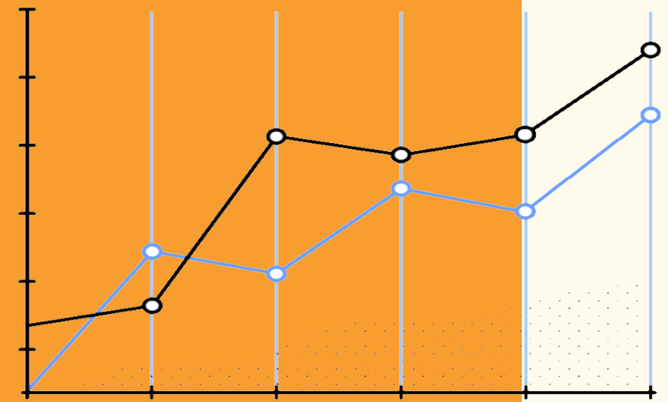
Data from Supermarket Sales

PROJECT 4
DATA VISUALIZATION

Thao & Elnara



1.Our Dataset

- Supermarket Sales
- 1000 rows x 18 columns
- No missing values, No issues
- No duplicates
- One column with repeatable value
- Data is not variable and for short period of time

```
sales = pd.read_csv('supermarket_sales - Sheet1.csv')
sales = sales.round(2)
sales.columns = sales.columns.str.capitalize()
sales.columns = sales.columns.str.replace(' ', '_')
sales.head()
```

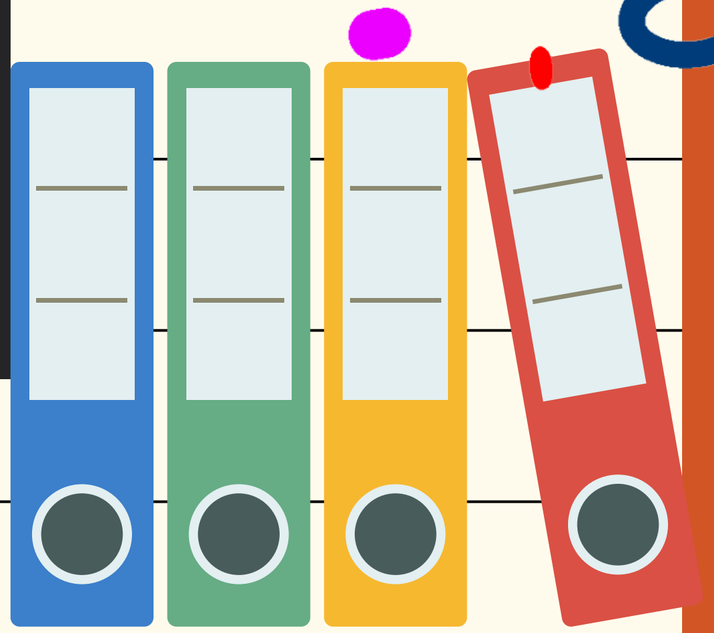
2. Manipulations with columns

Creating new columns for more data

```
sales['Date_formatted'] = pd.to_datetime(sales['Date'])
```

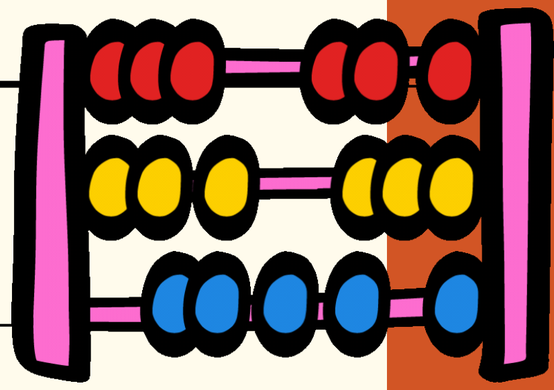
```
sales['season'] = (sales['Date_formatted'].dt.month%12 + 3)//3

seasons = {
            1: 'Winter',
            2: 'Spring',
            3: 'Summer',
            4: 'Autumn'
}

sales['season_name'] = sales['season'].map(seasons)
```

```
sales['Month'] = sales['Date_formatted'].dt.month
sales['Day'] = sales['Date_formatted'].dt.day
```



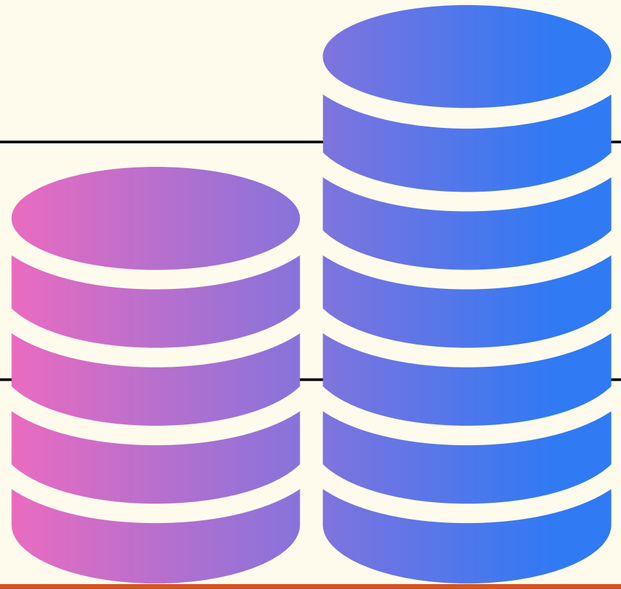
3. Manipulations with columns

Creating new columns for more data



```
sales['Time'] = pd.to_datetime(sales['Time'], format='%H:%M').dt.time
```

```
sales['Hour'] = pd.to_datetime(sales['Time'], format='%H:%M:%S').dt.hour
```



```
b = [0,4,8,12,16,20,24]
l = ['Late Night', 'Early Morning','Morning','Noon','Evening','Night']
sales['time_of_day_draft'] = pd.cut(sales['Hour'], bins=b, labels=l, include_lowest=True)

def f(x):
    if (x > 4) and (x <= 8):
        return 'Early Morning'
    elif (x > 8) and (x <= 12 ):
        return 'Morning'
    elif (x > 12) and (x <= 16):
        return'Noon'
    elif (x > 16) and (x <= 20) :
        return 'Evening'
    elif (x > 20) and (x <= 24):
        return'Night'
    elif (x <= 4):
        return'Late Night'

sales['time_of_day'] = sales['Hour'].apply(f)
```

4. Manipulations with columns

Checking numeric data

```
#checking numeric values in our data
round(sales.describe()[['Unit_price', 'Quantity', 'Tax_5%','Total','Cogs',  'Gross_margin_percentage',
    'Gross_income', 'Rating']], 2)
```

✓ 0.0s Python

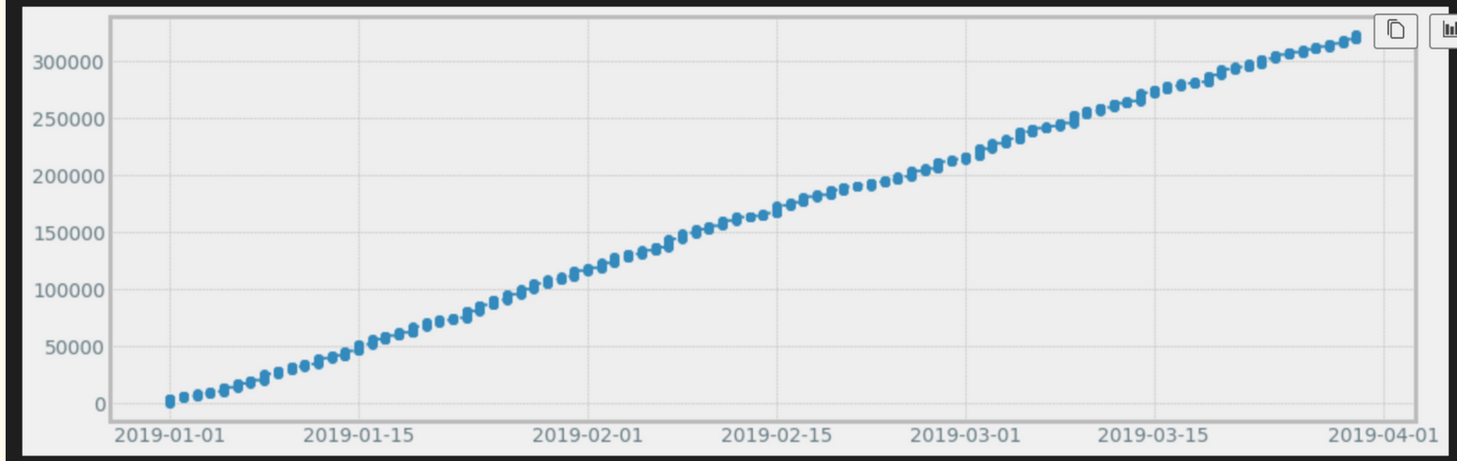
	Unit_price	Quantity	Tax_5%	Total	Cogs	Gross_margin_percentage	Gross_income	Rating
count	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
mean	55.67	5.51	15.38	322.97	307.59	4.76	15.38	6.97
std	26.49	2.92	11.71	245.89	234.18	0.00	11.71	1.72
min	10.08	1.00	0.51	10.68	10.17	4.76	0.51	4.00
25%	32.88	3.00	5.93	124.42	118.50	4.76	5.93	5.50
50%	55.23	5.00	12.09	253.85	241.76	4.76	12.09	7.00
75%	77.94	8.00	22.44	471.35	448.90	4.76	22.44	8.50
max	99.96	10.00	49.65	1042.65	993.00	4.76	49.65	10.00

5. Sales Overview

```python
plt.figure(figsize = (15,5))
plt.plot(sales['Date_formatted'], sales['Cummulative_Sales'], ls = '--', marker ='o')
```

✓ 0.4s Python

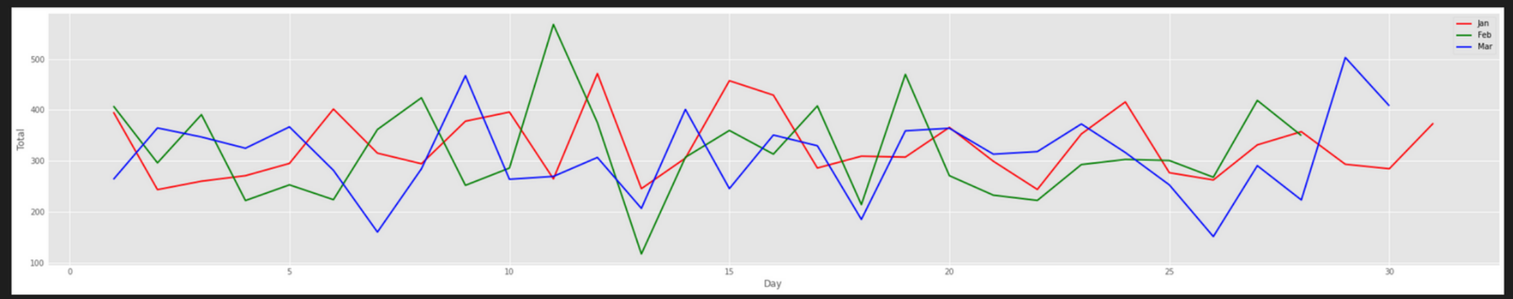
```
[<matplotlib.lines.Line2D at 0x14eade0e0>]
```




```python
plt.style.use('ggplot')
fig, ax = plt.subplots(figsize = (30, 6))

sns.lineplot(data = month_1, x = 'Day', y='Total', color = 'red', errorbar = None)
sns.lineplot(data = month_2, x = 'Day', y='Total', color = 'green', errorbar = None)
sns.lineplot(data = month_3, x = 'Day', y='Total', color = 'blue', errorbar = None)
ax.legend(labels = ['Jan','Feb','Mar'])
```

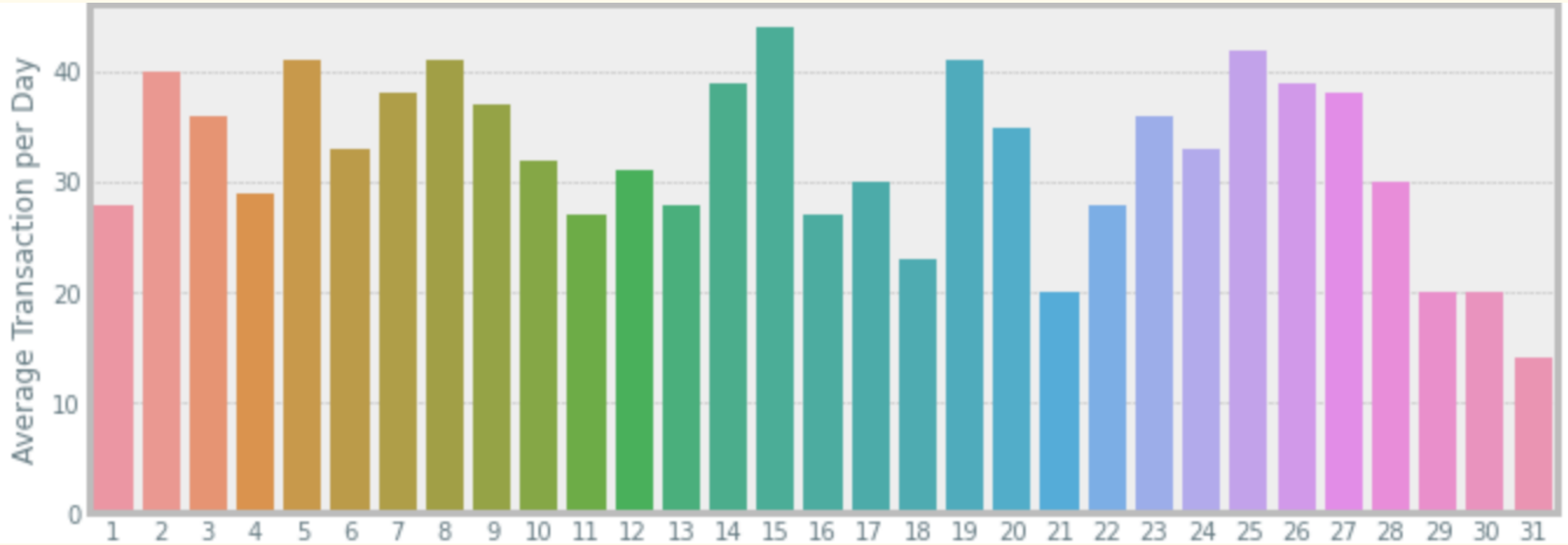
✓ 0.7s Python

```
<matplotlib.legend.Legend at 0x14d3fa0b0>
```



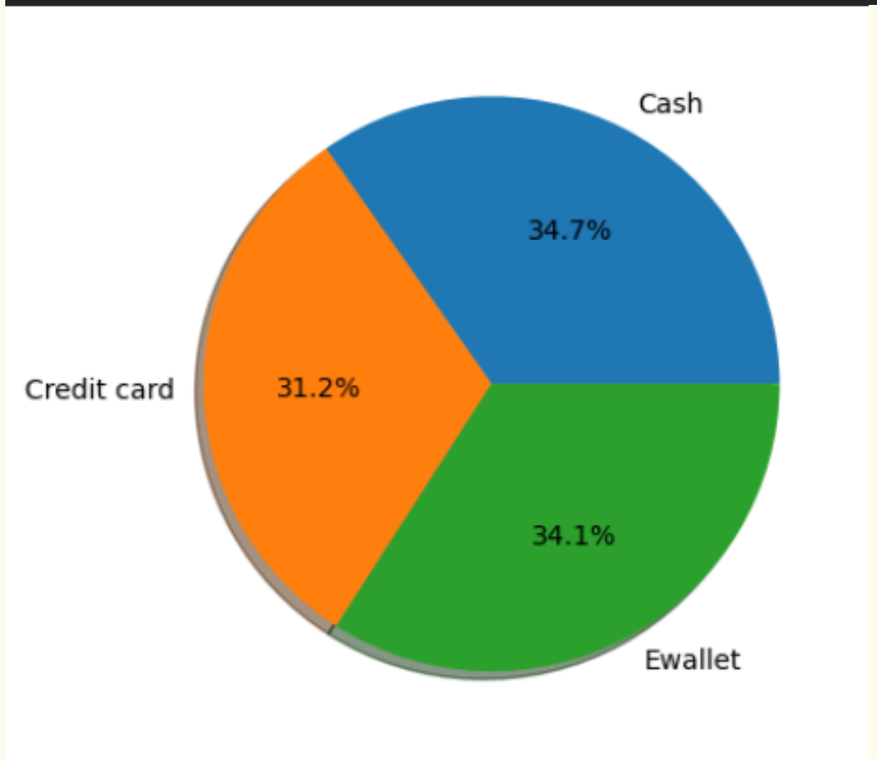
6. Transaction Overview

```
plt.figure(figsize=(10,4))
sns.barplot(data = ave_trans_by_day, x = 'Day', y='Invoice ID')
plt.title('Average Transaction per Day')
plt.xlabel('Day of the Month')
plt.ylabel('Average Transaction per Day')
```



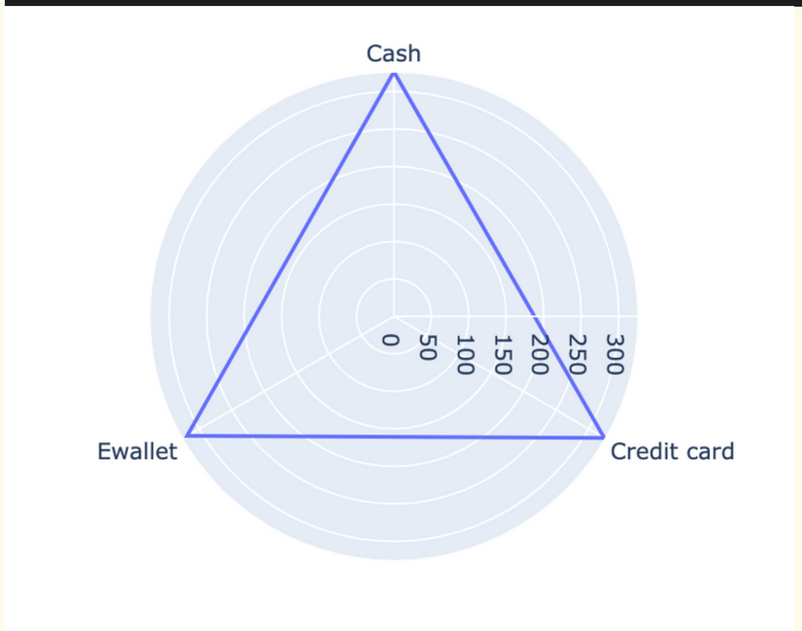
7. Types of payment by Total Sales

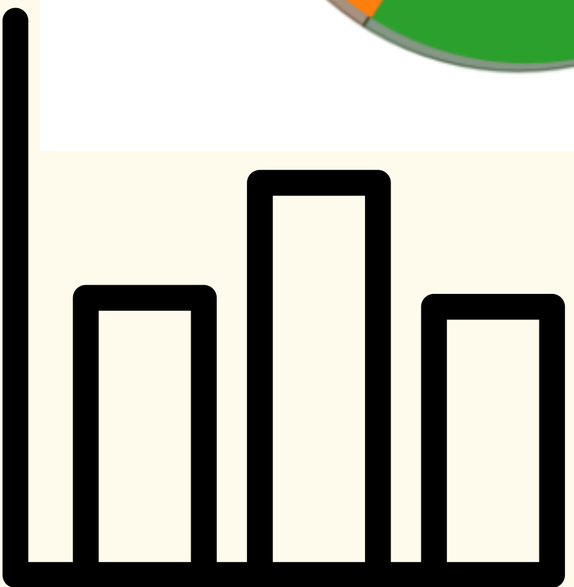
```
labels = ['Cash', 'Credit card', 'Ewallet']
plt.pie(ratio_total_invoice['Total'],autopct='%1.1f%%', labels=labels, shadow=True)
```



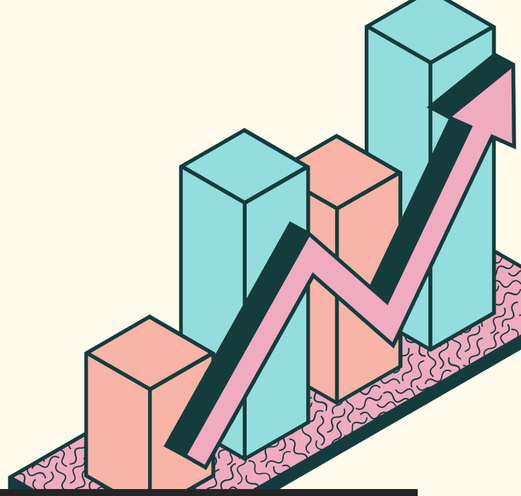

```
import plotly.express as px
import plotly.graph_objects as go

fig = go.Figure()
fig = px.line_polar(sales_per_transaction, r=sales_per_transaction['VPT'],
                    theta=sales_per_transaction['Payment'],
                    line_close=True,
                    width = 500, height = 400)
fig.show()
```

✓ 0.4s

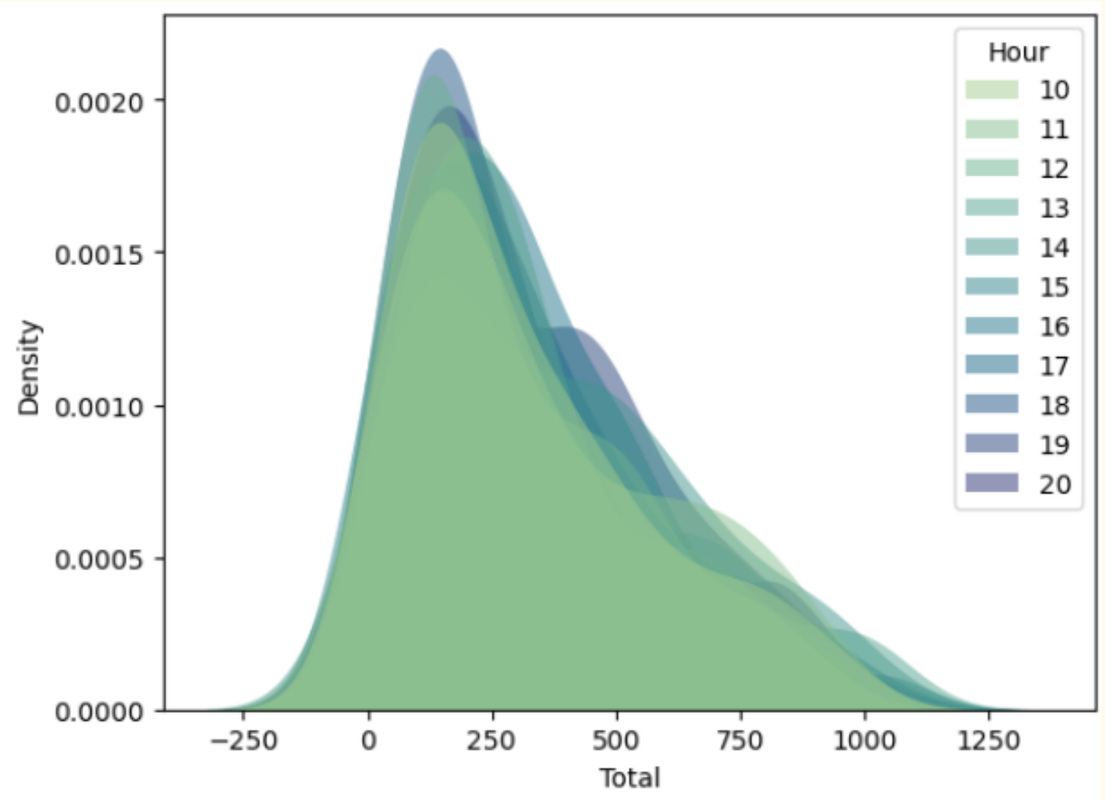




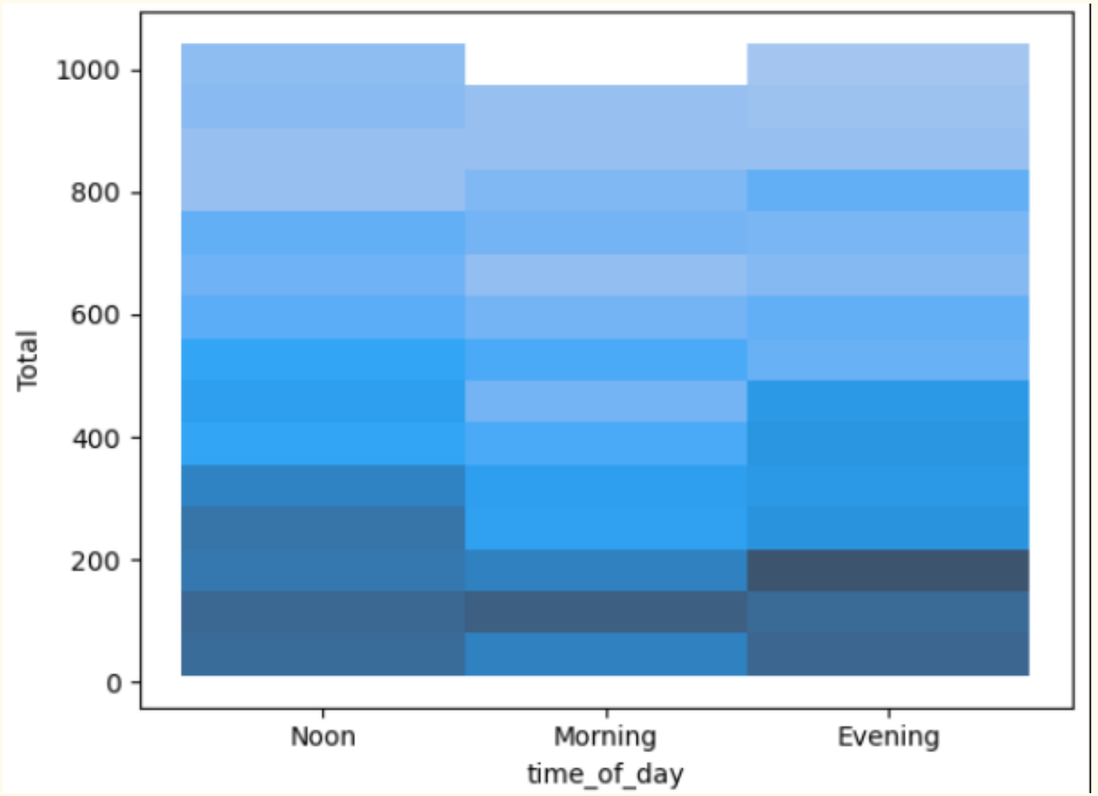
8. Frequencies Values of transaction by time of the day



```python
sns.kdeplot(
    data=sales, x="Total", hue="Hour",
    fill=True, common_norm=False, palette="crest",
    alpha=.5, linewidth=0,
)
```

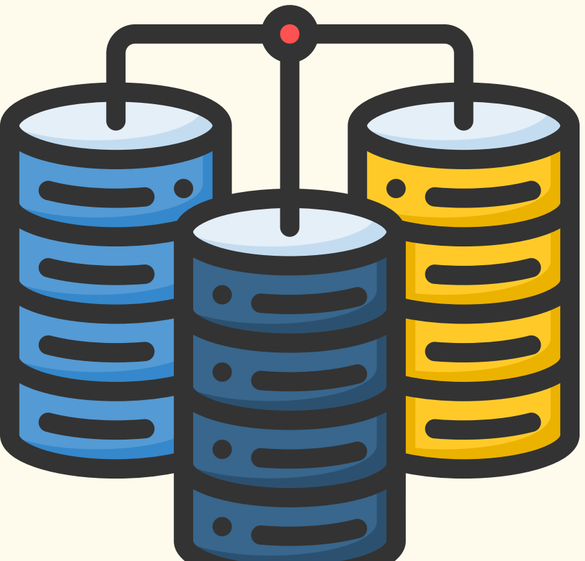


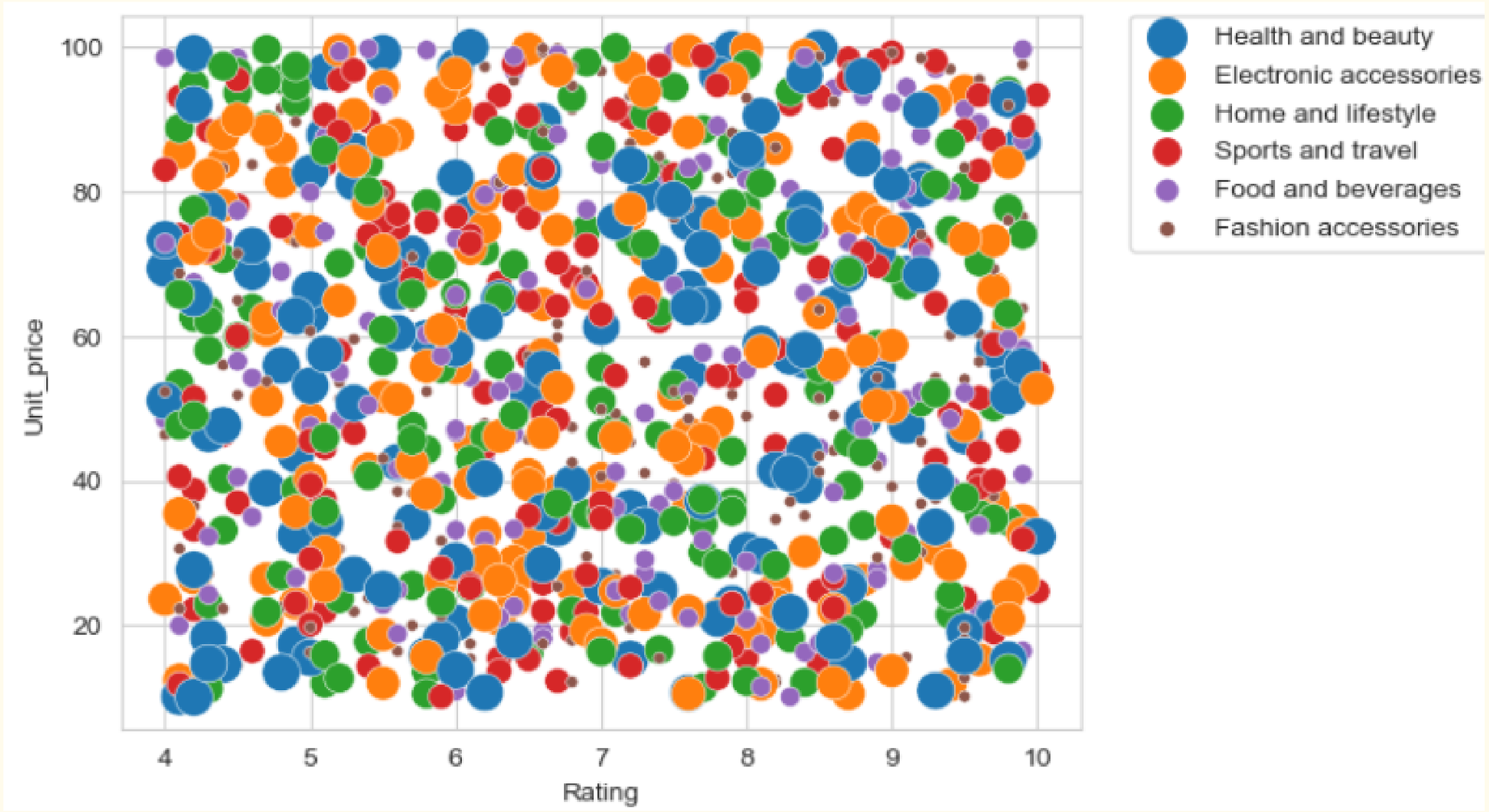

```python
sns.histplot(data=sales, x='time_of_day', y='Total')
```



9. Relation between Unit Price and Item Rating

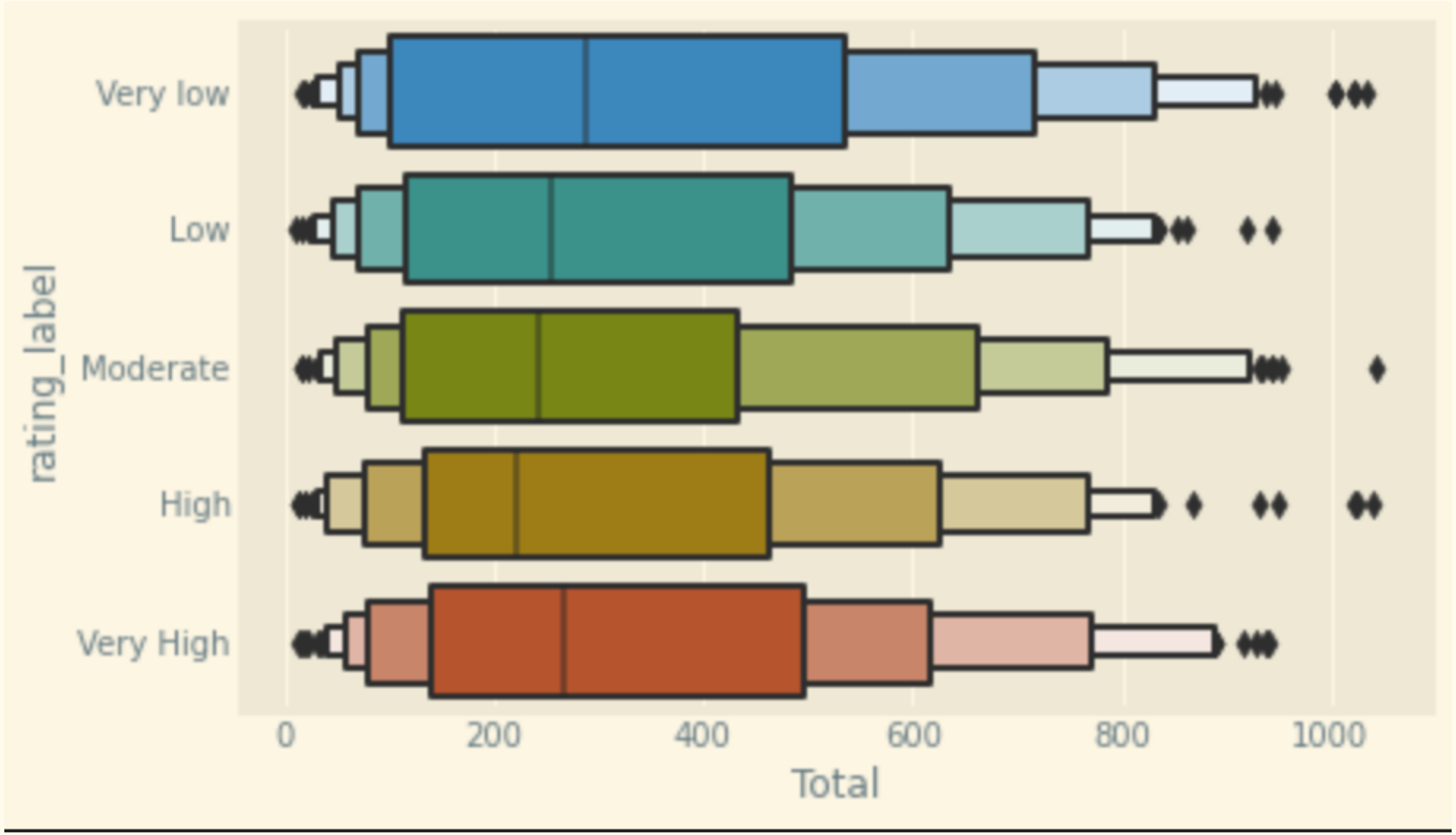
```
sns.scatterplot(data=sales, x ='Rating', y = 'Unit_price', hue = 'Product_line',
size = 'Product_line', sizes=(20,200), legend='full')
plt.legend(bbox_to_anchor=(1.05, 1), loc='upper left', borderaxespad=0)
```





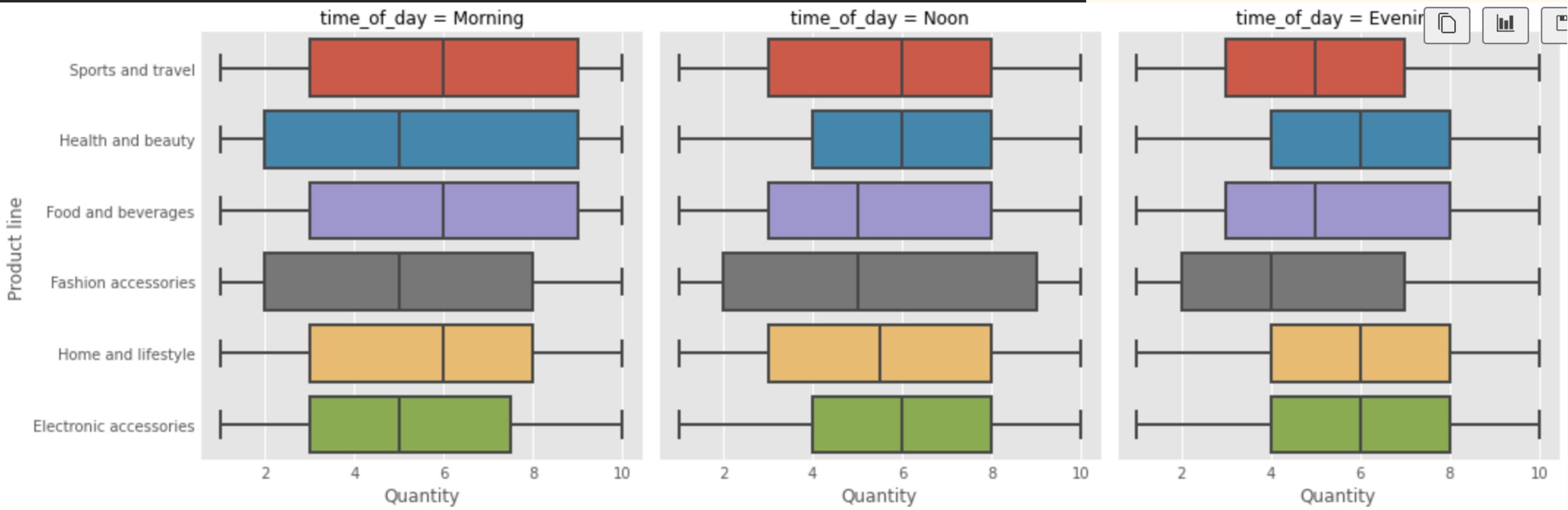
10. Total sales by Rating label

```
sns.boxenplot(data=sales, x="Total", y="rating_label", scale = 'linear')
```



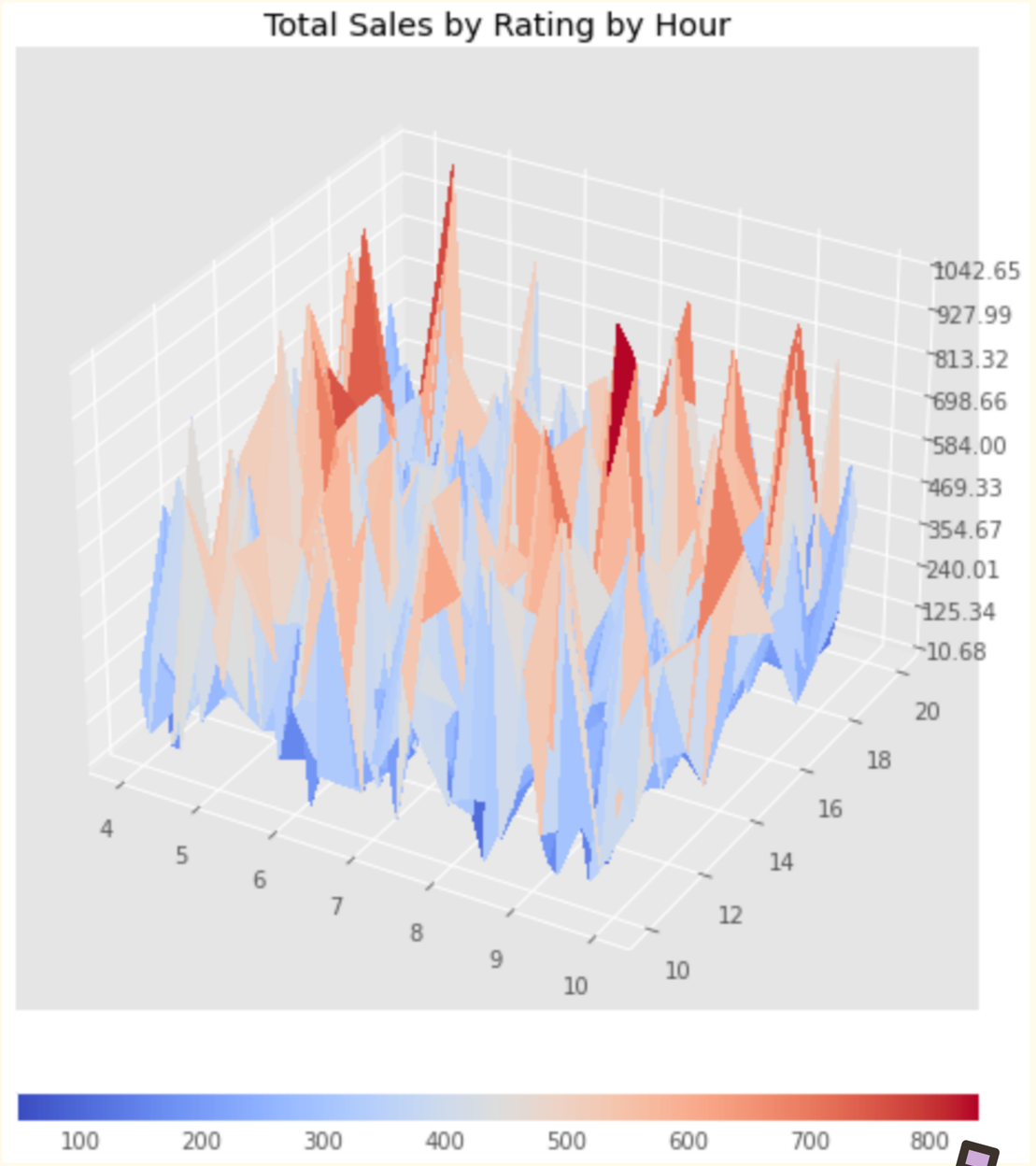
11. Total sales by Rating label

```
sns.catplot(data=sales, x='Quantity', y='Product line', col='time_of_day', kind='box')
```



12. Total sales by Rating by Hour

```
from matplotlib import cm
from matplotlib.ticker import LinearLocator
from mpl_toolkits.axes_grid1.inset_locator import inset_axes

fig, ax = plt.subplots(subplot_kw={"projection": "3d"}, figsize=(8, 8))

X = sales['Rating']
Y = sales['Hour']
Z = sales['Total']

# Plot the surface.
surf = ax.plot_trisurf(X, Y, Z, cmap=cm.coolwarm,
                       linewidth=0, antialiased=False)

# ax.set_zlim(-1.01, 1.01)
ax.zaxis.set_major_locator(LinearLocator(10))
ax.zaxis.set_major_formatter('{x:.02f}')

# Add a color bar which maps values to colors.
axins = inset_axes(ax,
                   width="100%",
                   height="3%",
                   loc='lower center',
                   borderpad=-5
                   )

fig.colorbar(surf, shrink=0.3, aspect=10, cax=axins, orientation="horizontal")
ax.set_title('Total Sales by Rating by Hour')
plt.show()
```



HOW WAS IT?
DID YOU HAVE FUN?

Thank you!