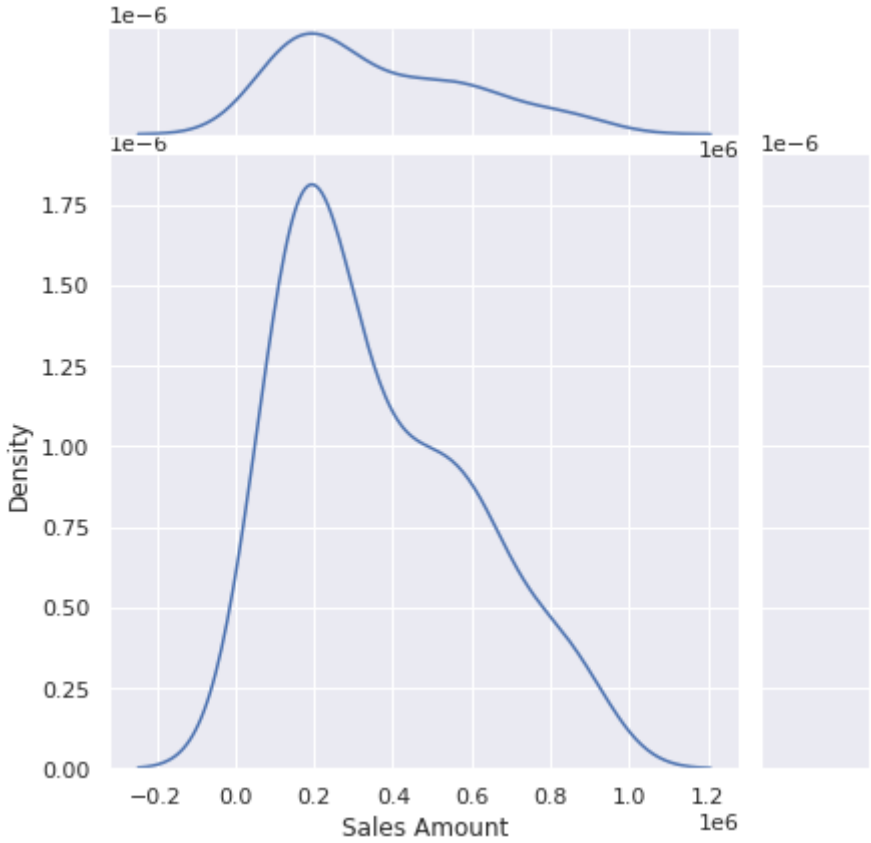

```
sns.lmplot(x='Sales Amount', y = 'Sales Quantity', data = monthly_sales[monthly_sales['
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc35bb00580>
```



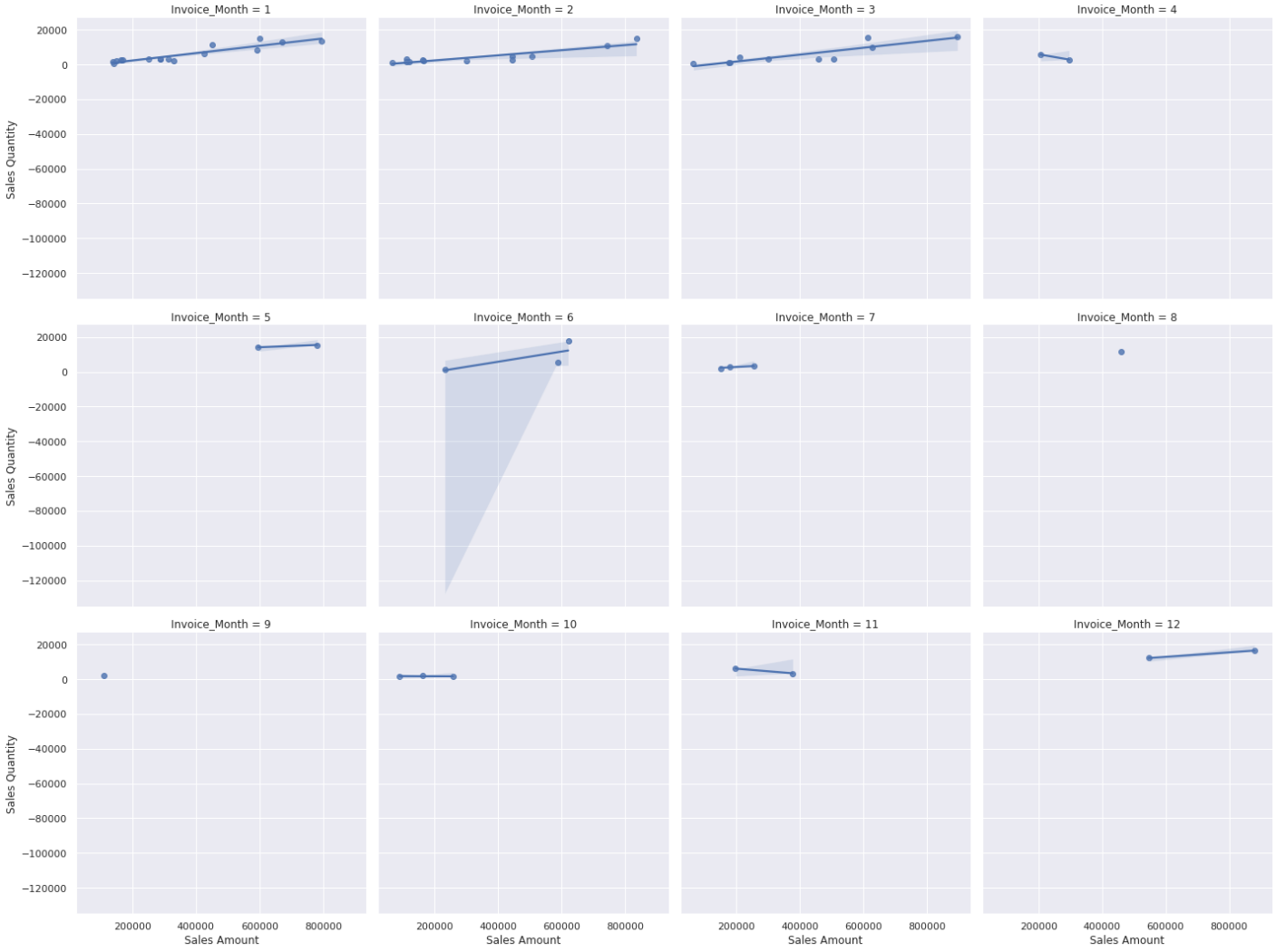

```
sns.jointplot(x='Sales Amount', data = monthly_sales[monthly_sales["Invoice_Year"] == 2
```

```
<seaborn.axisgrid.JointGrid at 0x7fc35b3b8100>
```



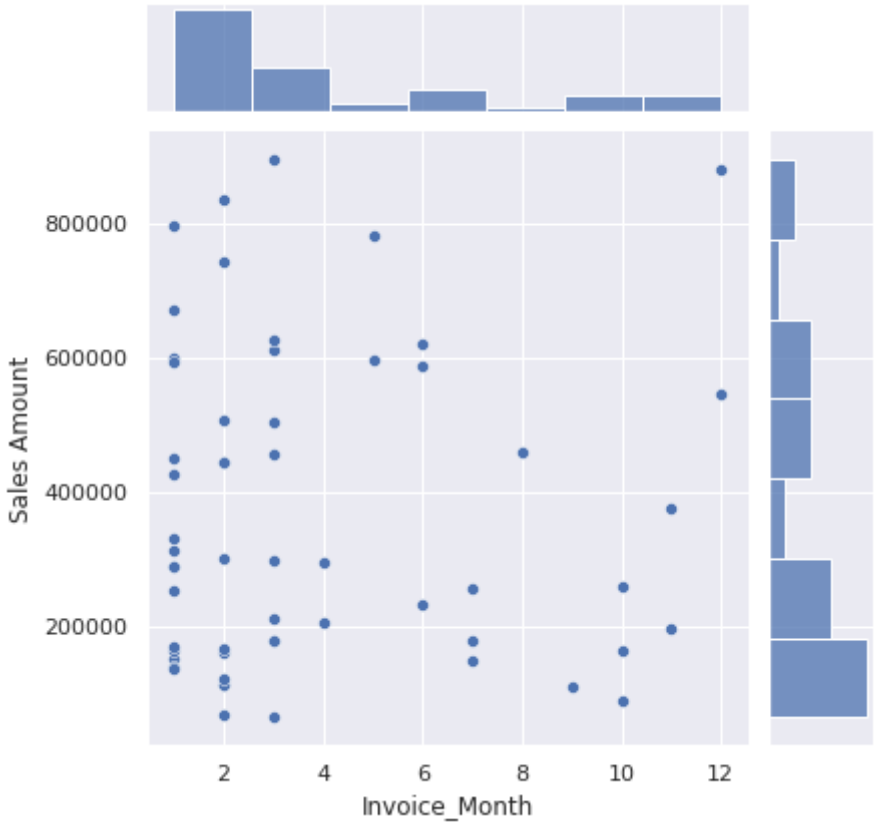
```
sns.lmplot(x='Sales Amount', y = 'Sales Quantity',
           data = monthly_sales[monthly_sales['Invoice_Year']== 2018], col = 'Invoice_Mo
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc360151d90>
```



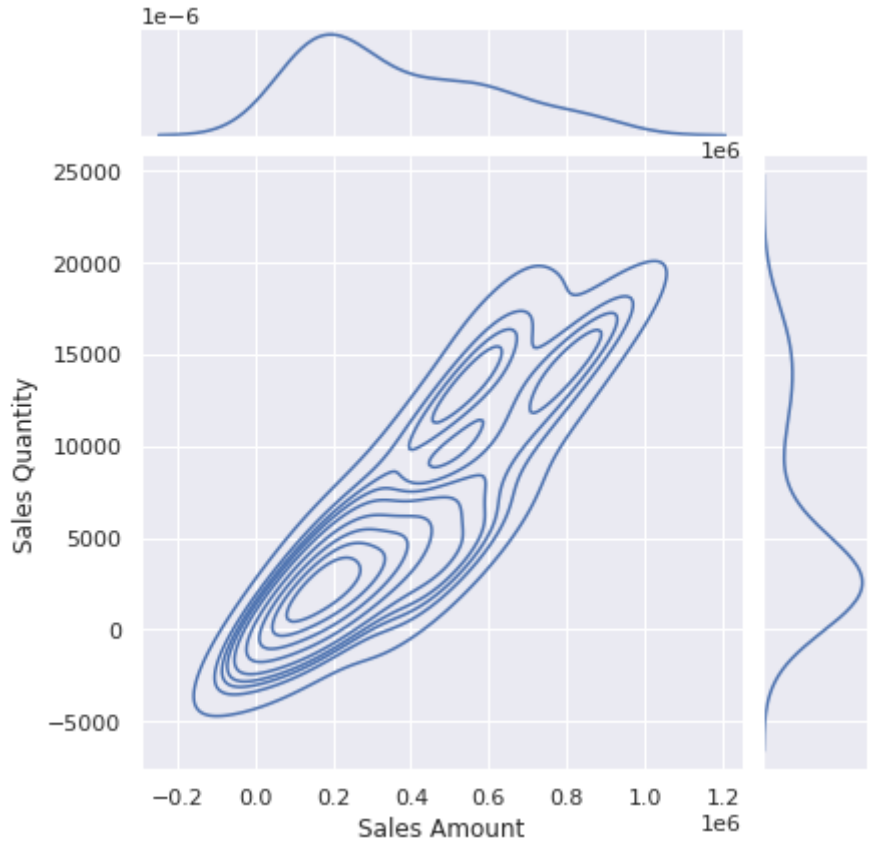

```
sns.jointplot(x='Invoice_Month', y = 'Sales Amount',data = monthly_sales[monthly_sales[
```

```
<seaborn.axisgrid.JointGrid at 0x7fc3603e64c0>
```



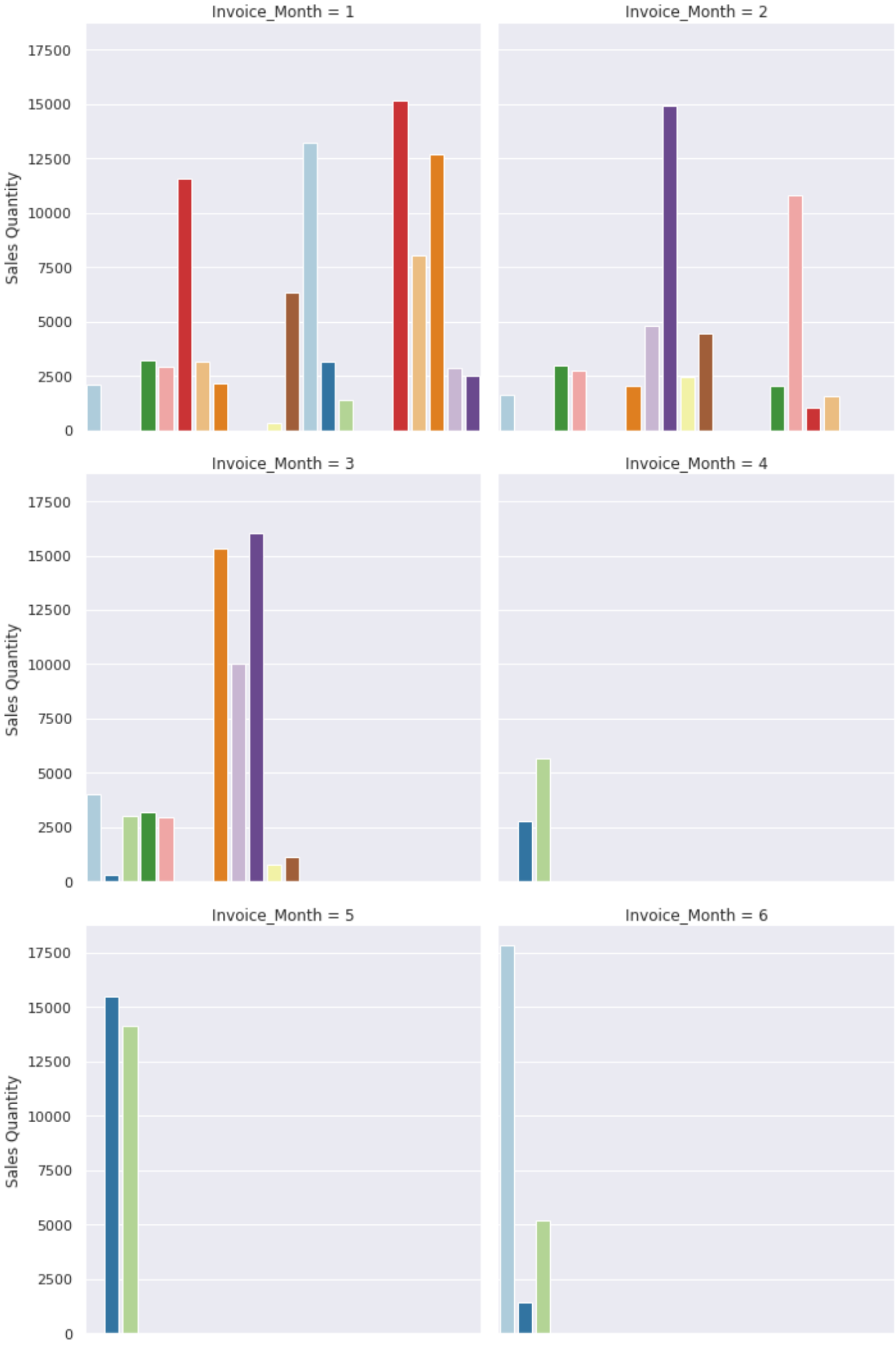

```
sns.jointplot(x='Sales Amount', y = 'Sales Quantity',
              data = monthly_sales[monthly_sales['Invoice_Year']==2018],kind = 'kde')
```

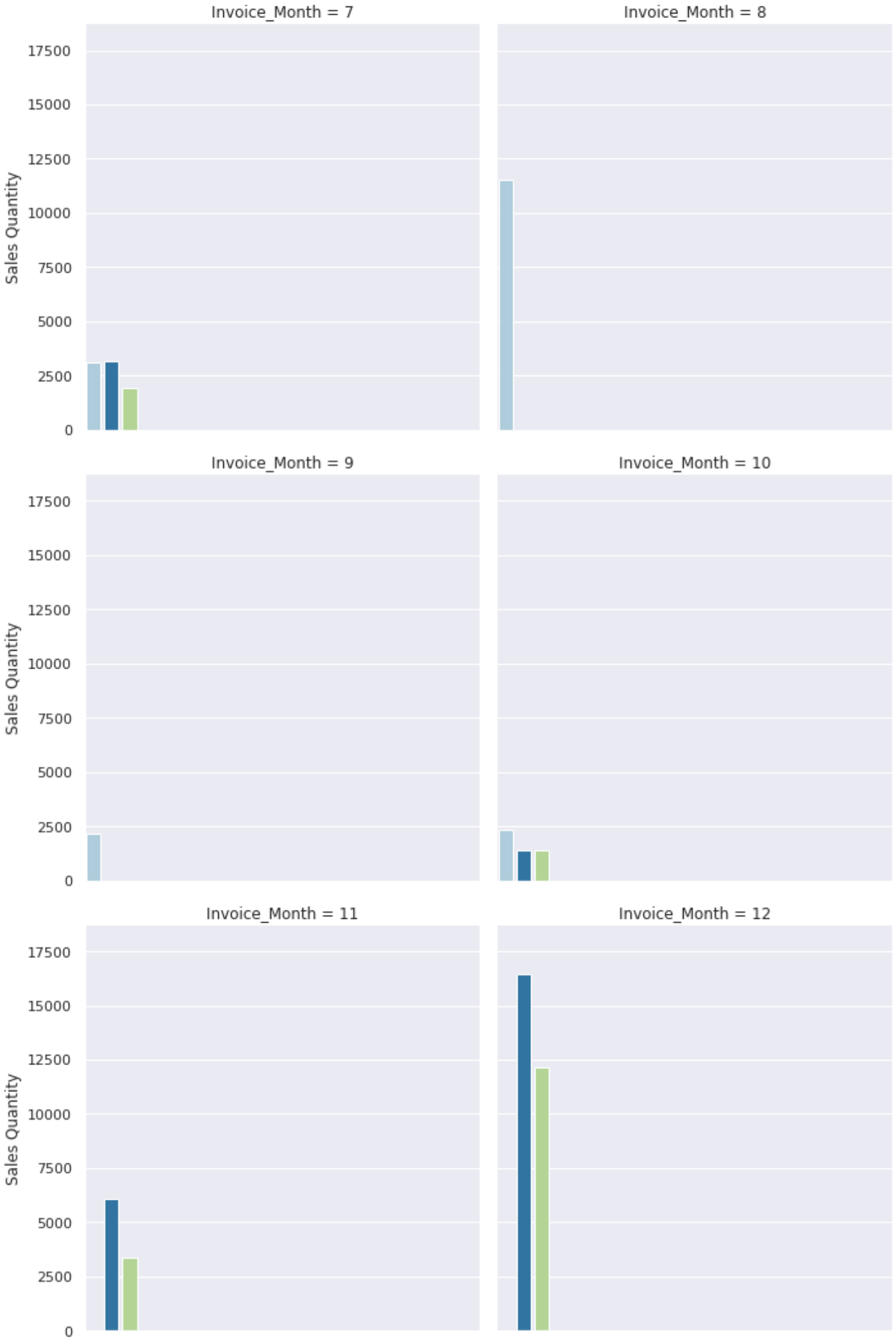
```
<seaborn.axisgrid.JointGrid at 0x7fc35bbeed90>
```

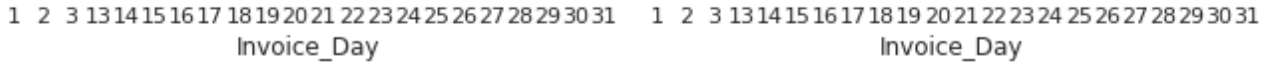



```
sns.catplot(y='Sales Quantity', x = 'Invoice_Day', data = monthly_sales[monthly_sales['
            palette = 'Paired', kind = 'bar', col ='Invoice_Month', col_wrap=2)
print('Monthly quantity purchased trend in 2018')
```

Monthly quantity purchased trend in 2018


Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Sales Quantity
17500
15000
12500
10000
7500
5000
2500
0


Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Quantity
17500
15000
12500
10000
7500
5000
2500
0



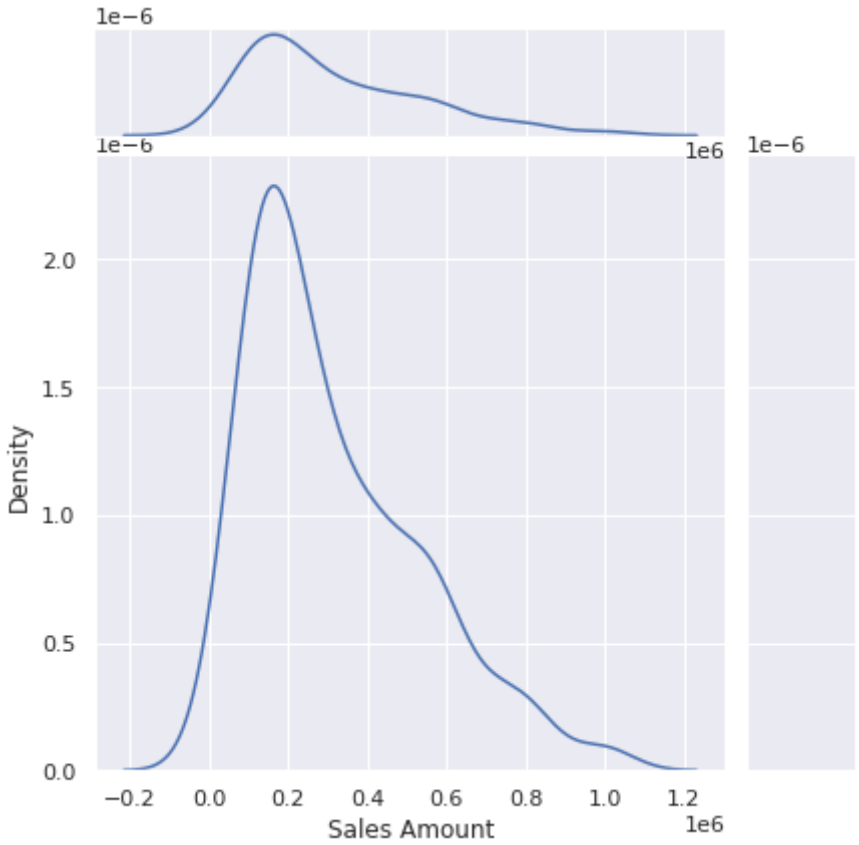

```
jovian.commit()
```

```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-
analysis
```

```
'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'
```

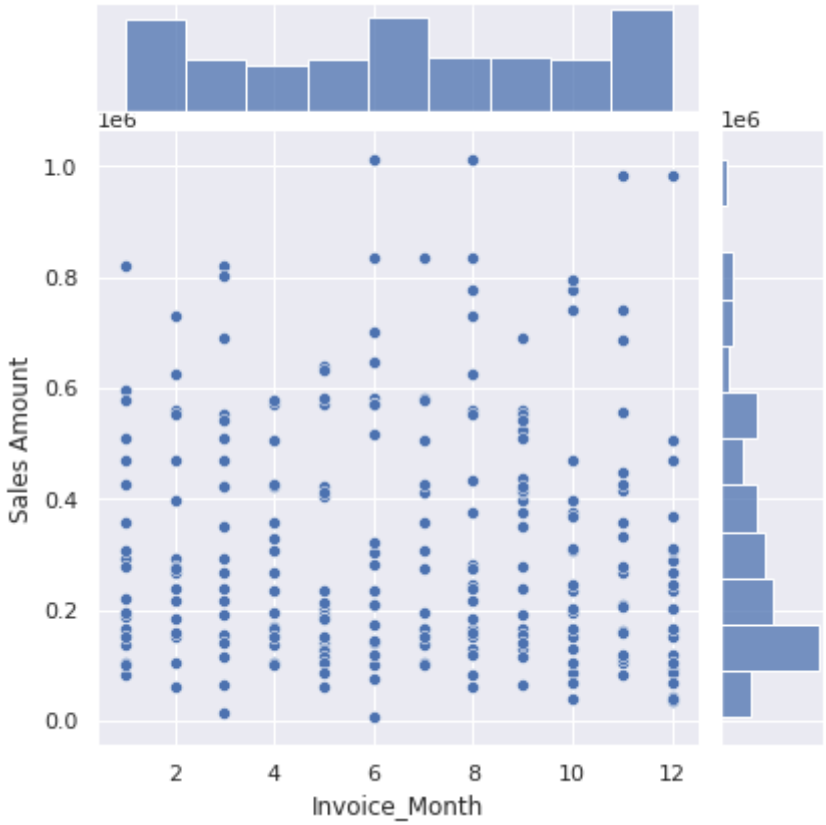
```
sns.jointplot(x='Sales Amount', data = monthly_sales[monthly_sales['Invoice_Year']==201
```

```
<seaborn.axisgrid.JointGrid at 0x7fc35aae9670>
```



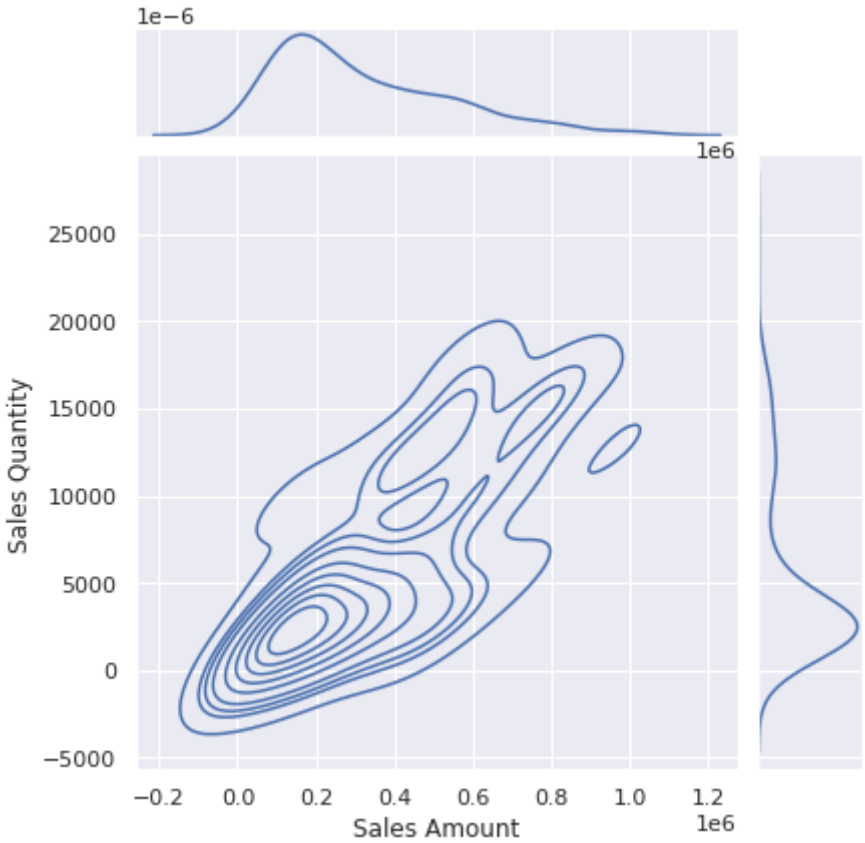

```
sns.jointplot(x='Invoice_Month', y = 'Sales Amount', data = monthly_sales[monthly_sales
```

```
<seaborn.axisgrid.JointGrid at 0x7fc35a945340>
```



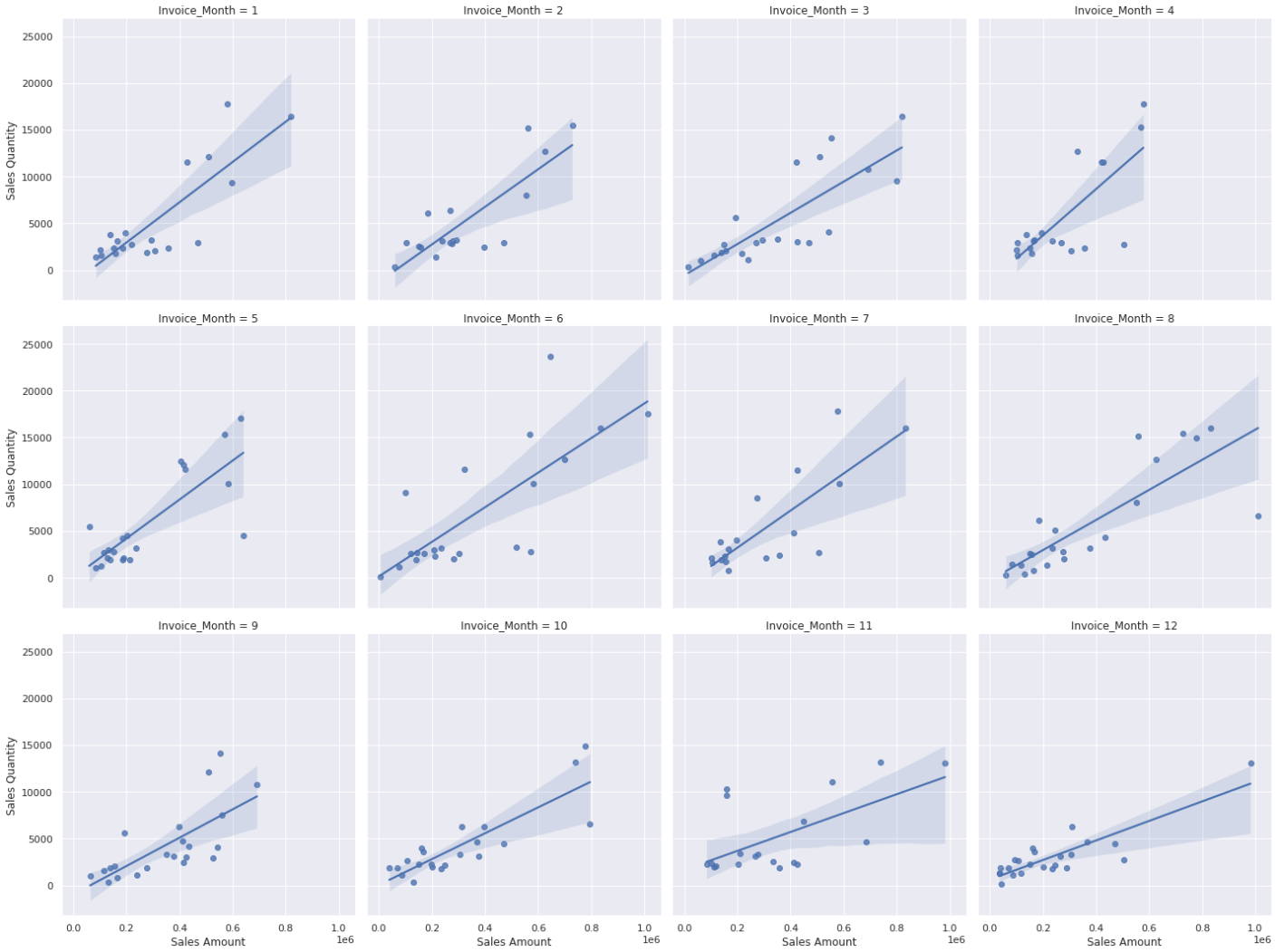

```
sns.jointplot(y='Sales Quantity', x='Sales Amount',
              data = monthly_sales[monthly_sales['Invoice_Year']==2019], kind='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc35a8a5640>
```



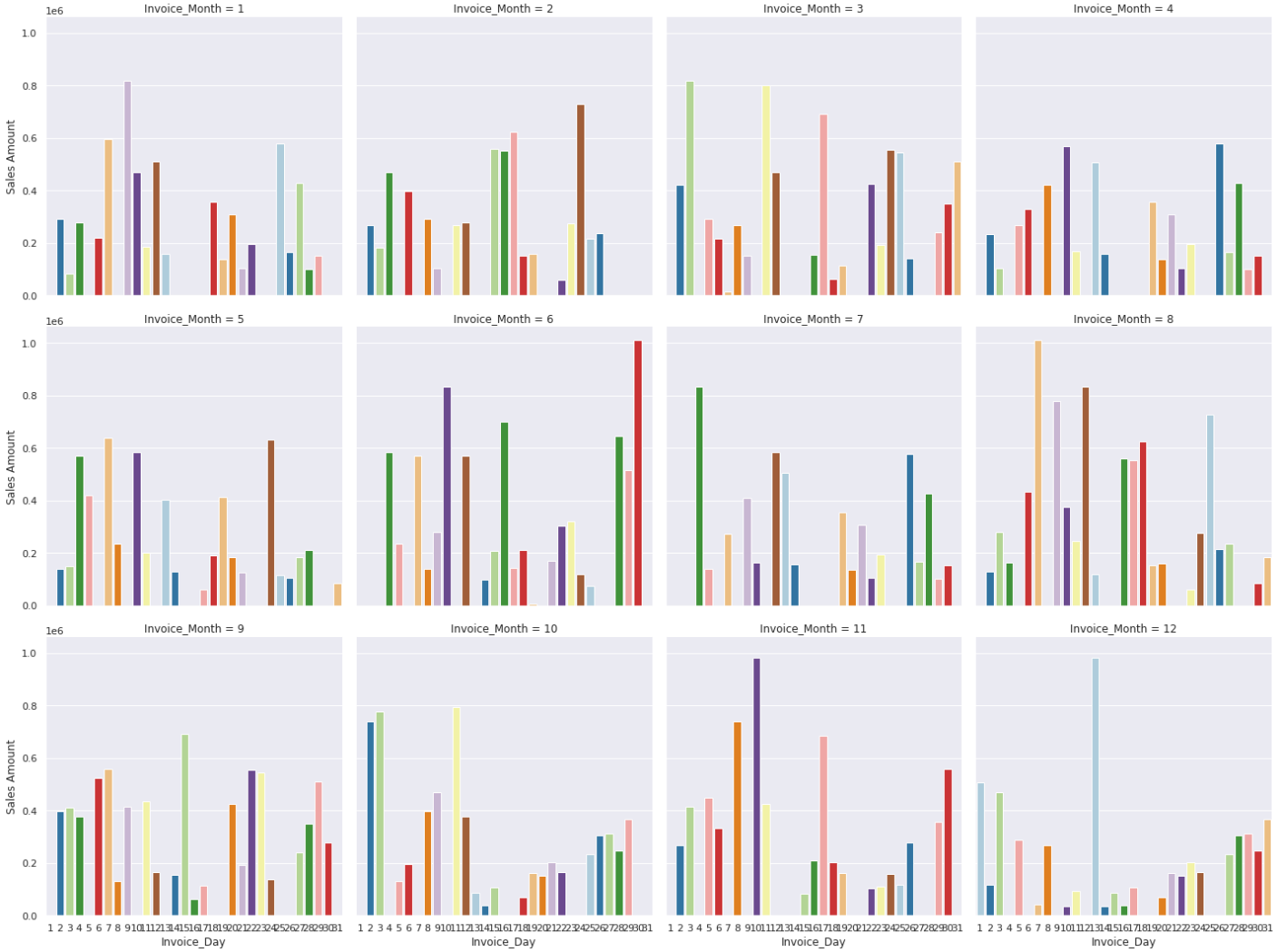

```
sns.lmplot(x='Sales Amount', y = 'Sales Quantity', data = monthly_sales[monthly_sales['
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc35a7646a0>
```



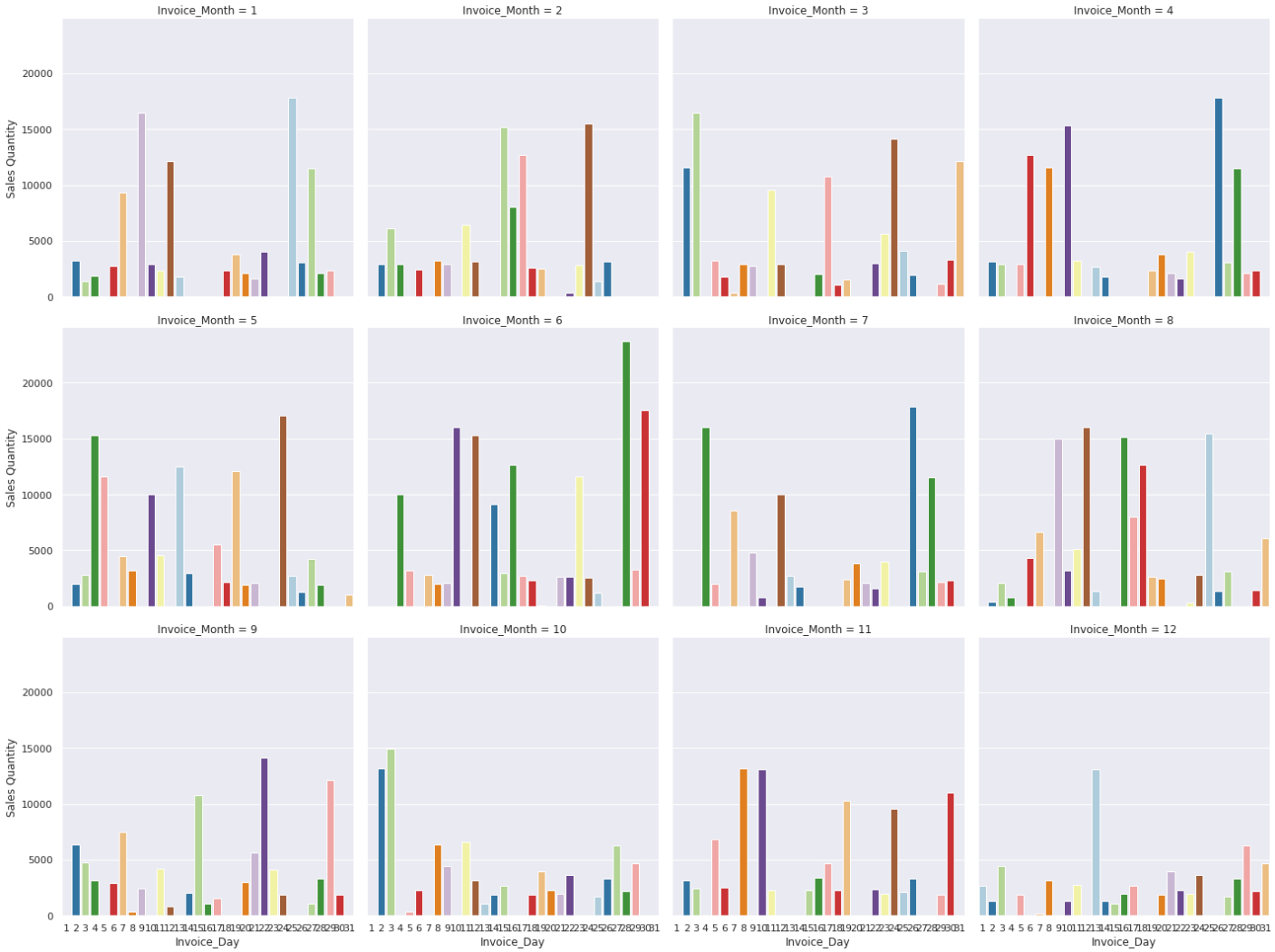

```
sns.catplot(x='Invoice_Day', y= 'Sales Amount', data = monthly_sales[monthly_sales['Inv
            palette = 'Paired', kind='bar', col='Invoice_Month', col_wrap=4)
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc35a2b9a30>
```



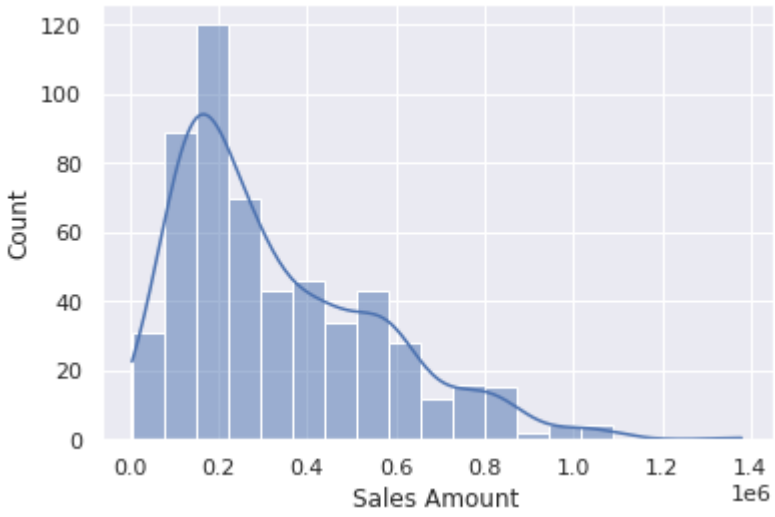

```
sns.catplot(y='Sales Quantity', x='Invoice_Day', data = monthly_sales[monthly_sales['In
            palette = 'Paired', kind='bar', col= 'Invoice_Month', col_wrap = 4)
print('Monthly quantity purchased trend in 2019')
```

```
Monthly quantity purchased trend in 2019
```




```
sns.histplot(monthly_sales['Sales Amount'],kde= True)
```

```
<AxesSubplot:xlabel='Sales Amount', ylabel='Count'>
```

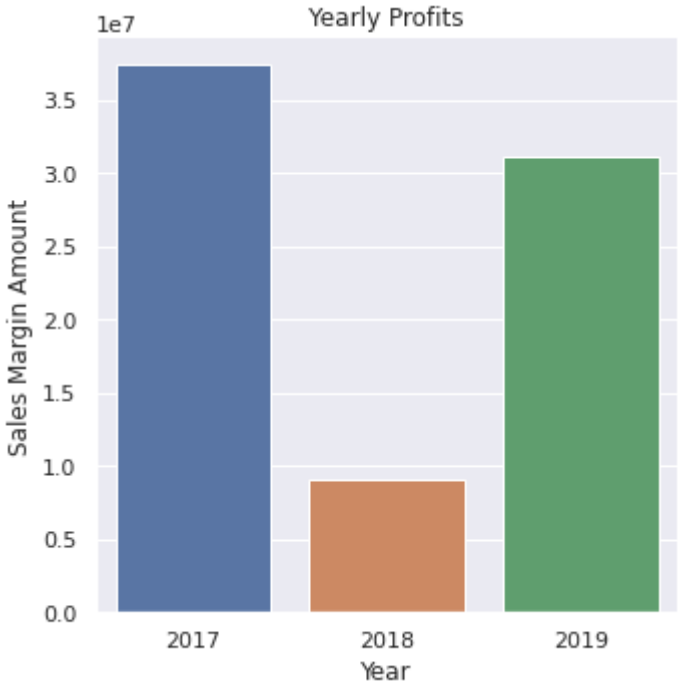


Profit Records:

```
sns.catplot(y='Sales Margin Amount', x='Invoice_Year', data = yearly_sales01,kind='bar'
plt.xlabel('Year')
plt.ylabel('Sales Margin Amount')
```

```
plt.title('Yearly Profits')
yearly_sales01[['Invoice_Year','Sales Margin Amount']]
```

	Invoice_Year	Sales Margin Amount
0	2017	37378575.75
1	2018	9147547.74
2	2019	31182301.05

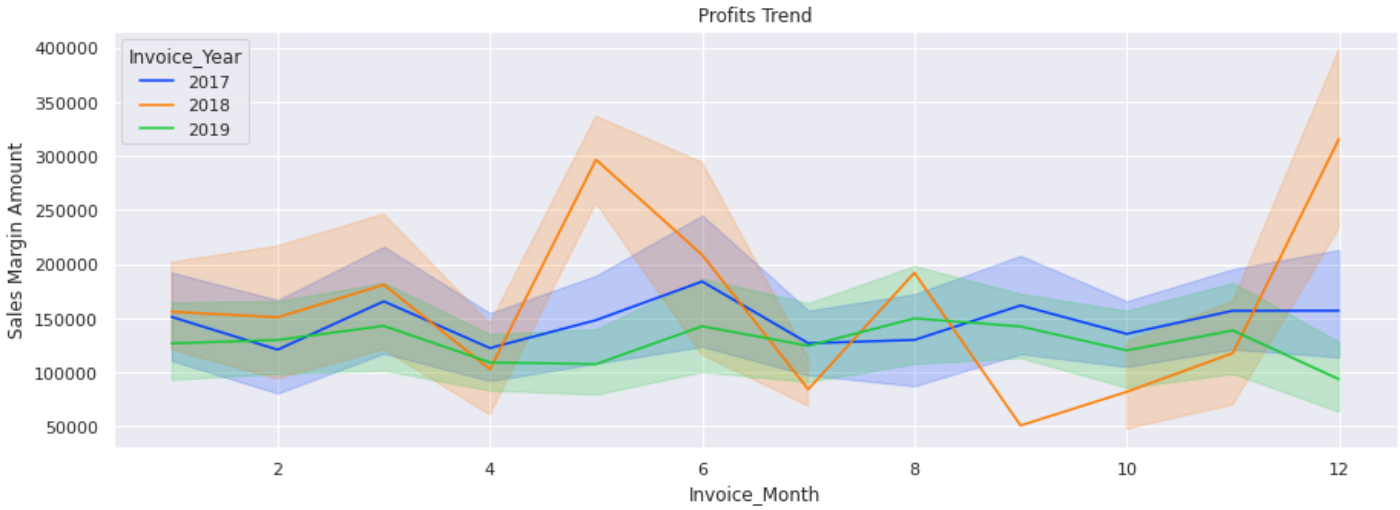



```
jovian.commit()
```

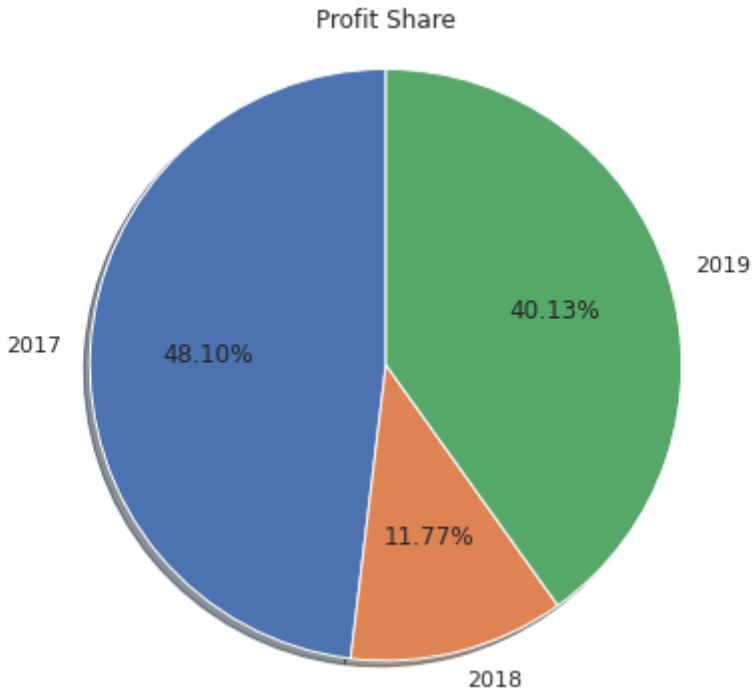
```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis
```

'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'

```
plt.figure(figsize=(15,5))
sns.lineplot(y='Sales Margin Amount', x= 'Invoice_Month',
             data= data02.groupby(['Invoice Date', 'Invoice_Year', 'Invoice_Month']).sum
              hue='Invoice_Year', palette = 'bright')
plt.title('Profits Trend')
plt.show()
```

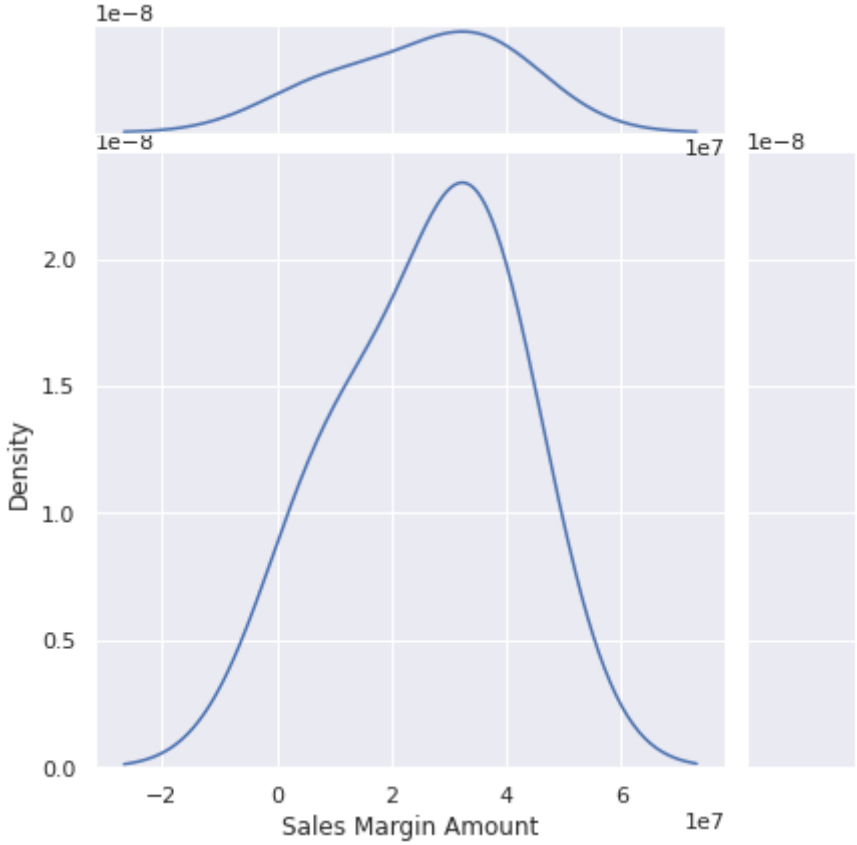



```
plt.figure(figsize=(10,6))
plt.pie('Sales Margin Amount', labels = 'Invoice_Year', data = yearly_sales01[['Invoice
       autopct = '%1.2f%%', shadow = True, startangle = 90)
plt.axis('equal')
plt.title('Profit Share')
plt.show()
```



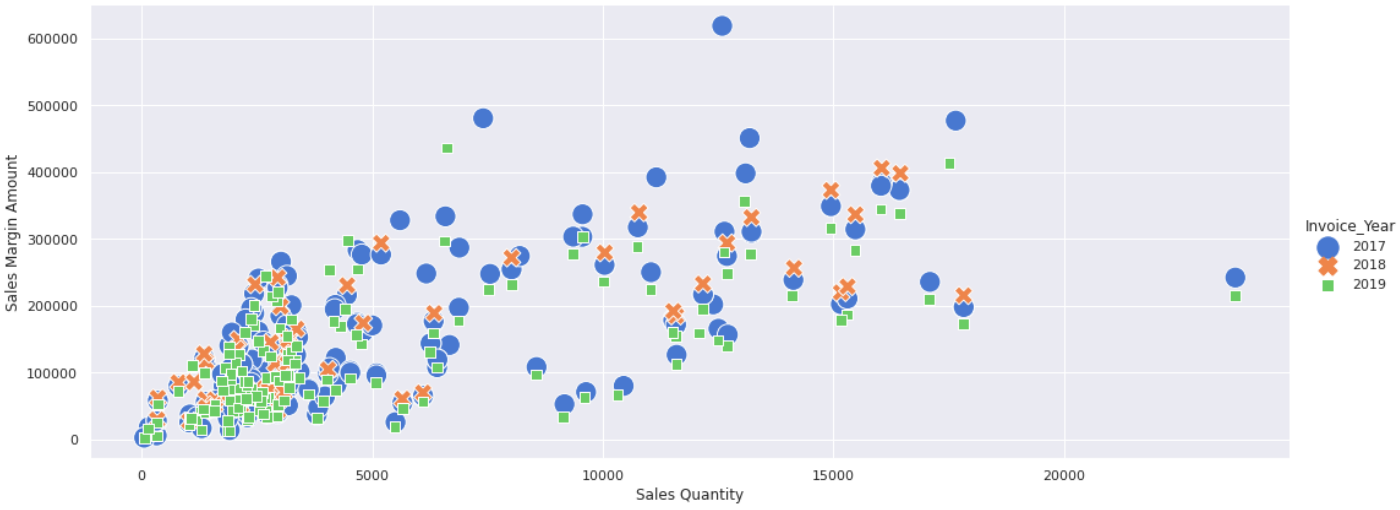

```
sns.jointplot(x='Sales Margin Amount', data = yearly_sales01, kind = 'kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc3591dcf10>
```




```
sns.relplot(x='Sales Quantity', y = 'Sales Margin Amount',
            data= data02.groupby(['Invoice_Year', 'Invoice_Month', 'Invoice_Day']).sum().
            hue = 'Invoice_Year', height = 6, aspect =2.5, size = 'Invoice_Year', palette
            style = 'Invoice_Year', sizes = (300,150))
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc3591dce20>
```




```
jovian.commit()
```

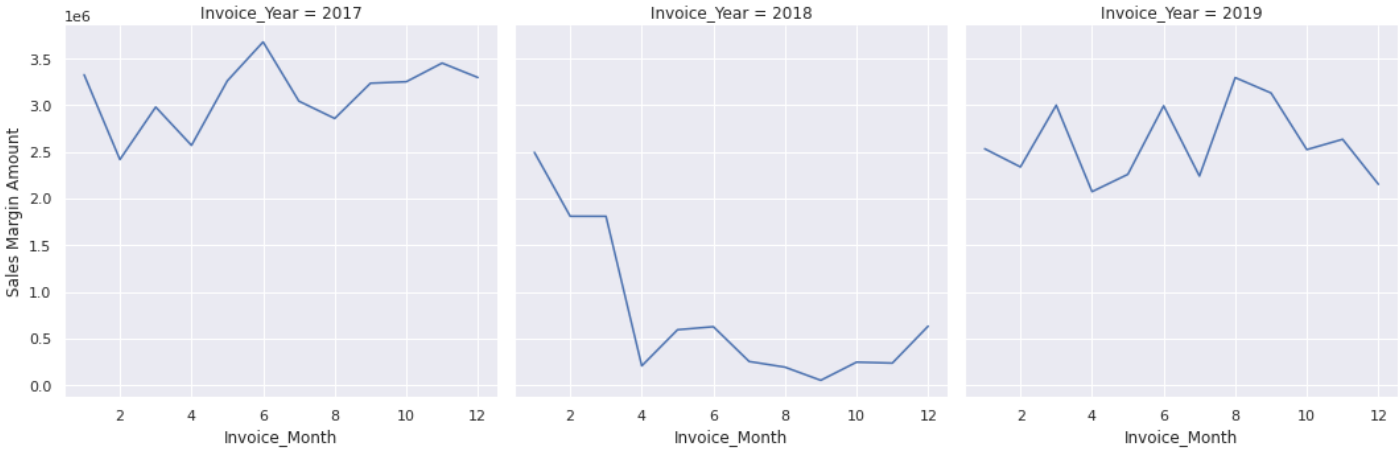
```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-
analysis
```

```
'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'
```

Yearly Month Wise Records:

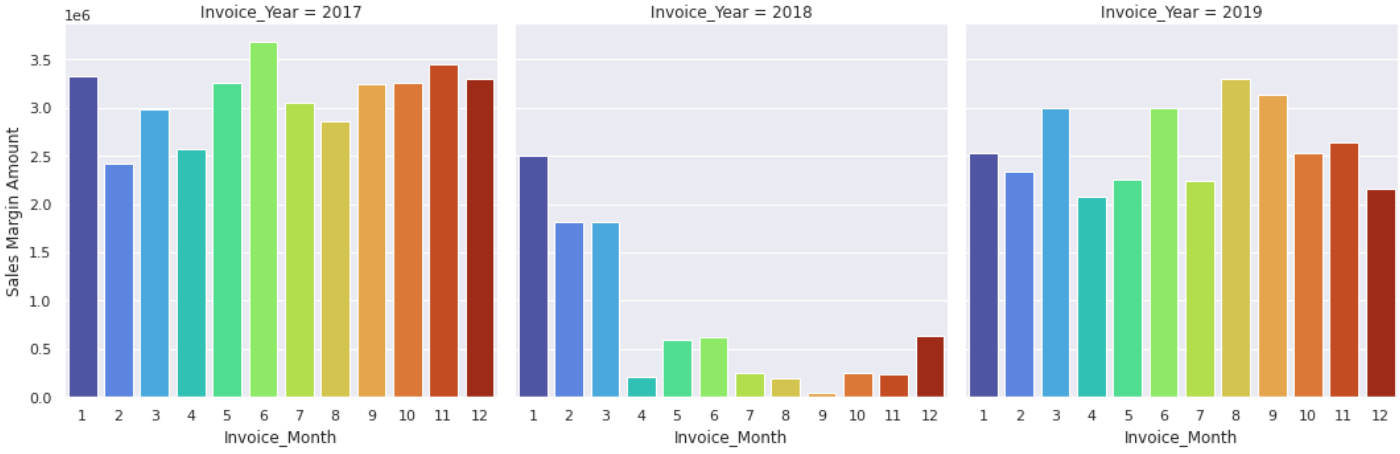
```
sns.relplot(x='Invoice_Month', y = 'Sales Margin Amount', data = yearly_monthwise, heig
            kind='line', aspect = 1, col = 'Invoice_Year')
plt.ylabel('Sales Margin Amount')
print('Yearly-Month wise sales margin trend')
```

```
Yearly-Month wise sales margin trend
```



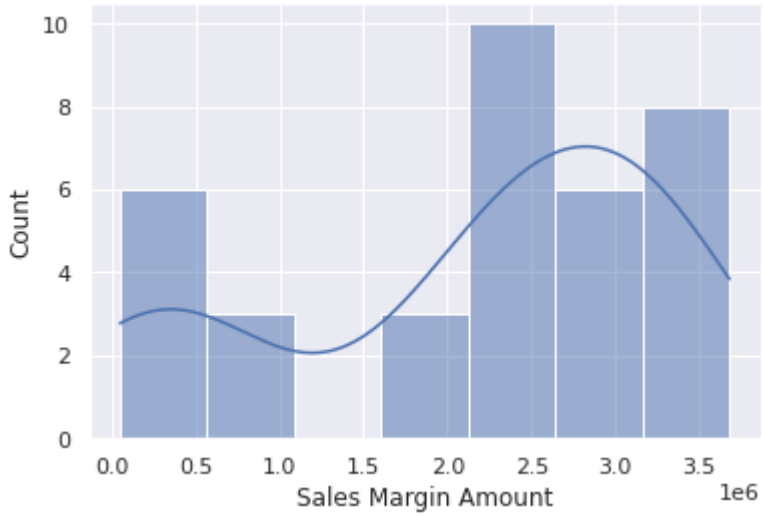

```
sns.catplot(y='Sales Margin Amount', x = 'Invoice_Month', data = yearly_monthwise,palet
            col='Invoice_Year', col_wrap=3,kind = 'bar')
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc358dec5e0>
```




```
sns.histplot(yearly_monthwise['Sales Margin Amount'],kde = True)
```

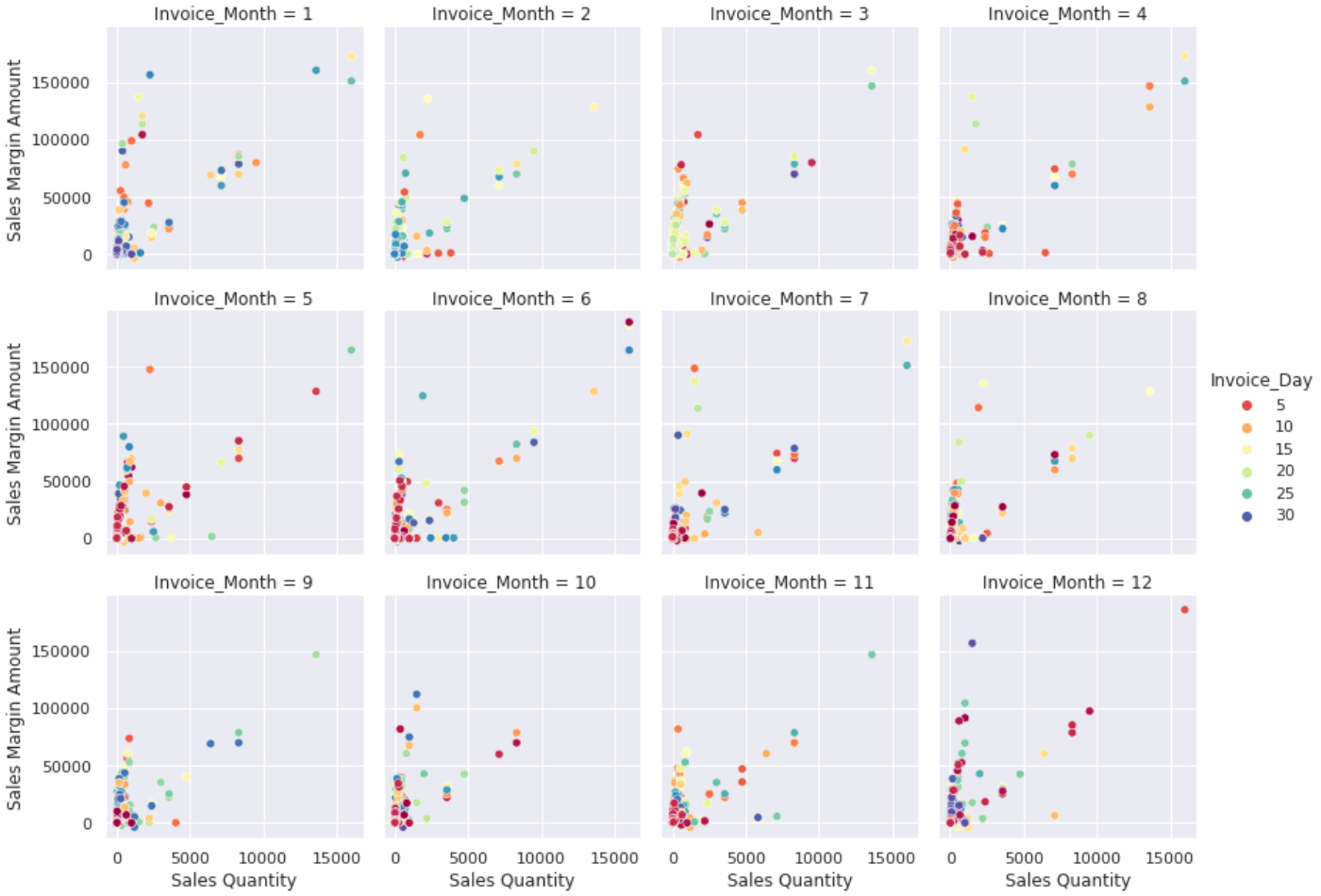
```
<AxesSubplot:xlabel='Sales Margin Amount', ylabel='Count'>
```



Monthly Records:

```
sns.relplot(x='Sales Quantity', y = 'Sales Margin Amount', data= data02,
            height = 3, aspect =1, hue = 'Invoice_Day', col = 'Invoice_Month', col_wrap=
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc358e0cd60>
```



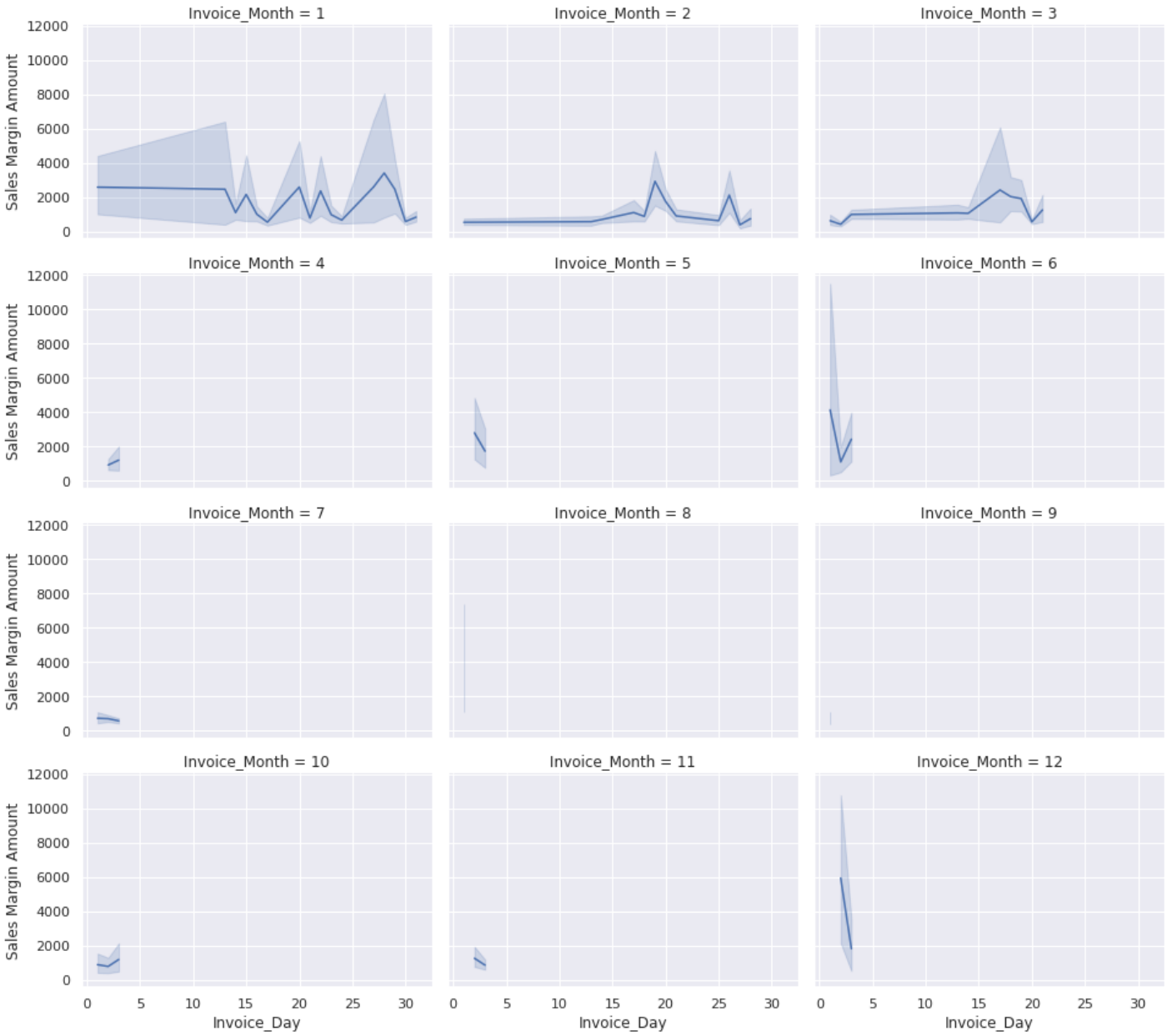

```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Margin Amount', data= data02.query('Invoice_Yea
            kind = 'line', col = 'Invoice_Month', col_wrap = 4, height = 3, aspect = 1.5
plt.ylabel('Sales Amount')
print('Monthly Sales Margin Amount Trend in 2017')
```

Monthly Sales Margin Amount Trend in 2017

<Figure size 576x1440 with 0 Axes>

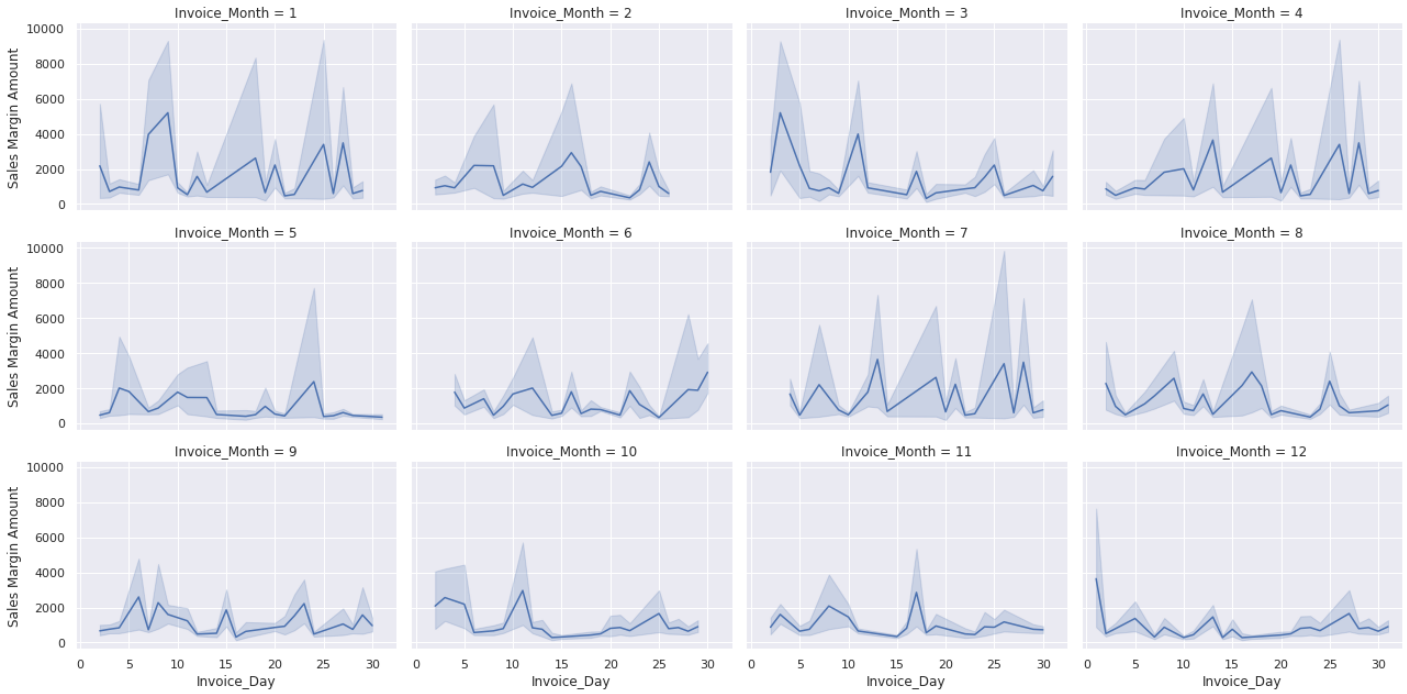

```
sns.relplot(x='Invoice_Day', y = 'Sales Margin Amount', data= data02.query('Invoice_Yea
            kind= 'line', col ='Invoice_Month', col_wrap = 3, height = 3, aspect =1.5)
plt.ylabel('Sales Amount')
print('Monthly Sales Margin Amount Trend in 2018')
```

Monthly Sales Margin Amount Trend in 2018




```
sns.relplot(x='Invoice_Day', y = 'Sales Margin Amount', data = data02.query('Invoice_Ye
            kind='line',col='Invoice_Month', col_wrap = 4, height = 3, aspect = 1.5)
plt.ylabel('Sales Amount')
print('Monthly Sales Margin Amount Trend in 2019')
```

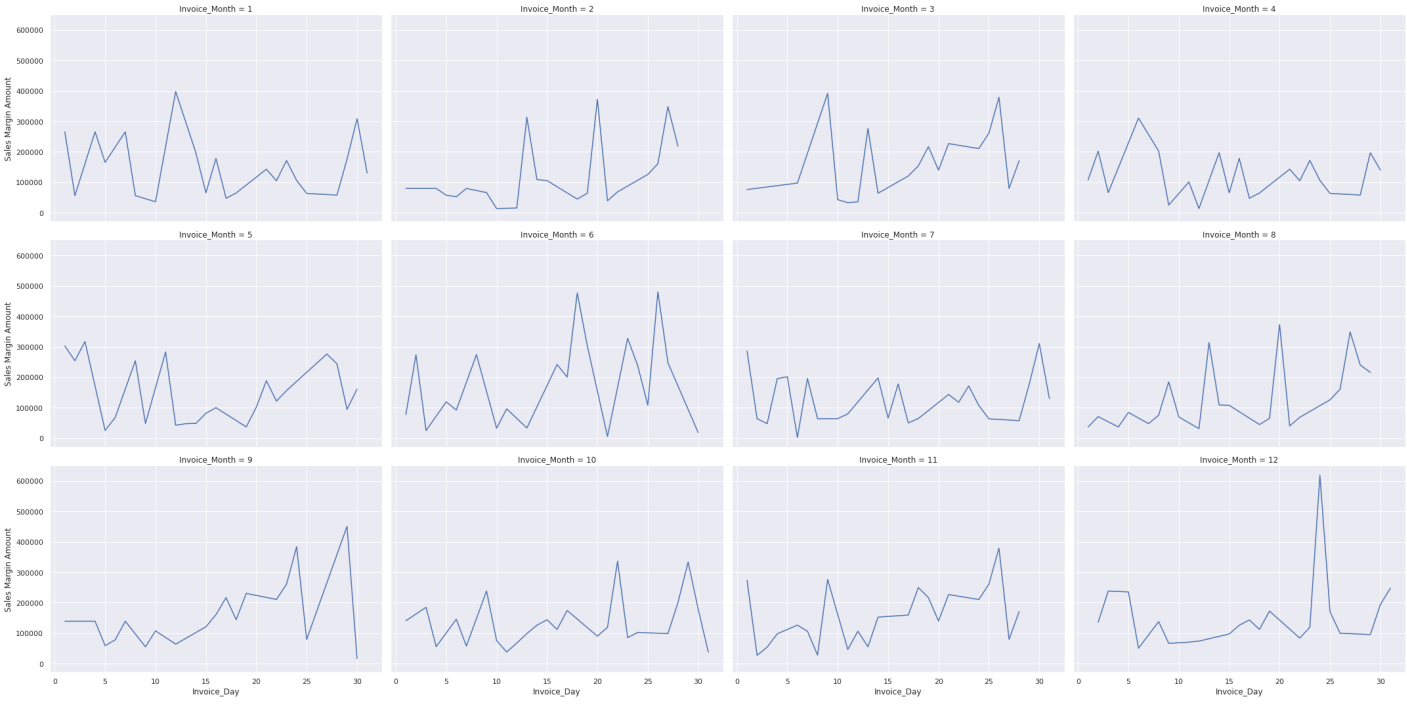
```
Monthly Sales Margin Amount Trend in 2019
```




```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Margin Amount', data = monthly_sales.query('Inv
            kind ='line', col = 'Invoice_Month',col_wrap = 4, aspect = 1.5)
plt.ylabel('Sales Amount')
print('Daily total profit ternd per month in 2017')
```

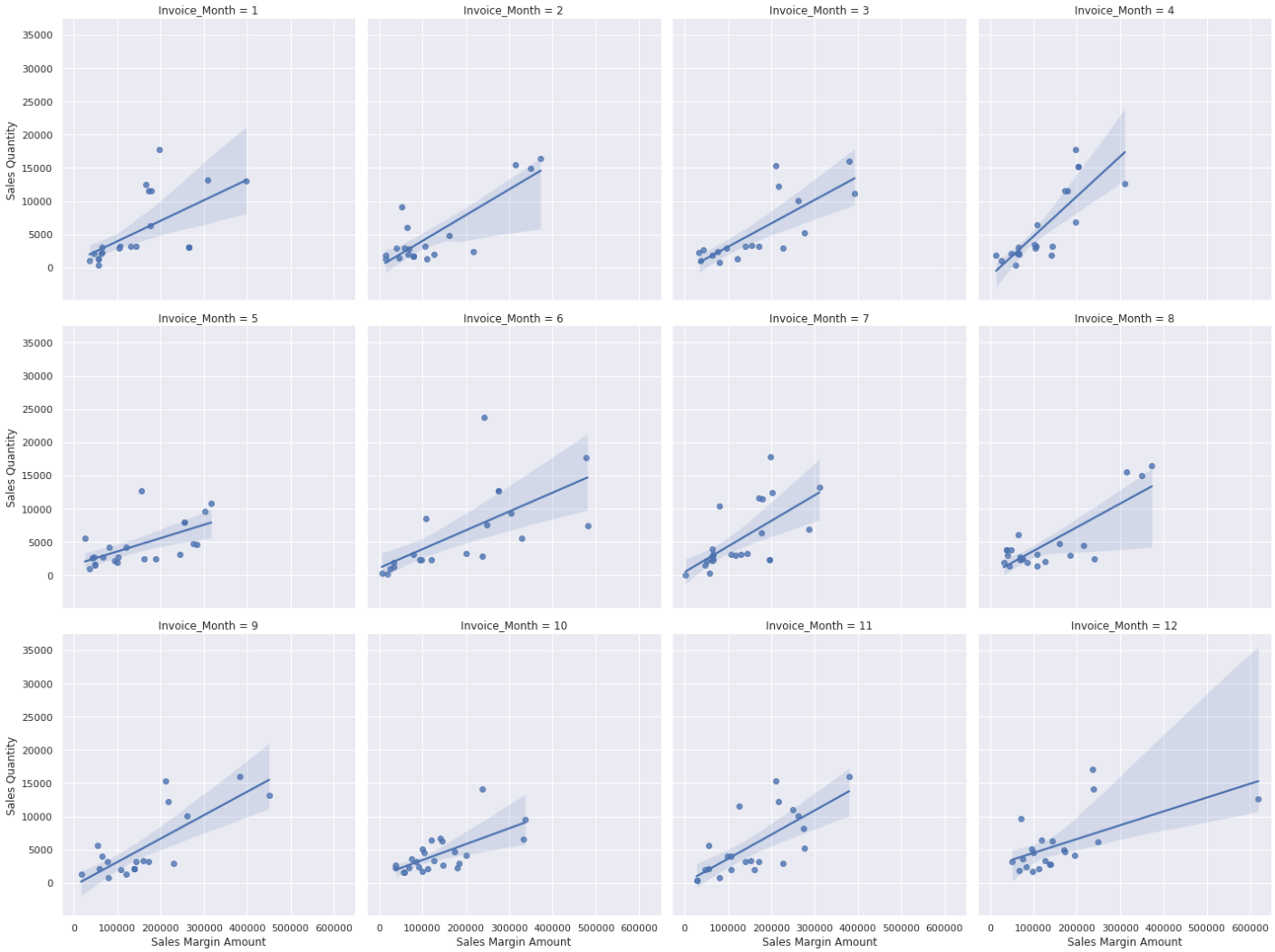
```
Daily total profit ternd per month in 2017
```

```
<Figure size 576x1440 with 0 Axes>
```



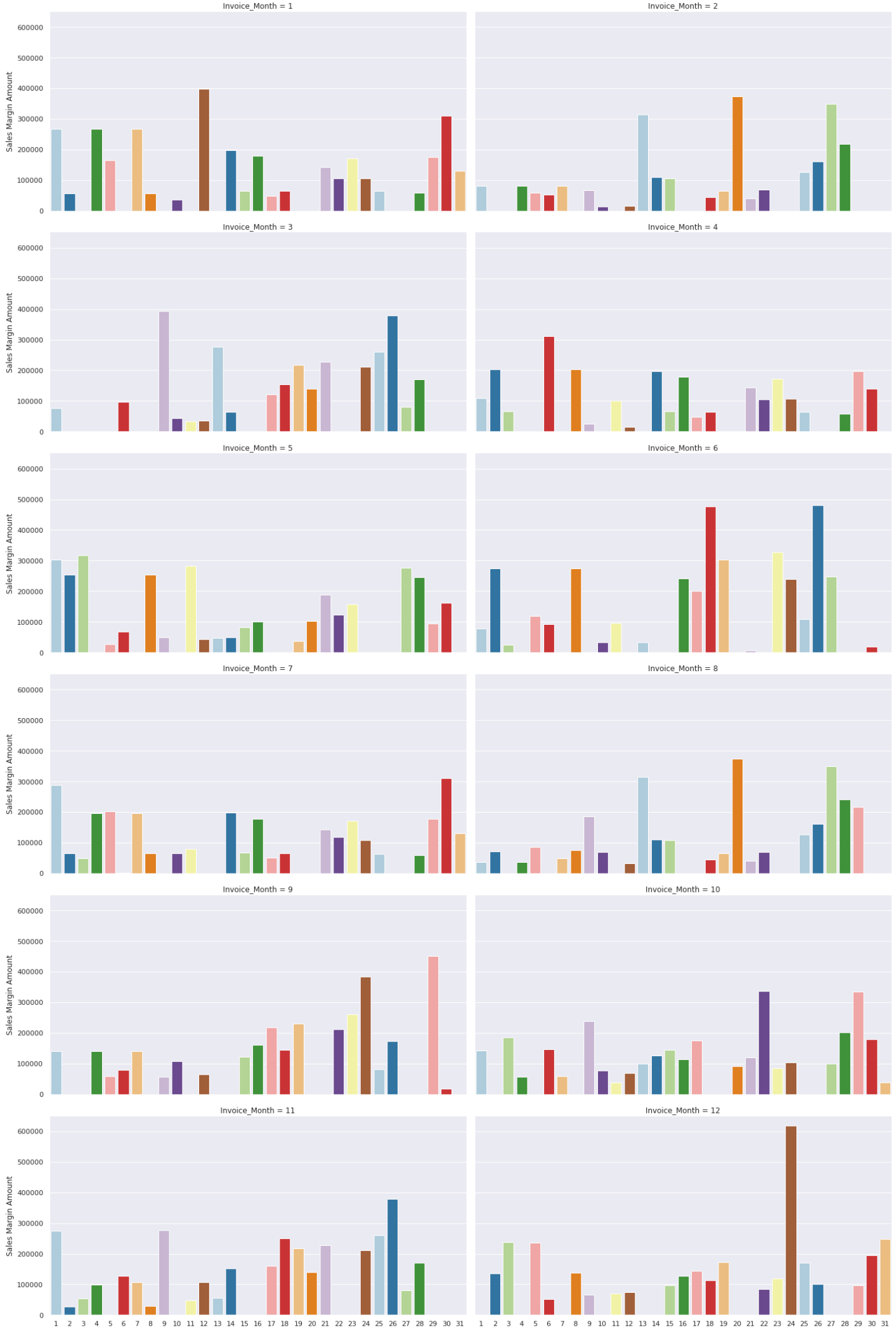

```
sns.lmplot(x='Sales Margin Amount', y = 'Sales Quantity',
           data = monthly_sales[monthly_sales['Invoice_Year']==2017], col = 'Invoice_Mon
```

```
<seaborn.axisgrid.FacetGrid at 0x7fc358573940>
```




```
sns.catplot(y='Sales Margin Amount', x = 'Invoice_Day', data = monthly_sales[monthly_sa
            palette = 'Paired', kind ='bar', col='Invoice_Month', col_wrap = 2)
```

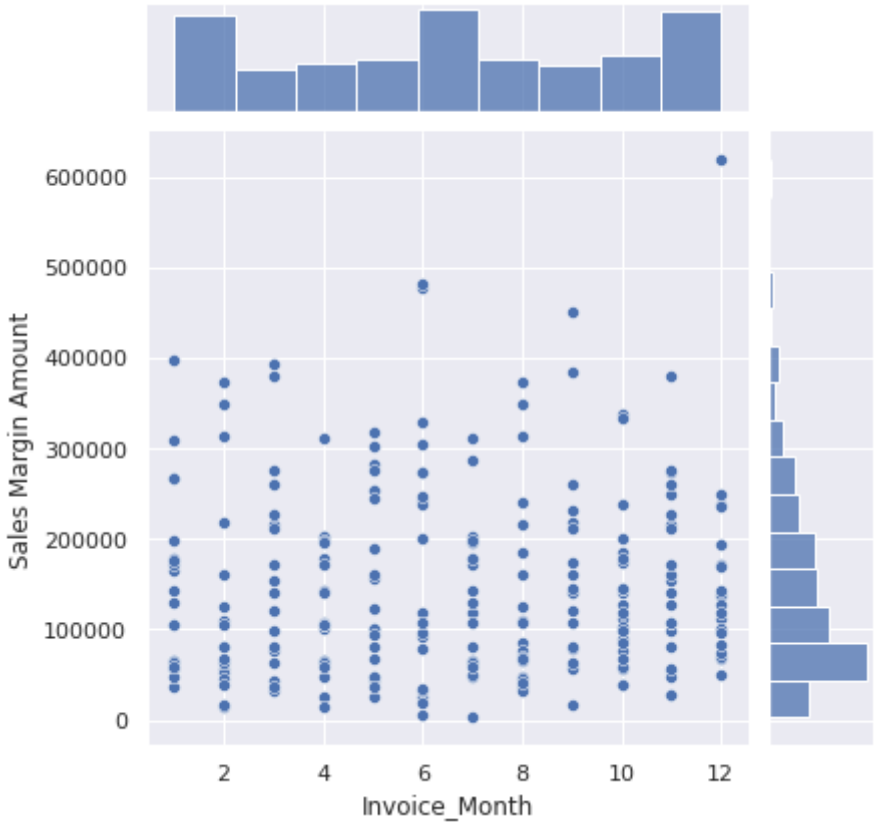
```
<seaborn.axisgrid.FacetGrid at 0x7fc35855ce50>
```


Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Margin Amount
0
100000
200000
300000
400000
500000
600000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Invoice_Day Invoice_Day

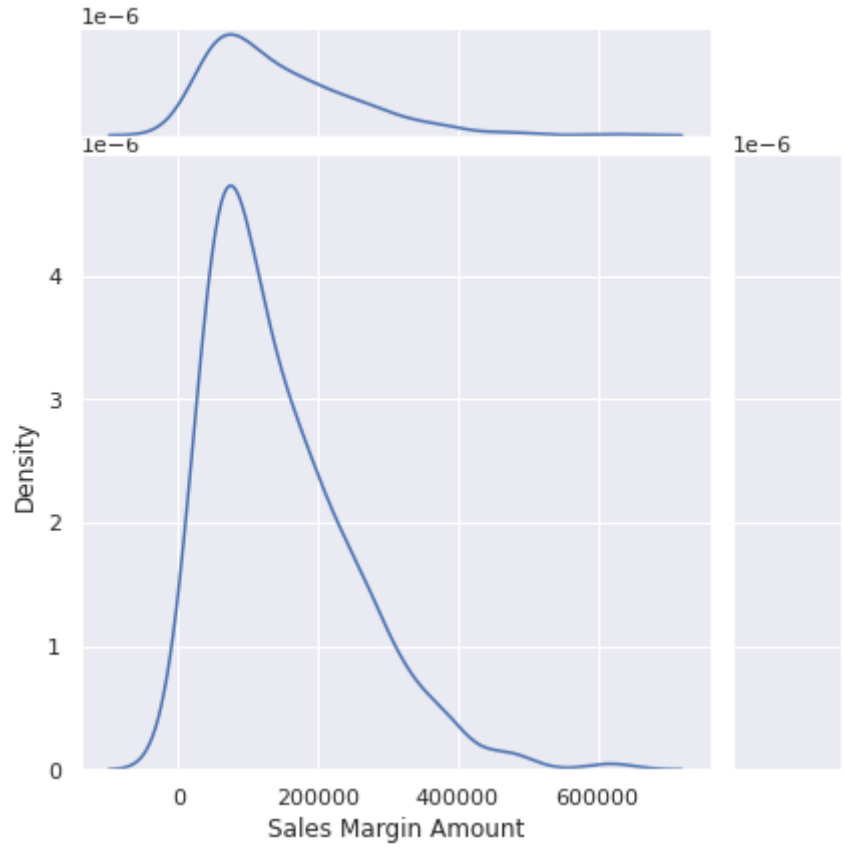
```
sns.jointplot(x='Invoice_Month', y = 'Sales Margin Amount', data = monthly_sales[monthl
```

```
<seaborn.axisgrid.JointGrid at 0x7fc359d82e80>
```



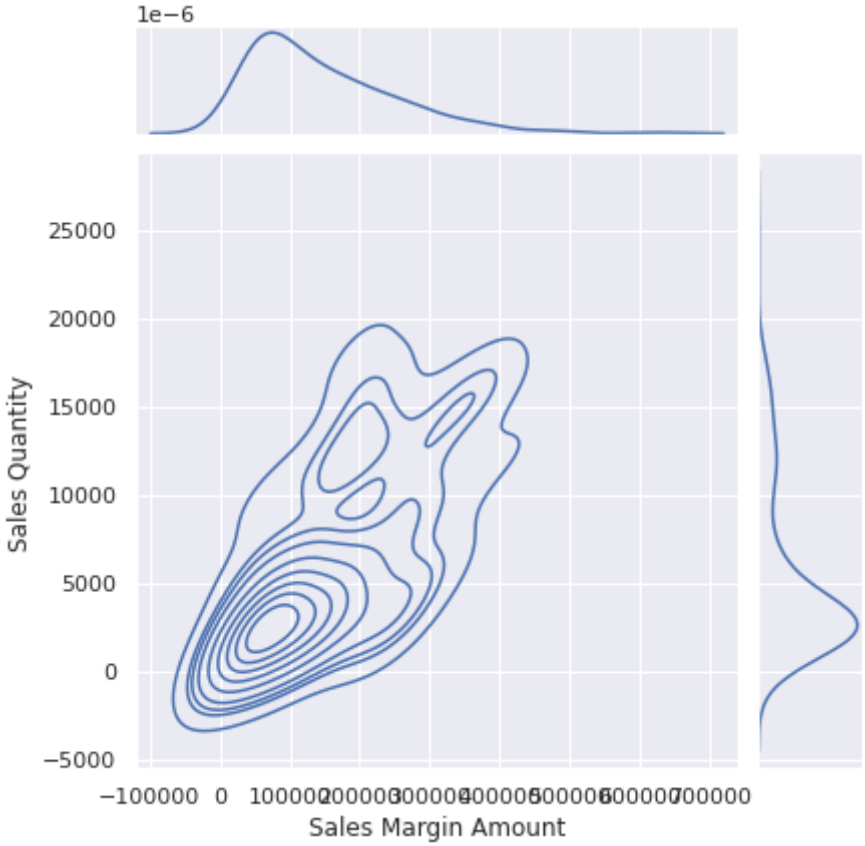

```
sns.jointplot(x='Sales Margin Amount', data = monthly_sales[monthly_sales['Invoice_Year
```

```
<seaborn.axisgrid.JointGrid at 0x7fc3584d24c0>
```



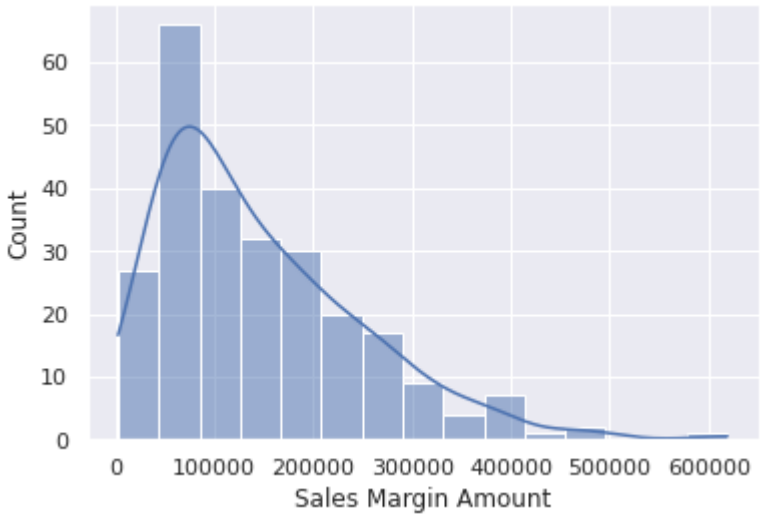
```
sns.jointplot(y='Sales Quantity', x='Sales Margin Amount',
           data = monthly_sales[monthly_sales['Invoice_Year']==2017],kind ='kde')
```

```
<seaborn.axisgrid.JointGrid at 0x7fc358b61490>
```




```
sns.histplot(monthly_sales.query('Invoice_Year ==2017')['Sales Margin Amount'],kde = Tr
```

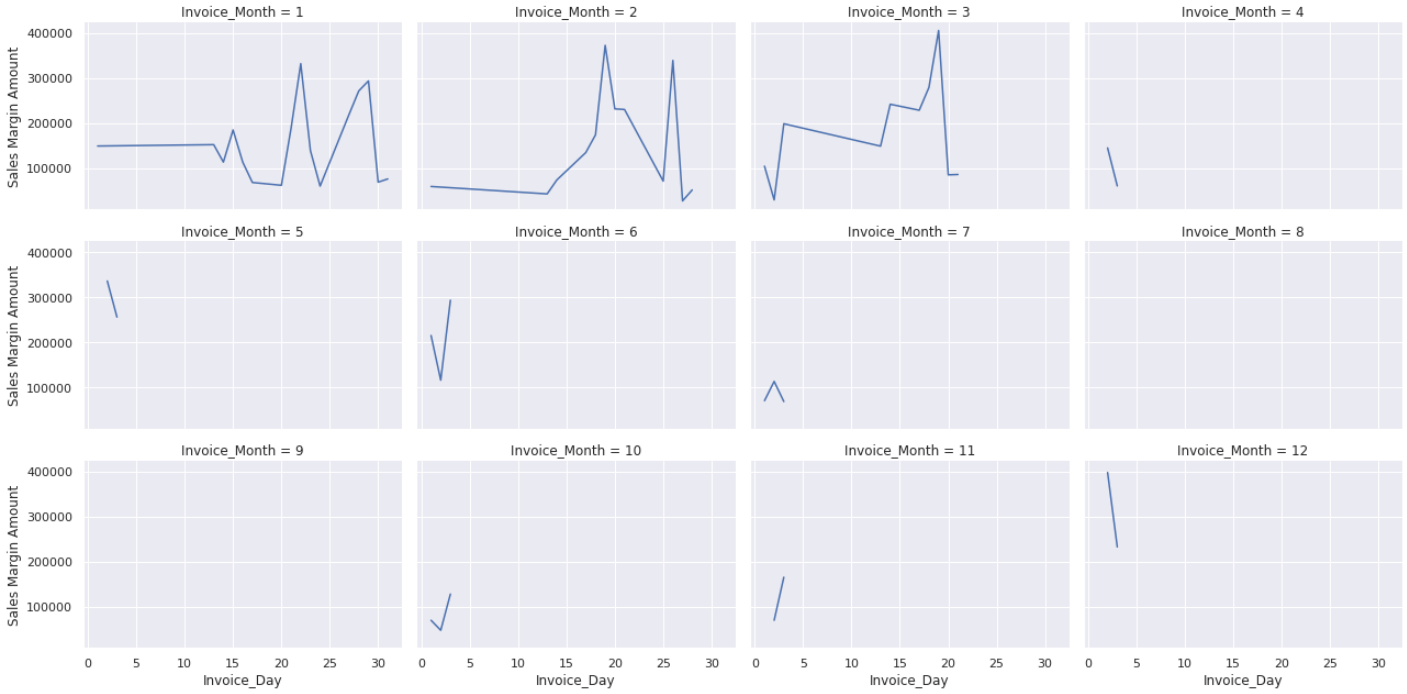
```
<AxesSubplot:xlabel='Sales Margin Amount', ylabel='Count'>
```




```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Margin Amount', data = monthly_sales.query('Inv
           kind = 'line', col = 'Invoice_Month', col_wrap = 4, height = 3, aspect = 1.5
plt.ylabel('Sales Amount')
print('Daily Total profit  per month trend in 2018')
```

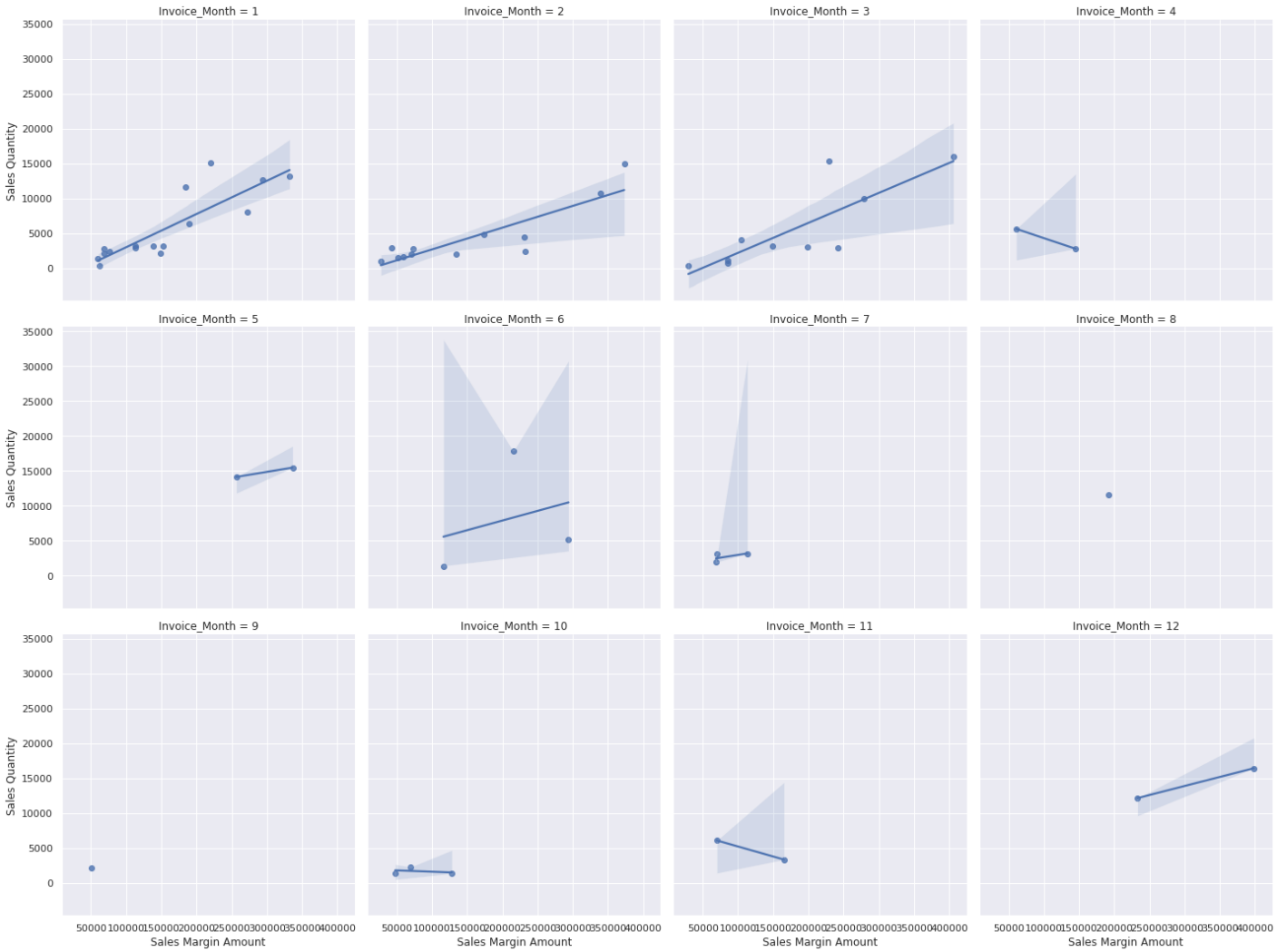
```
Daily Total profit  per month trend in 2018
```

<Figure size 576x1440 with 0 Axes>



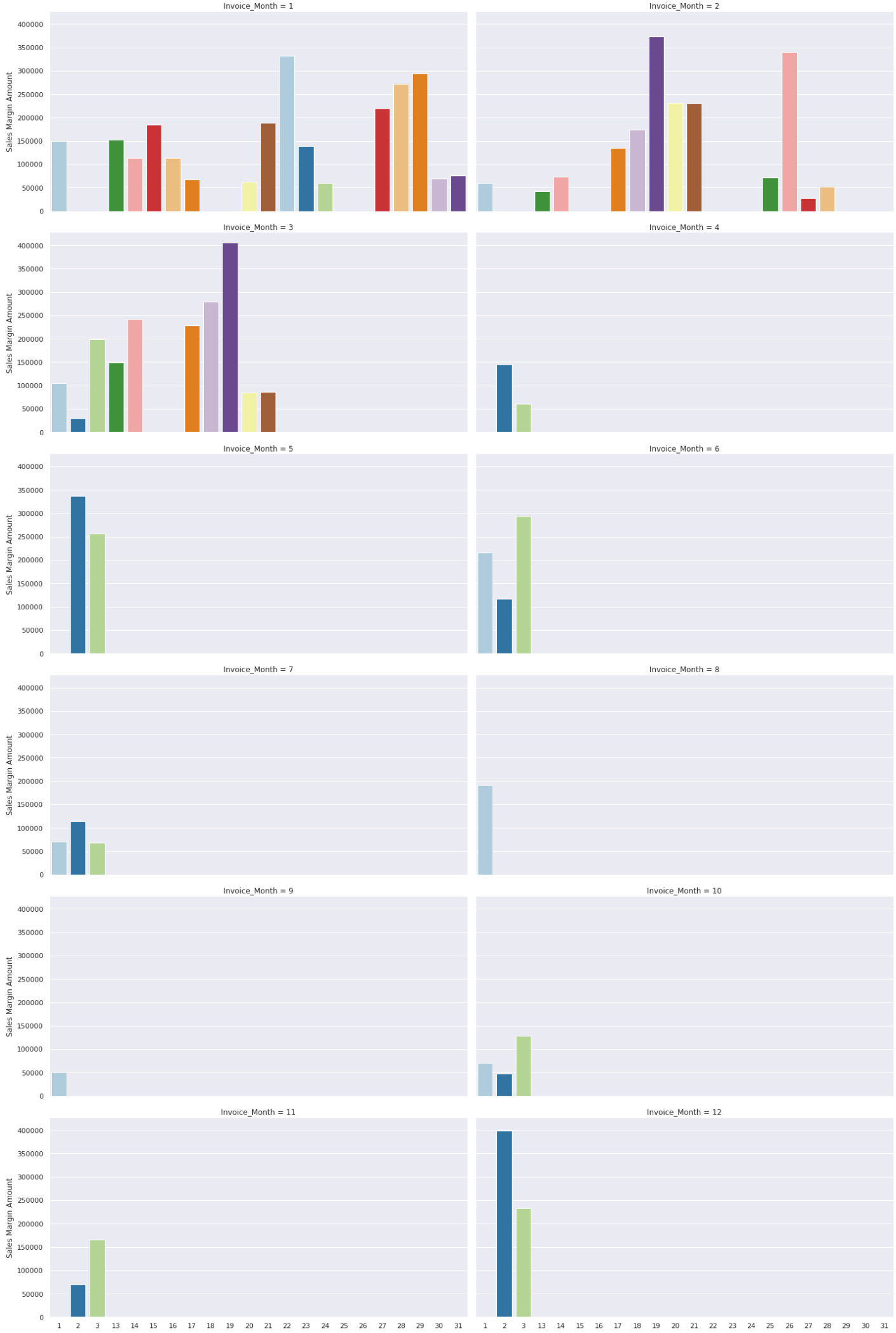

```
sns.lmplot(x='Sales Margin Amount', y = 'Sales Quantity',
           data = monthly_sales[monthly_sales['Invoice_Year']==2018], col ='Invoice_Mont
```

<seaborn.axisgrid.FacetGrid at 0x7fc34bd2e280>



```
sns.catplot(y='Sales Margin Amount', x = 'Invoice_Day', data = monthly_sales[monthly_sa
            palette = 'Paired', kind = 'bar', col ='Invoice_Month',col_wrap = 2)
```

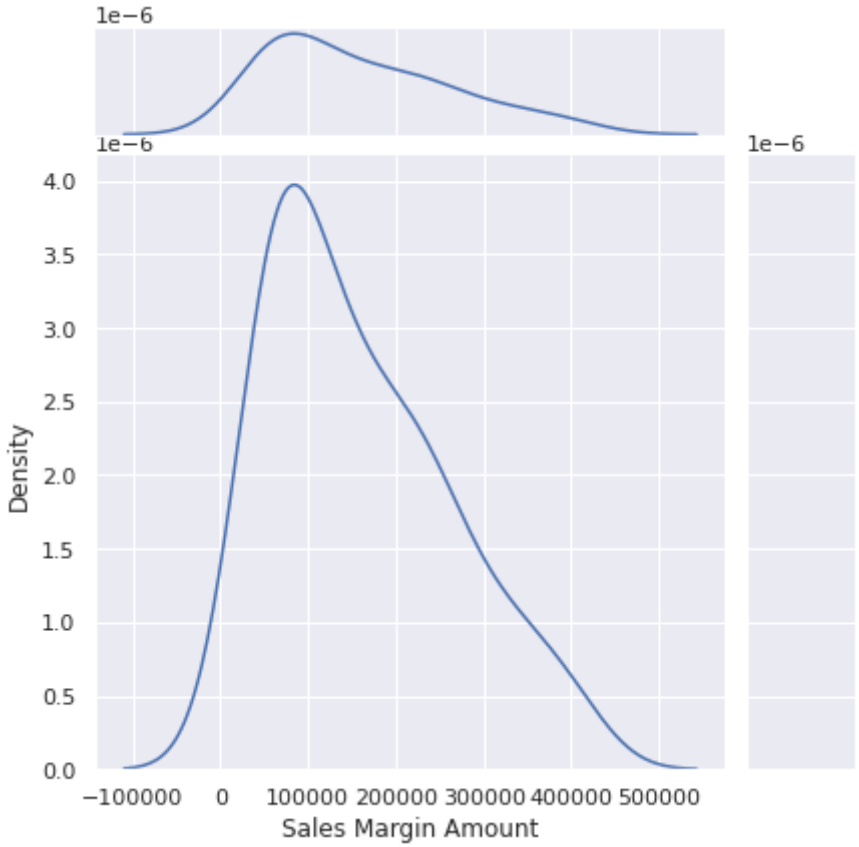
```
<seaborn.axisgrid.FacetGrid at 0x7fc34b67aeb0>
```


Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Margin Amount
0
50000
100000
150000
200000
250000
300000
350000
400000
1 2 3 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Invoice_Day Invoice_Day

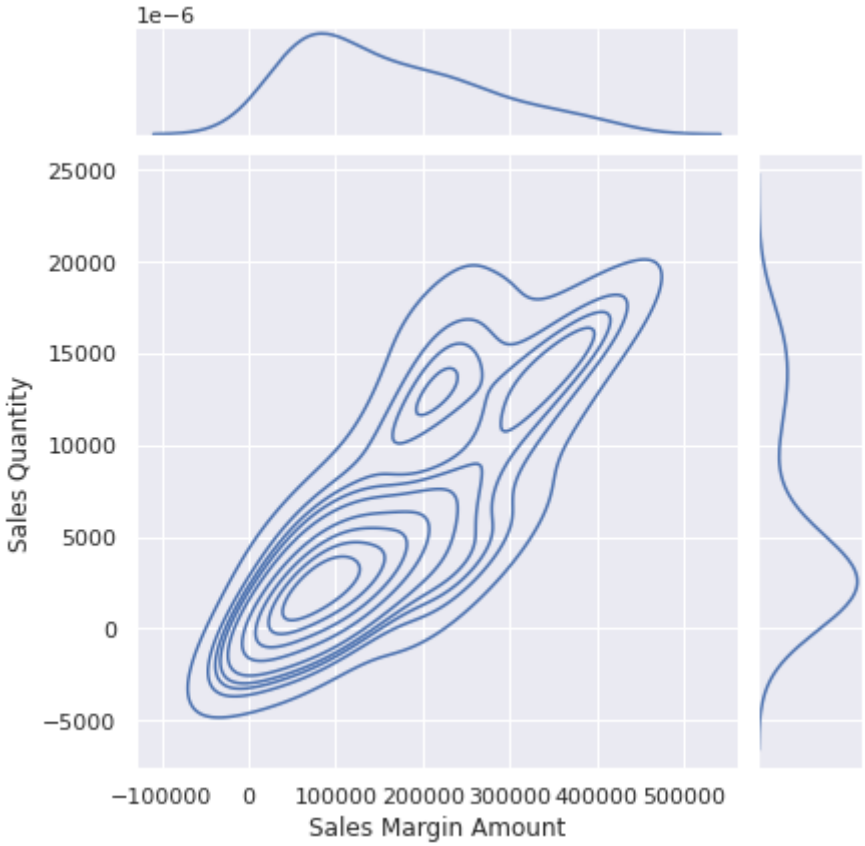
```
sns.jointplot(x='Sales Margin Amount', data =monthly_sales[monthly_sales['Invoice_Year'
```

```
<seaborn.axisgrid.JointGrid at 0x7fc349f14c70>
```



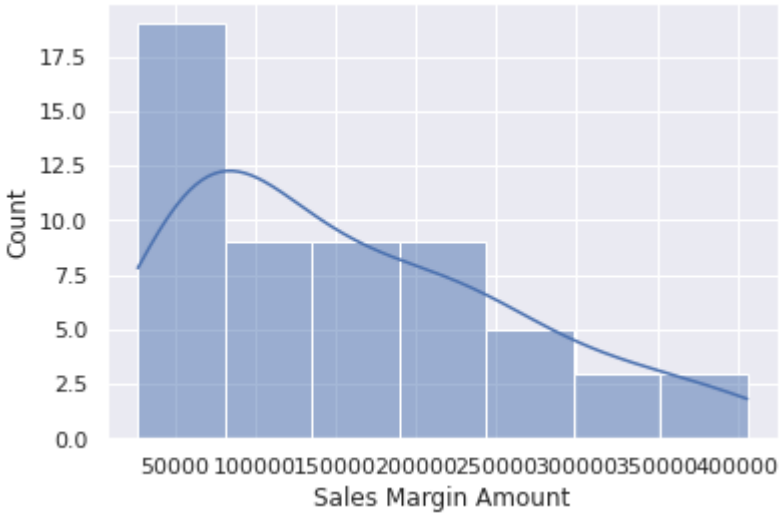

```
sns.jointplot(y='Sales Quantity', x = 'Sales Margin Amount', data = monthly_sales[month
```

```
<seaborn.axisgrid.JointGrid at 0x7fc349f22a30>
```



```
sns.histplot(monthly_sales.query('Invoice_Year==2018')['Sales Margin Amount'],kde = Tru
```

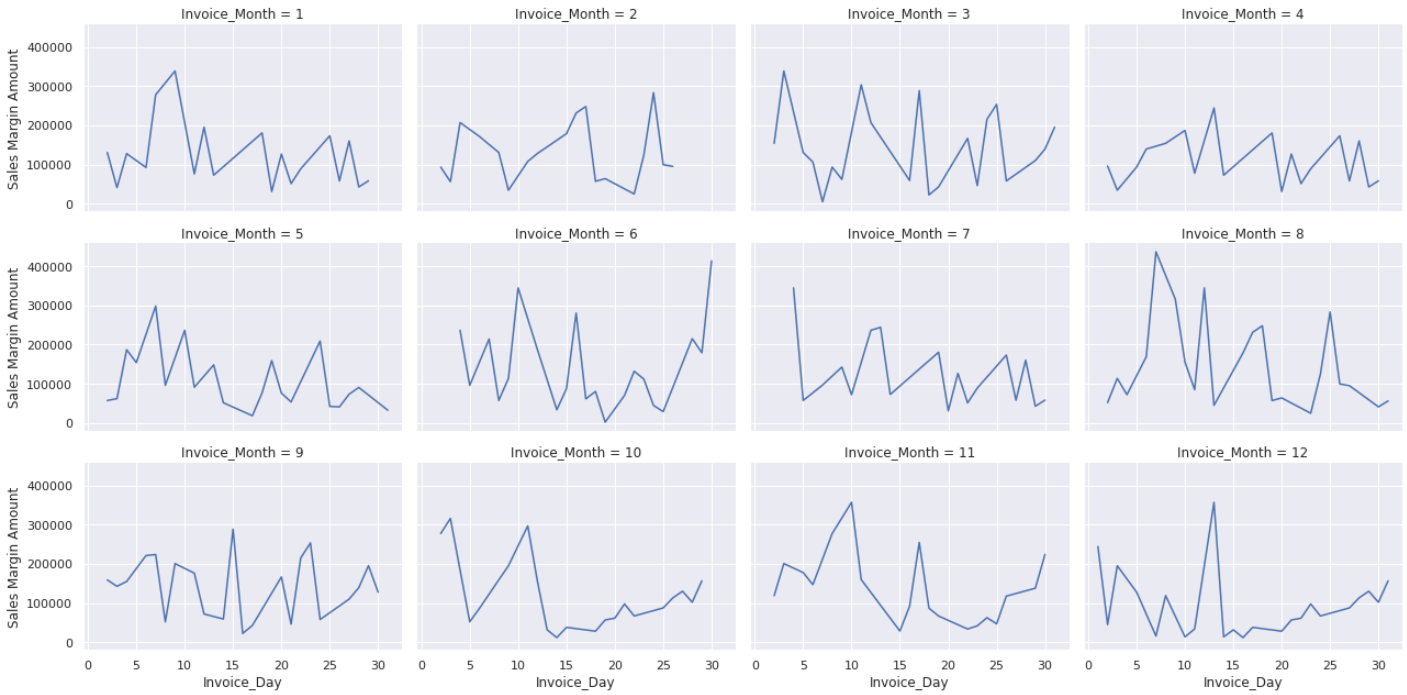
```
<AxesSubplot:xlabel='Sales Margin Amount', ylabel='Count'>
```




```
plt.figure(figsize=(8,20))
sns.relplot(x='Invoice_Day', y = 'Sales Margin Amount', data = monthly_sales.query('Inv
            kind ='line', col = 'Invoice_Month', col_wrap = 4, height =3, aspect =1.5)
plt.ylabel('Sales Amount')
print('Daily total profit trend per month in 2019')
```

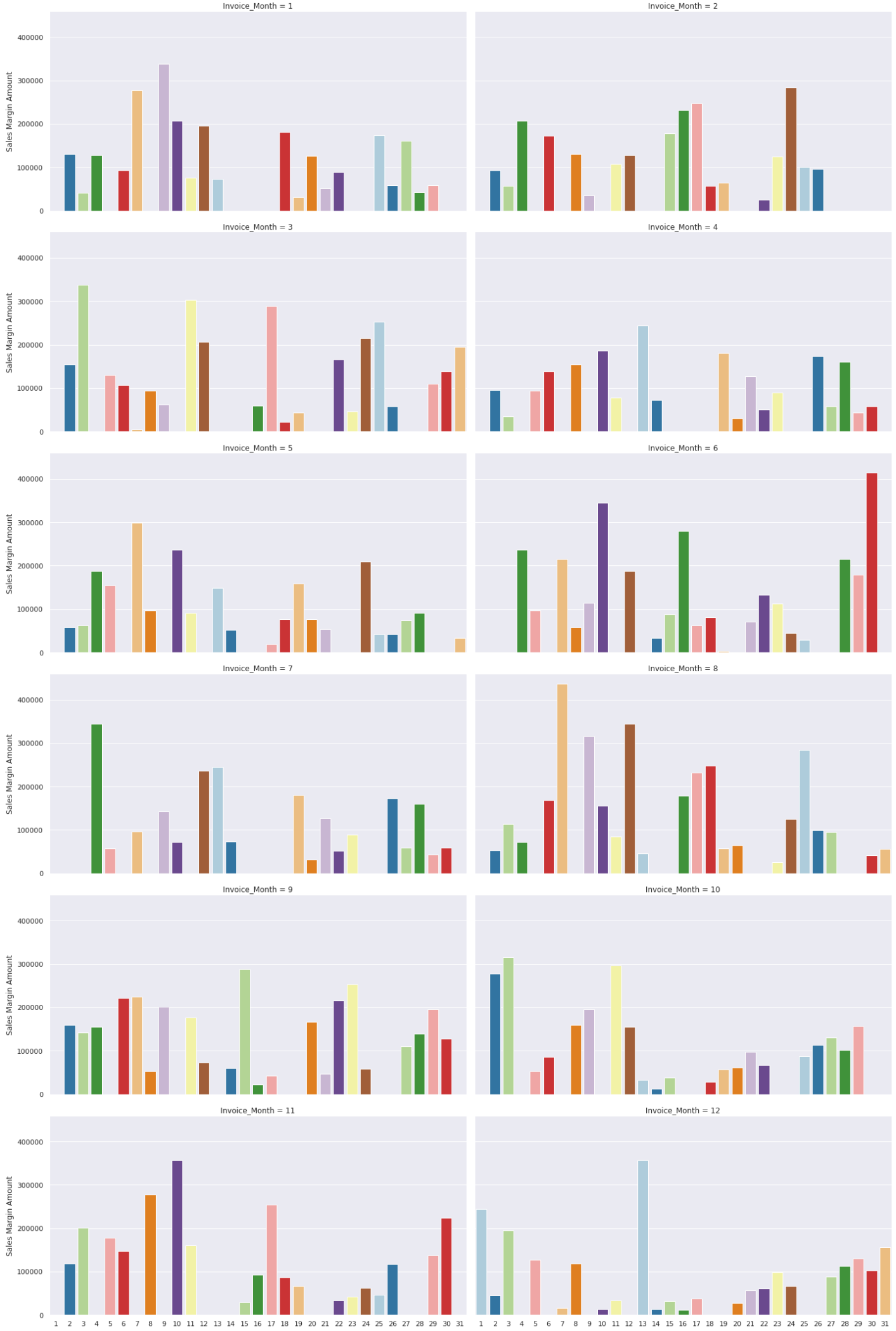
```
Daily total profit trend per month in 2019
```

```
<Figure size 576x1440 with 0 Axes>
```




```
sns.catplot(y='Sales Margin Amount', x = 'Invoice_Day', data = monthly_sales[monthly_sa
            aspect = 2, palette = 'Paired', kind ='bar', col = 'Invoice_Month', col_wrap
```

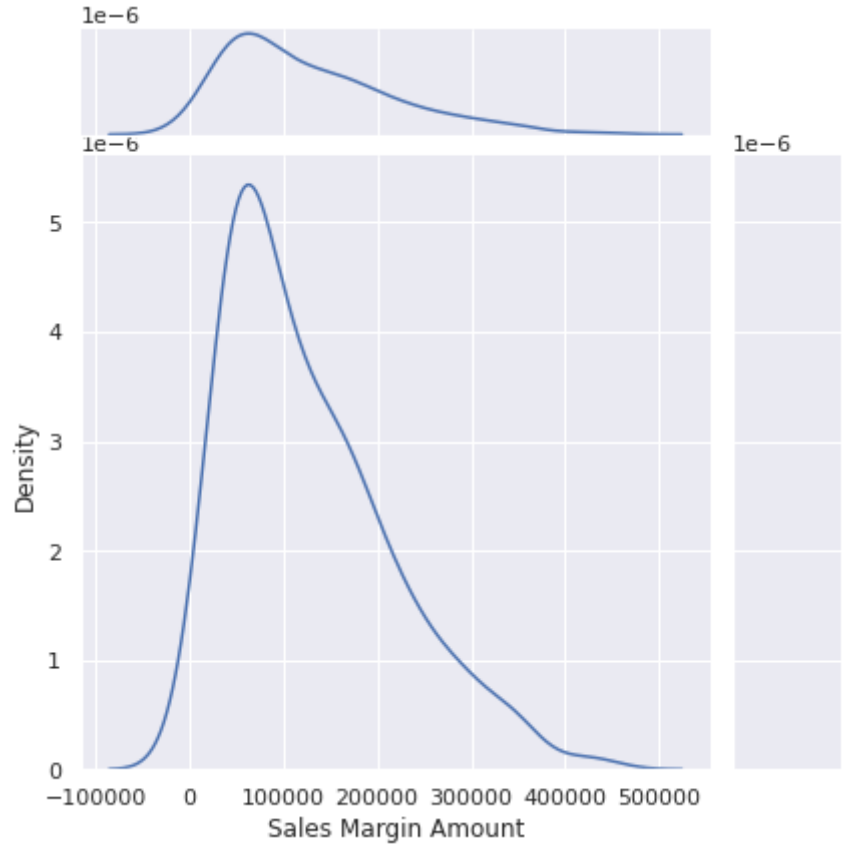
```
<seaborn.axisgrid.FacetGrid at 0x7fc349ae6a00>
```


Invoice_Month = 1
Invoice_Month = 2
Invoice_Month = 3
Invoice_Month = 4
Invoice_Month = 5
Invoice_Month = 6
Invoice_Month = 7
Invoice_Month = 8
Invoice_Month = 9
Invoice_Month = 10
Invoice_Month = 11
Invoice_Month = 12
Sales Margin Amount
0
100000
200000
300000
400000
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Invoice_Day Invoice_Day

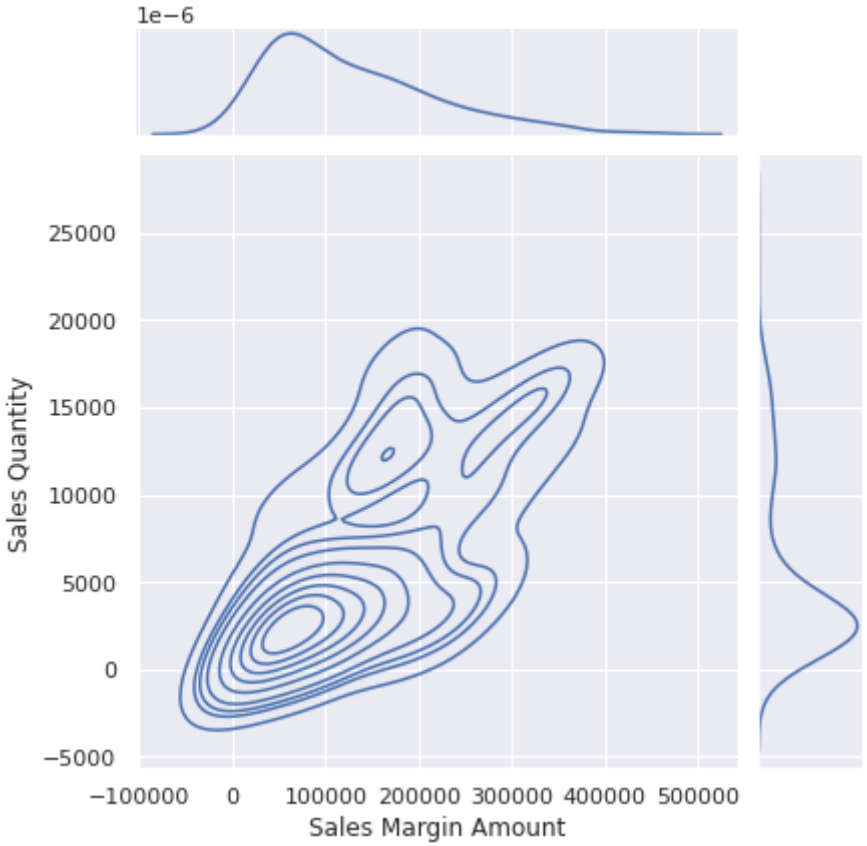
```
sns.jointplot(x='Sales Margin Amount',data = monthly_sales[monthly_sales['Invoice_Year'
```

```
<seaborn.axisgrid.JointGrid at 0x7fc3496d6d90>
```



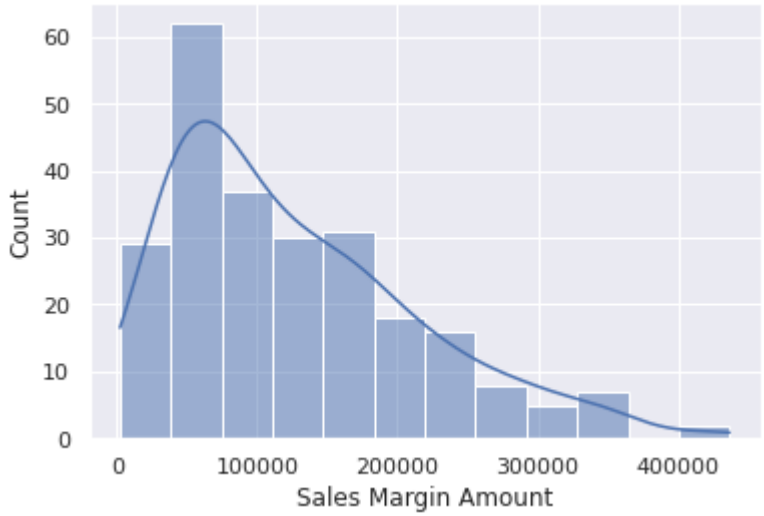

```
sns.jointplot(x='Sales Margin Amount', y = 'Sales Quantity', data = monthly_sales[month
```

```
<seaborn.axisgrid.JointGrid at 0x7fc34bc8b670>
```



```
sns.histplot(monthly_sales.query('Invoice_Year ==2019')['Sales Margin Amount'], kde = T
```

```
<AxesSubplot:xlabel='Sales Margin Amount', ylabel='Count'>
```

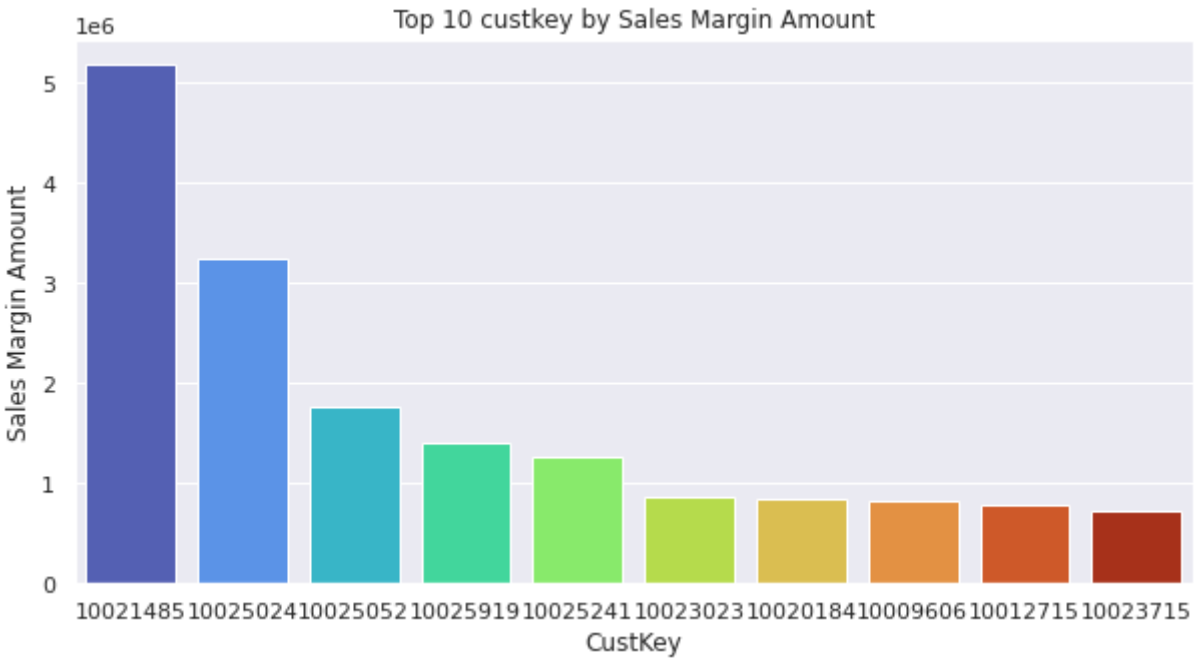


Top 10 Records:

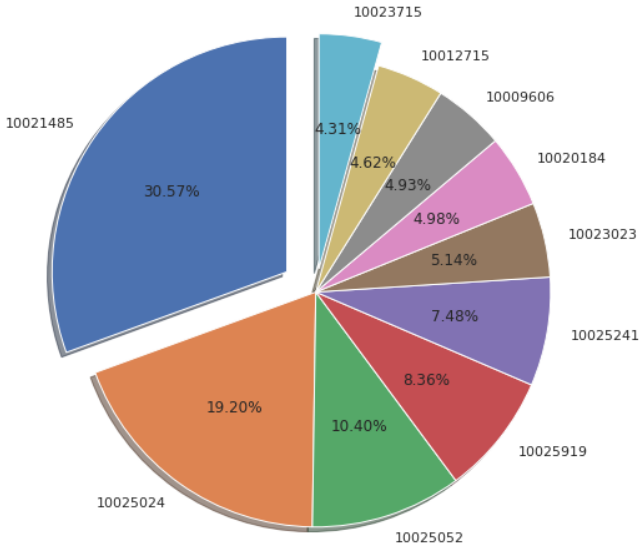
```
top10 = Yearly_Sales[Yearly_Sales['Invoice_Year']==2017].groupby(['Invoice_Year','CustK
top10 = top10.sort_values('Sales Margin Amount', ascending = False).reset_index().head(
```

```
plt.figure(figsize=(10,5))
sns.barplot(x='CustKey',y='Sales Margin Amount', data = top10, palette ='turbo',
            order = top10.CustKey)
plt.title('Top 10 custkey by Sales Margin Amount')
top10[['CustKey', 'Sales Margin Amount']]
```

	CustKey	Sales Margin Amount
0	10021485	5176926.16
1	10025024	3250809.87
2	10025052	1761731.21
3	10025919	1415771.68
4	10025241	1267256.40
5	10023023	870216.36
6	10020184	843138.90
7	10009606	835041.55
8	10012715	782863.50
9	10023715	729615.06




```
plt.figure(figsize=(20,8))
plt.pie('Sales Margin Amount', labels ='CustKey',data = top10, autopct ='%1.2f%%',shadc
plt.axis('equal')
plt.show()
```




```
jovian.commit()
```

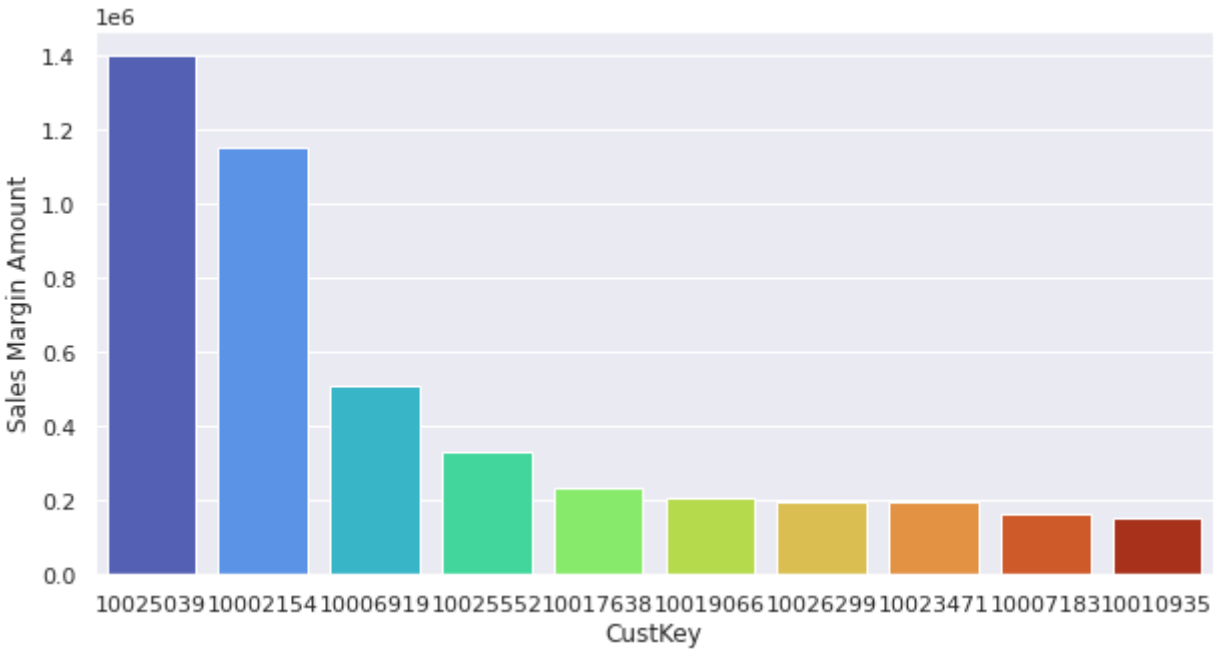
```
[jovian] Updating notebook "sikhapandey0120/amazon-sales-data-analysis" on
https://jovian.ai
[jovian] Committed successfully! https://jovian.ai/sikhapandey0120/amazon-sales-data-
analysis
```

```
'https://jovian.ai/sikhapandey0120/amazon-sales-data-analysis'
```

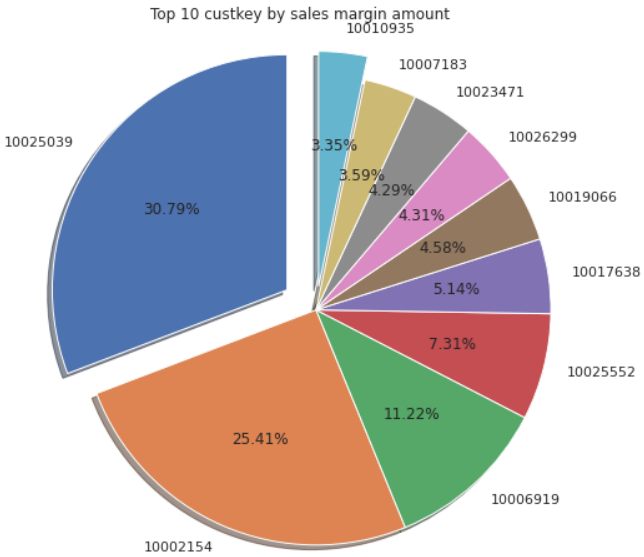
```
top10 = Yearly_Sales[Yearly_Sales['Invoice_Year']==2018].groupby(['Invoice_Year','CustK
top10 = top10.sort_values('Sales Margin Amount',ascending = False).reset_index().head(1
```

```
plt.figure(figsize=(10,5))
sns.barplot(x='CustKey',y = 'Sales Margin Amount',data = top10,palette='turbo',
            order =top10.CustKey)
top10[['CustKey','Sales Margin Amount']]
```

	CustKey	Sales Margin Amount
0	10025039	1398188.29
1	10002154	1154085.14
2	10006919	509760.00
3	10025552	332110.69
4	10017638	233470.15
5	10019066	207997.56
6	10026299	195547.06
7	10023471	195034.39
8	10007183	162898.91
9	10010935	152253.65




```
plt.figure(figsize=(20,8))
plt.pie('Sales Margin Amount', labels ='CustKey',data=top10,autopct='%1.2f%%',
       shadow =True, startangle =90,explode =(0.15,0,0,0,0,0,0,0,0,0.1))
plt.axis('equal')
plt.title('Top 10 custkey by sales margin amount')
plt.show()
```



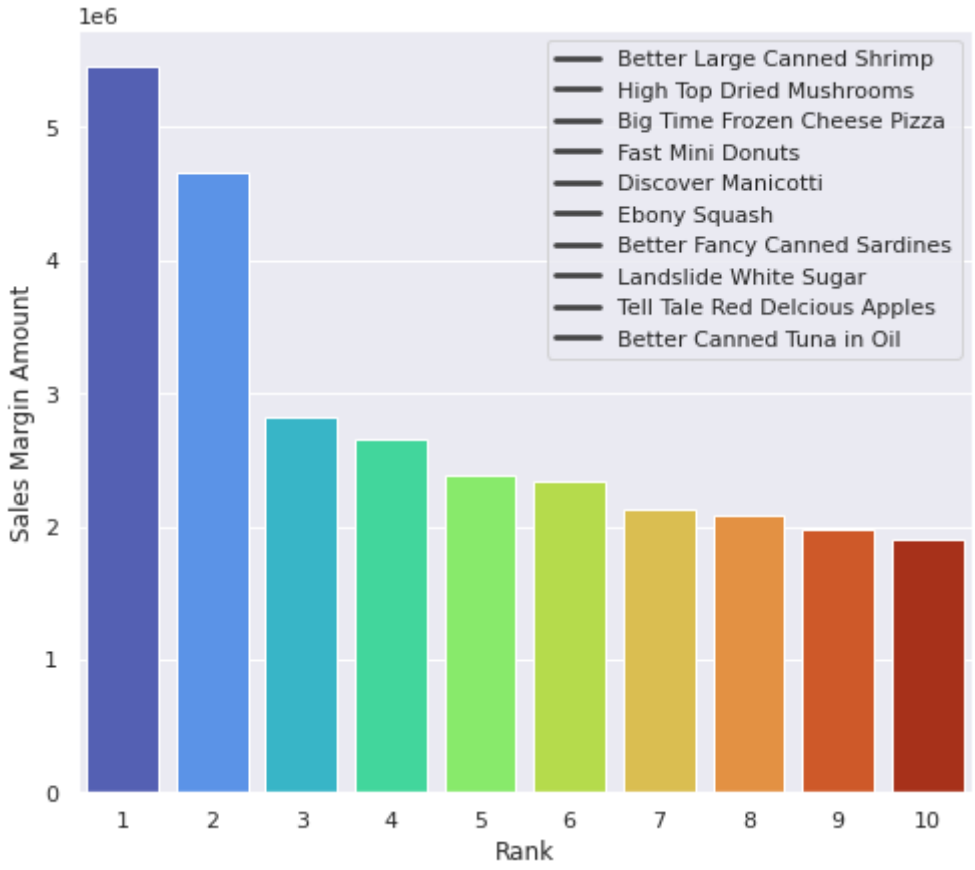
NOTE: Among the top 10 custkey the most sales margin amount ,custkey 10025039 contributed around 30.79% of the sales margin amount.

```
high_profit = data01.groupby('Item').sum().sort_values('Sales Margin Amount', ascending
high_profit.index+=1
high_profit=high_profit.reset_index().rename(columns={'index':'Rank'})
```

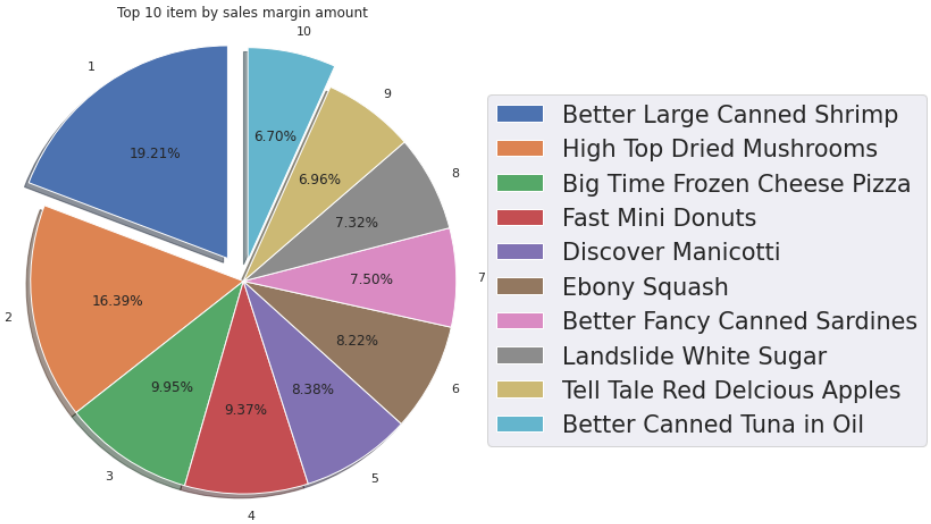
```
plt.figure(figsize=(8,7))
sns.barplot(x='Rank',y='Sales Margin Amount',data = high_profit.head(10),palette ='turb
plt.legend(high_profit['Item'].head(10))
high_profit[['Rank','Item','Sales Margin Amount']]
```

	Rank	Item	Sales Margin Amount
0	1	Better Large Canned Shrimp	5459826.26
1	2	High Top Dried Mushrooms	4659100.16
2	3	Big Time Frozen Cheese Pizza	2826772.99
3	4	Fast Mini Donuts	2663325.66
4	5	Discover Manicotti	2381667.84
...	...	...	...
645	646	Landslide Low Fat Apple Butter	-4026.61
646	647	Carlson Blueberry Yogurt	-4278.90
647	648	Just Right Chicken Soup	-17203.76
648	649	Best Choice Fondue Mix	-22426.53
649	650	Fast Lemon Cookies	-46106.59

650 rows × 3 columns




```
plt.figure(figsize=(22,8))
plt.pie('Sales Margin Amount',labels='Rank',data =high_profit.head(10),autopct='%1.2f%%
       shadow = True, startangle = 90, explode = (0.13,0,0,0,0,0,0,0,0,0.1))
plt.axis('equal')
plt.title('Top 10 item by sales margin amount')
plt.legend(high_profit['Item'].head(10),loc=7,fontsize='xx-large')
plt.show()
```

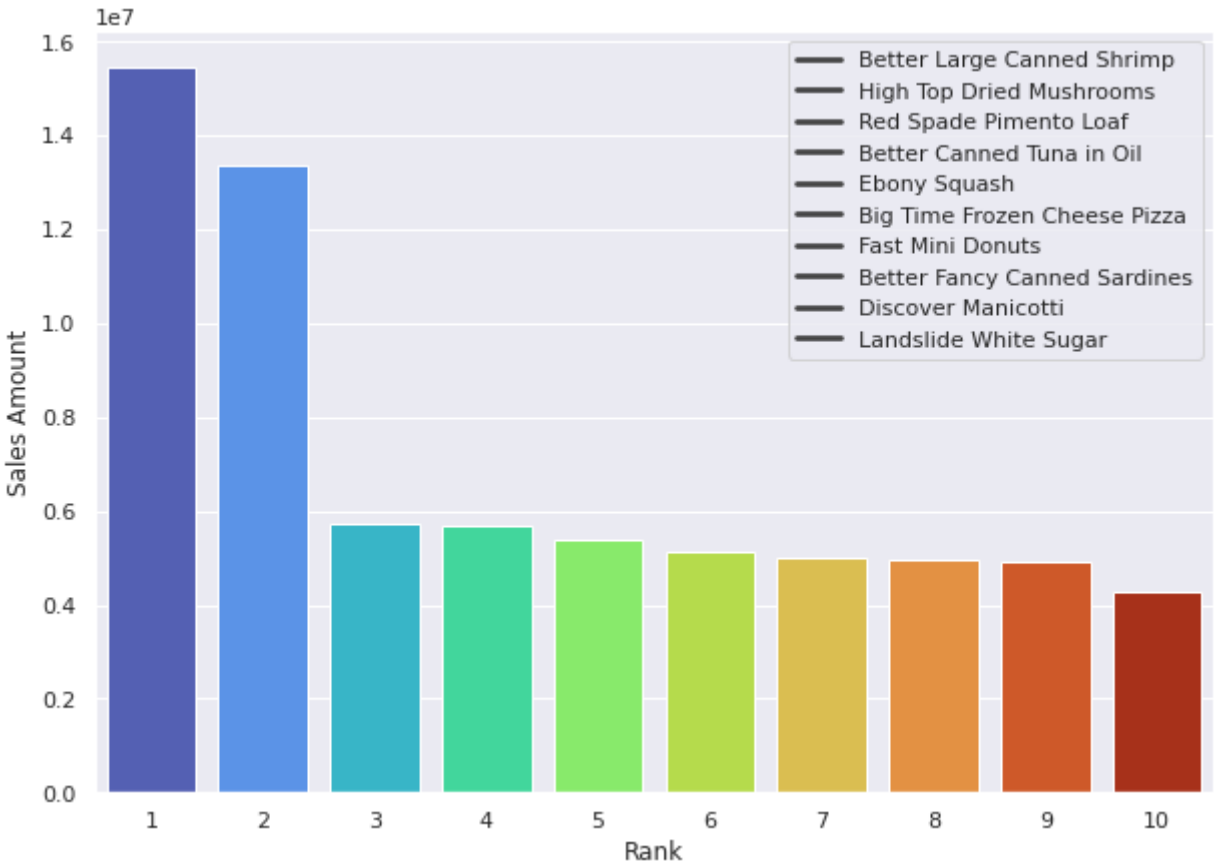


High Sales:

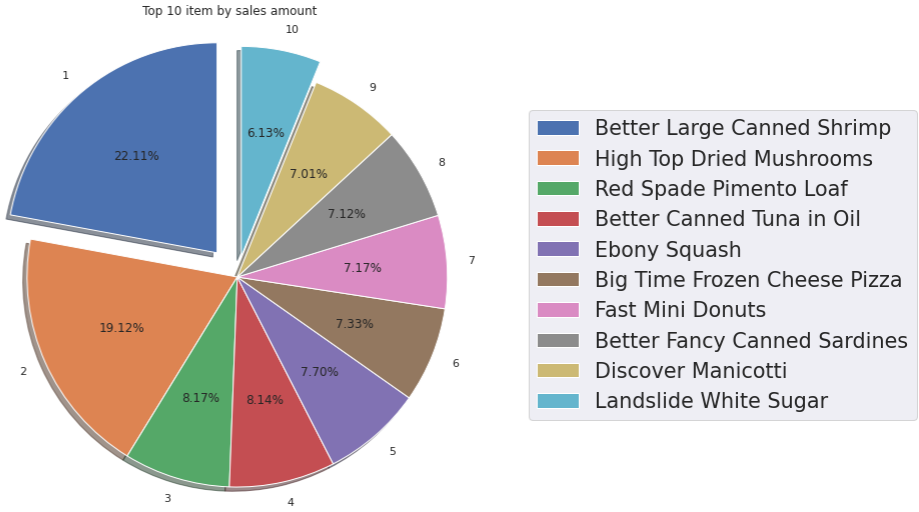
```
high_sales = data01.groupby('Item').sum().sort_values('Sales Amount',ascending = False)
high_sales.index+=1
high_sales=high_sales.reset_index().rename(columns={'index':'Rank'})
```

```
plt.figure(figsize=(10,7))
sns.barplot(x='Rank',y='Sales Amount',data = high_sales.head(10),palette ='turbo')
plt.legend(high_sales['Item'].head(10))
high_sales[['Rank','Item','Sales Amount']].head(10)
```

	Rank	Item	Sales Amount
0	1	Better Large Canned Shrimp	15454172.47
1	2	High Top Dried Mushrooms	13367714.53
2	3	Red Spade Pimento Loaf	5711486.45
3	4	Better Canned Tuna in Oil	5693075.12
4	5	Ebony Squash	5380727.75
5	6	Big Time Frozen Cheese Pizza	5127171.17
6	7	Fast Mini Donuts	5009499.08
7	8	Better Fancy Canned Sardines	4975348.35
8	9	Discover Manicotti	4901139.48
9	10	Landslide White Sugar	4282290.62

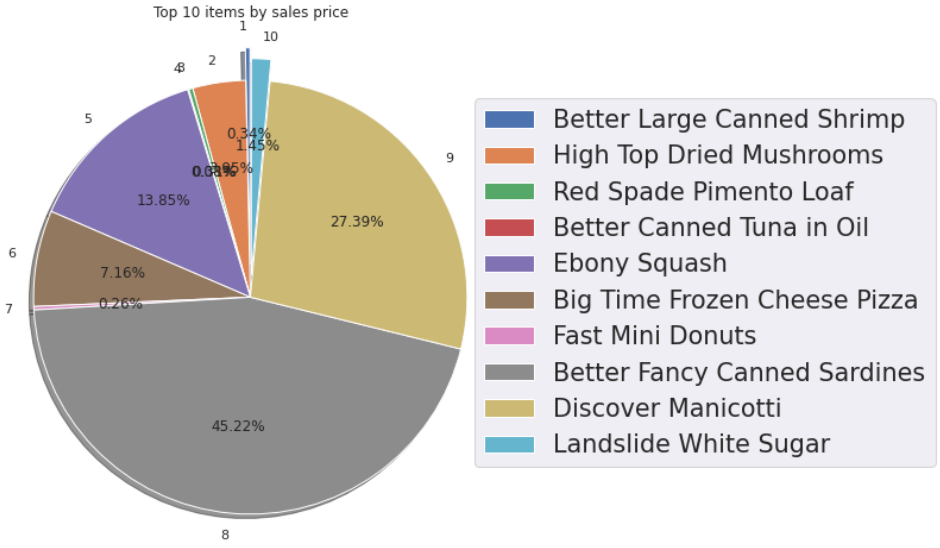



```
plt.figure(figsize=(25,9))
plt.pie('Sales Amount',labels='Rank',data = high_sales.head(10),autopct ='%1.2f%%',
        shadow = True, startangle =90, explode =(0.15,0,0,0,0,0,0,0,0,0.1))
plt.axis('equal')
plt.title('Top 10 item by sales amount')
plt.legend(high_sales['Item'].head(10),loc=7, fontsize ='xx-large')
plt.show()
```




```
hight_sales= data01.groupby('Item').sum().sort_values('Sales Price',ascending=False).re
high_sales.index+=1
hight_sales= hight_sales.reset_index().rename(columns={'index':'rank'})
```

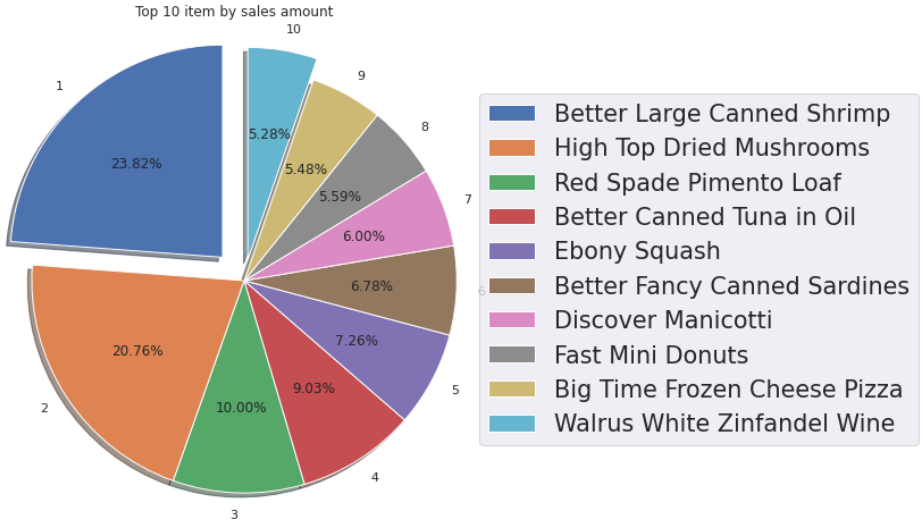
```
plt.figure(figsize=(21,8))
plt.pie('Sales Price', labels ='Rank', data = high_sales.head(10),autopct='%1.2f%%',
       shadow= True, startangle =90, explode = (0.15,0,0,0,0,0,0,0,0,0.1))
plt.axis('equal')
plt.title('Top 10 items by sales price')
plt.legend(high_sales['Item'].head(10),loc=7, fontsize ='xx-large')
plt.show()
```




```
high_cost = data01.groupby('Item').sum().sort_values('Sales Cost Amount',ascending = Fa
high_cost.index+=1
high_cost=high_cost.reset_index().rename(columns={'index':'Rank'})
```

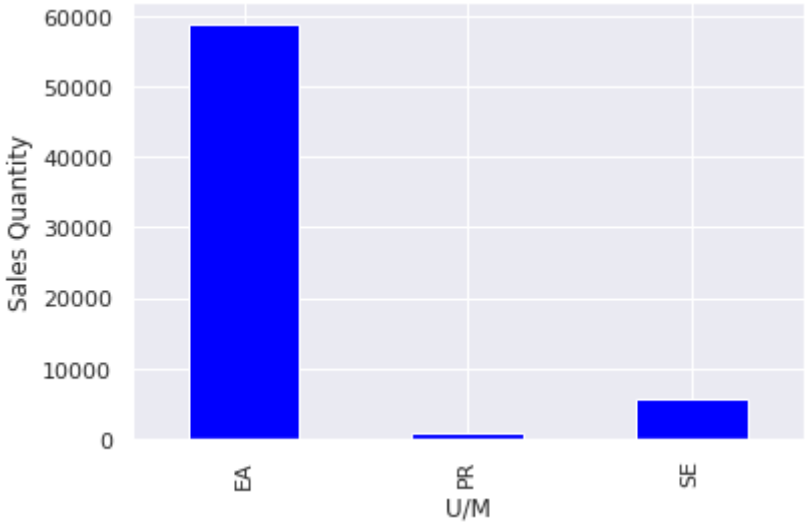
```
plt.figure(figsize=(22,8))
plt.pie('Sales Cost Amount',labels = 'Rank',data = high_cost.head(10),autopct='%1.2f%%'
       shadow = True, startangle =90, explode = (0.15,0,0,0,0,0,0,0,0,0.1))
plt.axis('equal')
plt.title('Top 10 item by sales amount')
```

```
plt.legend(high_cost['Item'].head(10),loc = 7, fontsize='xx-large')
plt.show()
```



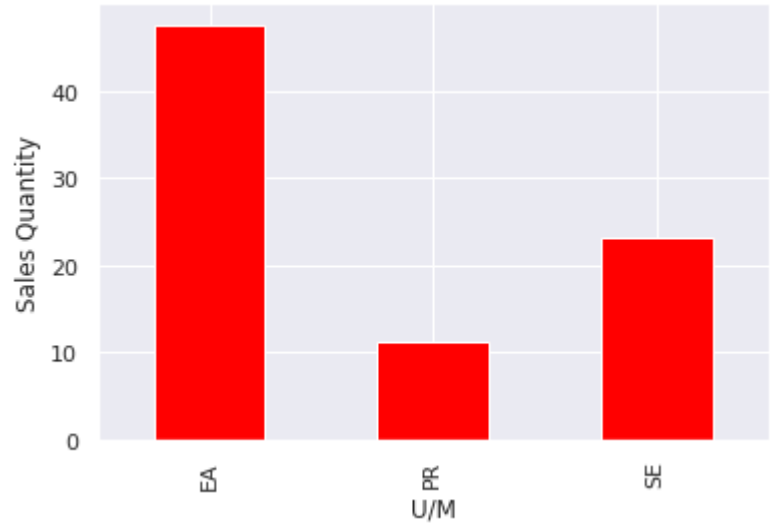

```
data01.groupby('U/M')['Sales Quantity'].count().plot(kind='bar',color ='Blue')
plt.ylabel('Sales Quantity')
```

```
Text(0, 0.5, 'Sales Quantity')
```



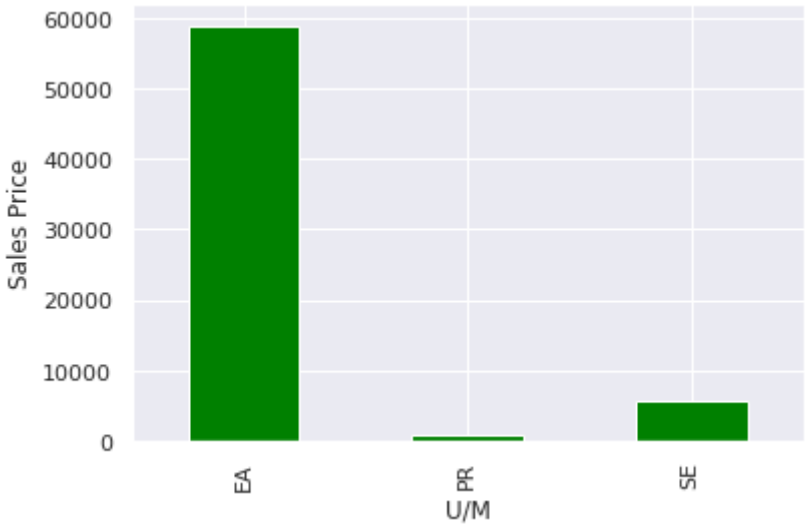

```
data01.groupby('U/M')['Sales Quantity'].mean().plot(kind='bar',color ='Red')
plt.ylabel('Sales Quantity')
```

```
Text(0, 0.5, 'Sales Quantity')
```



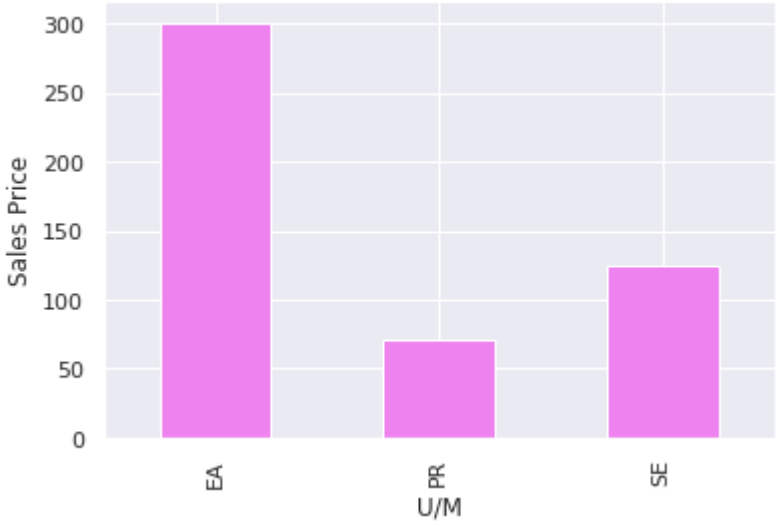

```
data01.groupby('U/M')['Sales Price'].count().plot(kind='bar',color='Green')
plt.ylabel('Sales Price')
```

```
Text(0, 0.5, 'Sales Price')
```



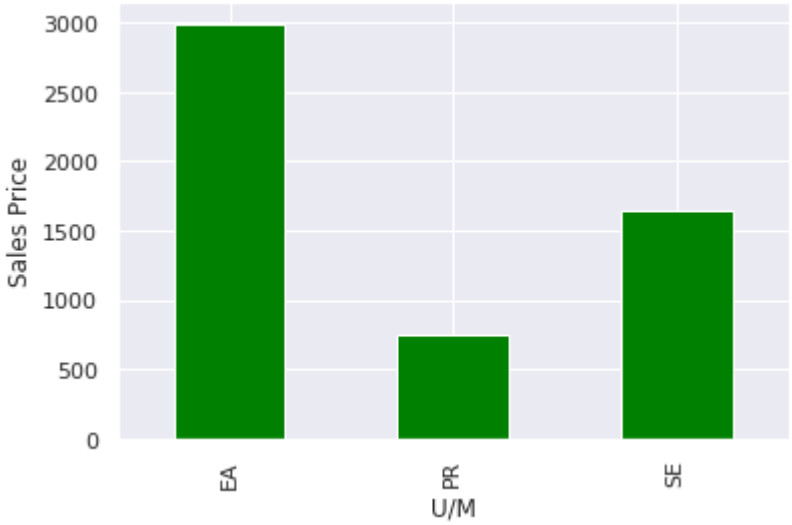

```
data01.groupby('U/M')['Sales Price'].mean().plot(kind='bar',color='violet')
plt.ylabel('Sales Price')
```

```
Text(0, 0.5, 'Sales Price')
```



```
data01.groupby('U/M')['Sales Amount'].mean().plot(kind='bar',color='Green')
plt.ylabel('Sales Price')
```

```
Text(0, 0.5, 'Sales Price')
```




```
data01.groupby('U/M')['Sales Amount'].mean().plot(kind='bar',color='viole')
plt.ylabel('Sales Price')
```