Pizza Sales Analysis

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
```

Import Raw Data

```
df = pd.read_csv("C:\\Users\Rc\\Downloads\\Python Data Analysis\\pizza_sales.csv")
df
```

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category
0	1	1	hawaiian_m	1	01-01-2015	11:38:36	13.25	13.25	M	Classic
1	2	2	classic_dlx_m	1	01-01-2015	11:57:40	16.00	16.00	M	Classic
2	3	2	five_cheese_l	1	01-01-2015	11:57:40	18.50	18.50	L	Veggie
3	4	2	ital_supr_l	1	01-01-2015	11:57:40	20.75	20.75	L	Supreme
4	5	2	mexicana_m	1	01-01-2015	11:57:40	16.00	16.00	M	Veggie

MetaData of Raw Data

```
df.head(6)
```

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category	pizz
0	1	1	hawaiian_m	1	01-01-2015	11:38:36	13.25	13.25	M	Classic	Mo
1	2	2	classic_dlx_m	1	01-01-2015	11:57:40	16.00	16.00	M	Classic	M
2	3	2	five_cheese_l	1	01-01-2015	11:57:40	18.50	18.50	L	Veggie	Moz Pro

```
df.tail()
```

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category
48615	48616	21348	ckn_alfredo_m	1	31-12-2015	21:23:10	16.75	16.75	M	Chicken
48616	48617	21348	four_cheese_l	1	31-12-2015	21:23:10	17.95	17.95	L	Veggie

```
print("The Metadata of the dataset :", df.shape)
```

```
The Metadata of the dataset : (48620, 12)
```

```
print("The Rows of the dataset :", df.shape[0])
```

```
The Rows of the dataset : 48620
```

```
print("The Columns of the dataset :", df.shape[1])
```

```
The Columns of the dataset : 12
```

```
df.columns
```

```
Index(['pizza_id', 'order_id', 'pizza_name_id', 'quantity', 'order_date',
       'order_time', 'unit_price', 'total_price', 'pizza_size',
       'pizza_category', 'pizza_ingredients', 'pizza_name'],
      dtype='object')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48620 entries, 0 to 48619
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   pizza_id           48620 non-null  int64  
 1   order_id           48620 non-null  int64  
 2   pizza_name_id      48620 non-null  object 
 3   quantity           48620 non-null  int64  
 4   order_date         48620 non-null  object 
 5   order_time         48620 non-null  object 
 6   unit_price         48620 non-null  float64
 7   total_price        48620 non-null  float64
 8   pizza_size         48620 non-null  object 
 9   pizza_category     48620 non-null  object 
 10  pizza_ingredients  48620 non-null  object 
 11  pizza_name         48620 non-null  object 
dtypes: float64(2), int64(3), object(7)
memory usage: 4.5+ MB
```

```
df.info
```

```
<bound method DataFrame.info of        pizza_id  order_id  pizza_name_id  quantity  order_date order_time  \
0             1         1     hawaiian_m         1  01-01-2015   11:38:36
1             2         2  classic_dlx_m         1  01-01-2015   11:57:40
2             3         2  five_cheese_l         1  01-01-2015   11:57:40
3             4         2    ital_supr_l         1  01-01-2015   11:57:40
4             5         2     mexicana_m         1  01-01-2015   11:57:40
...         ...       ...            ...       ...         ...        ...
48615     48616     21348  ckn_alfredo_m         1  31-12-2015   21:23:10
48616     48617     21348  four_cheese_l         1  31-12-2015   21:23:10
48617     48618     21348   napolitana_s         1  31-12-2015   21:23:10
48618     48619     21349     mexicana_l         1  31-12-2015   22:09:54
48619     48620     21350      bbq_ckn_s         1  31-12-2015   23:02:05

       unit_price  total_price pizza_size pizza_category  \
0           13.25        13.25          M        Classic
1           16.00        16.00          M        Classic
2           18.50        18.50          L         Veggie
3           20.75        20.75          L        Supreme
4           16.00        16.00          M         Veggie
...           ...          ...        ...            ...
48615       16.75        16.75          M        Chicken
48616       17.95        17.95          L         Veggie
48617       12.00        12.00          S        Classic
48618       20.25        20.25          L         Veggie
48619       12.75        12.75          S        Chicken

                                  pizza_ingredients  \
0                Sliced Ham, Pineapple, Mozzarella Cheese
1      Pepperoni, Mushrooms, Red Onions, Red Peppers,...
2      Mozzarella Cheese, Provolone Cheese, Smoked Go...
3      Calabrese Salami, Capocollo, Tomatoes, Red Oni...
4      Tomatoes, Red Peppers, Jalapeno Peppers, Red O...
...                                                  ...
48615  Chicken, Red Onions, Red Peppers, Mushrooms, A...
48616  Ricotta Cheese, Gorgonzola Piccante Cheese, Mo...
48617  Tomatoes, Anchovies, Green Olives, Red Onions,...
48618  Tomatoes, Red Peppers, Jalapeno Peppers, Red O...
48619  Barbecued Chicken, Red Peppers, Green Peppers,...
```

```
                      pizza_name
0              The Hawaiian Pizza
1        The Classic Deluxe Pizza
2           The Five Cheese Pizza
3       The Italian Supreme Pizza
4              The Mexicana Pizza
...                           ...
48615   The Chicken Alfredo Pizza
48616       The Four Cheese Pizza
48617        The Napolitana Pizza
48618          The Mexicana Pizza
48619  The Barbecue Chicken Pizza

[48620 rows x 12 columns]>
```

DataTypes of the Raw Data

```
df.dtypes
```

```
pizza_id               int64
order_id               int64
pizza_name_id         object
quantity               int64
order_date            object
order_time            object
unit_price           float64
total_price          float64
pizza_size            object
pizza_category        object
pizza_ingredients     object
pizza_name            object
dtype: object
```

```
df.describe()
```

	pizza_id	order_id	quantity	unit_price	total_price
count	48620.000000	48620.000000	48620.000000	48620.000000	48620.000000
mean	24310.500000	10701.479761	1.019622	16.494132	16.821474
std	14035.529381	6180.119770	0.143077	3.621789	4.437398
min	1.000000	1.000000	1.000000	9.750000	9.750000
25%	12155.750000	5337.000000	1.000000	12.750000	12.750000
50%	24310.500000	10682.500000	1.000000	16.500000	16.500000
75%	36465.250000	16100.000000	1.000000	20.250000	20.500000
max	48620.000000	21350.000000	4.000000	35.950000	83.000000

KPI's

```
total_revenue = df['total_price'].sum()
total_pizzas_sold = df['quantity'].sum()
total_orders = df['order_id'].nunique()
avg_order_value = total_revenue/total_orders
avg_pizzas_per_order = total_pizzas_sold/total_orders
print(f"Total Revenue:${total_revenue:,.2f}")
print(f"Total Pizzas Sold:{total_pizzas_sold:,}")
print(f"Total Orders:{total_orders:,}")
print(f"Avg Order Value :${avg_order_value:,.2f}")
print(f"Avg Pizzas Per Order:{avg_pizzas_per_order:,.2f}")
```

```
Total Revenue:$817,860.05
Total Pizzas Sold:49,574
Total Orders:21,350
Avg Order Value :$38.31
Avg Pizzas Per Order:2.32
```

Charts

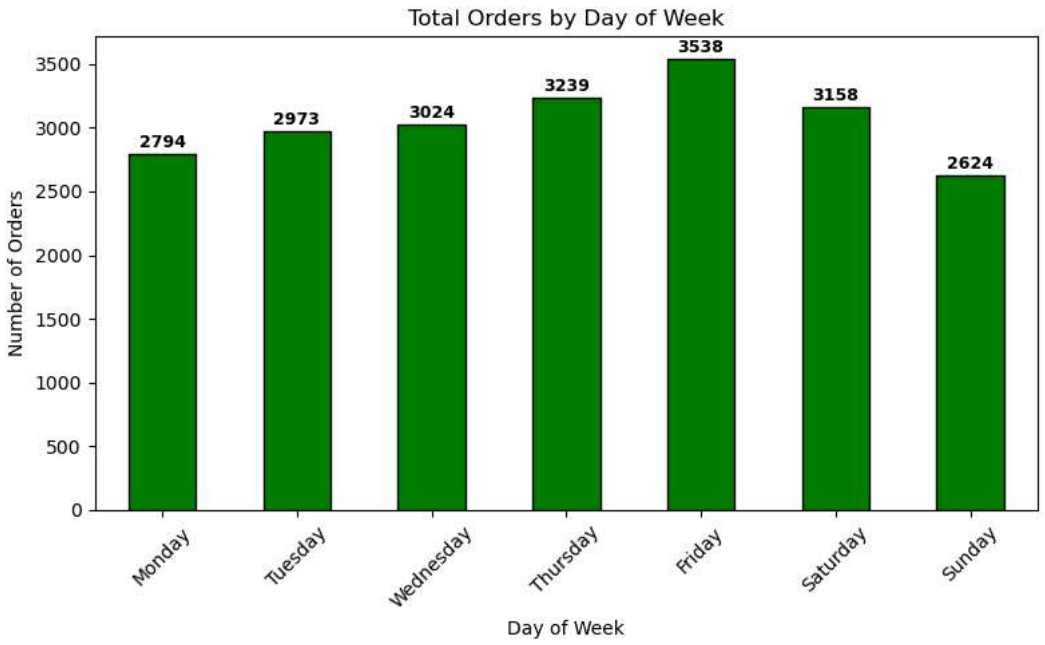
Ingrediant Analysis

```
ingredient = (
    df['pizza_ingredients']
    .str.split(',')
    .explode()
    .str.strip()
    .value_counts()
    .reset_index()
    .rename(columns = {'index':'Count','pizza_ingredients':'Ingredients'})
)
print(ingredient.head(15))
```

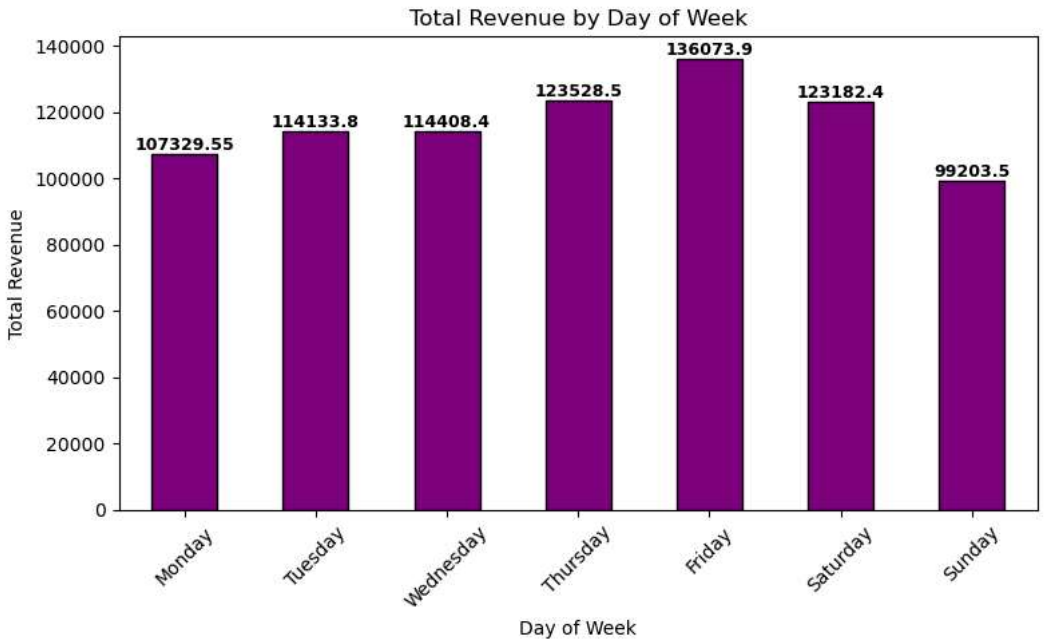
```
          Ingredients  count
0              Garlic  27422
1            Tomatoes  26601
2          Red Onions  19547
3         Red Peppers  16284
4   Mozzarella Cheese  10333
5           Pepperoni  10300
6             Spinach  10012
7           Mushrooms   9624
8             Chicken   8443
9           Capocollo   6572
10       Green Olives   6174
11         Artichokes   5682
12   Jalapeno Peppers   5643
13      Green Peppers   5224
14        Feta Cheese   4748
```

Daily Trend - Total Orders

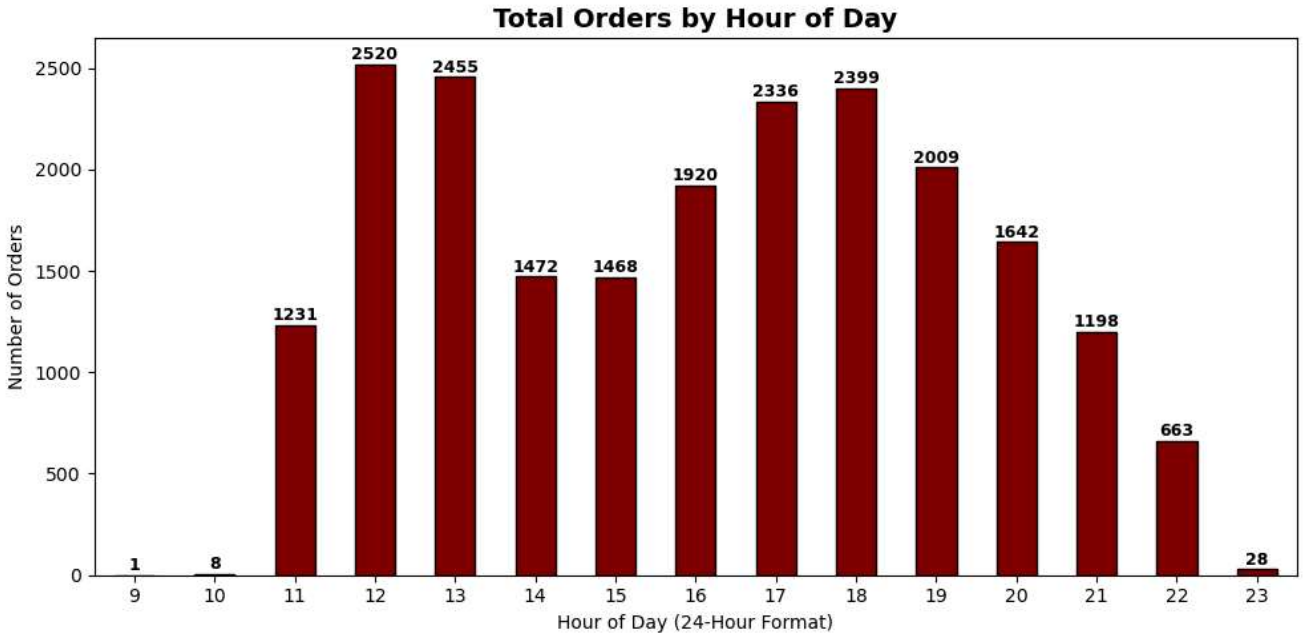
```
import pandas as pd
import matplotlib.pyplot as plt

# Convert 'order_date' to datetime
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)

# Extract day name
df['day_name'] = df['order_date'].dt.day_name()

# Define weekday order
weekday_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

# Make 'day_name' an ordered categorical variable
df['day_name'] = pd.Categorical(df['day_name'], categories=weekday_order, ordered=True)

# Group by day_name and count unique order IDs
orders_by_day = df.groupby('day_name', observed=False)['order_id'].nunique()

# Plot
ax = orders_by_day.plot(kind='bar', figsize=(8, 5), color='green', edgecolor='black')

plt.title("Total Orders by Day of Week")
plt.xlabel("Day of Week")
plt.ylabel("Number of Orders")
plt.xticks(rotation=45)

# Add value labels on bars
for i, val in enumerate(orders_by_day):
    plt.text(i, val + 20, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```




```
import pandas as pd
import matplotlib.pyplot as plt

# Convert 'order_date' to datetime
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)

# Extract day name
df['day_name'] = df['order_date'].dt.day_name()

# Define weekday order
weekday_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

# Make 'day_name' an ordered categorical variable
df['day_name'] = pd.Categorical(df['day_name'], categories=weekday_order, ordered=True)

# Group by day_name and count unique order IDs
orders_by_day = df.groupby('day_name', observed=False)['total_price'].sum()

# Plot
ax = orders_by_day.plot(kind='bar', figsize=(8, 5), color='purple', edgecolor='black')

plt.title("Total Revenue by Day of Week")
plt.xlabel("Day of Week")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)

# Add value labels on bars
for i, val in enumerate(orders_by_day):
    plt.text(i, val + 20, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```

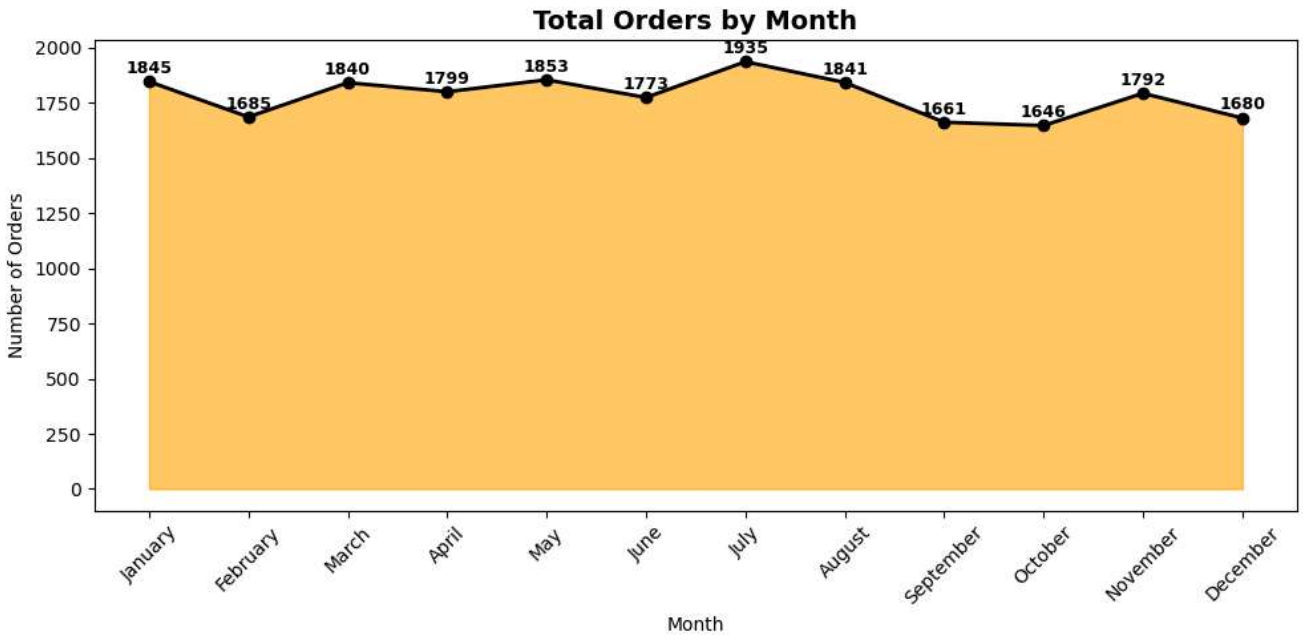



```
import pandas as pd
import matplotlib.pyplot as plt

# Convert order_time to datetime format
df['order_time'] = pd.to_datetime(df['order_time'], format='%H:%M:%S')

# Extract hour from order_time
df['order_hour'] = df['order_time'].dt.hour

# Group by hour and count unique order IDs
orders_by_hour = df.groupby('order_hour', observed=False)['order_id'].nunique()

# Plot bar chart
ax = orders_by_hour.plot(
    kind='bar',
    figsize=(10, 5),
    color='maroon',
    edgecolor='black'
)

# Add titles and labels
plt.title("Total Orders by Hour of Day", fontsize=14, fontweight='bold')
plt.xlabel("Hour of Day (24-Hour Format)")
plt.ylabel("Number of Orders")
plt.xticks(rotation=0)

# Add value labels on each bar
for i, val in enumerate(orders_by_hour):
    plt.text(i, val + 5, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

# Adjust layout and display plot (outside the loop)
plt.tight_layout()
plt.show()
```



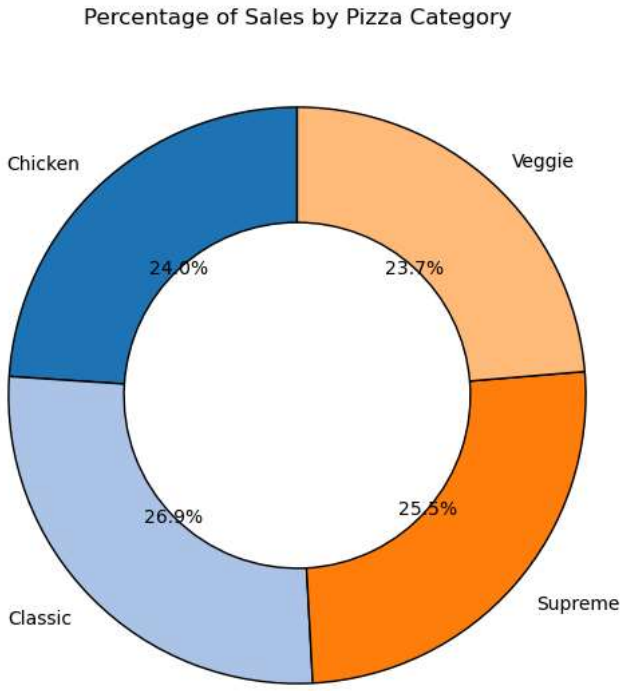
Monthly Trend -Total orders

```
import pandas as pd
import matplotlib.pyplot as plt

# Convert 'order_date' to datetime format
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)

# Extract month name
df['month_name'] = df['order_date'].dt.month_name()

# Define correct month order
month_order = ["January", "February", "March", "April", "May", "June",
               "July", "August", "September", "October", "November", "December"]

# Convert month_name to categorical type with ordered months
df['month_name'] = pd.Categorical(df['month_name'], categories=month_order, ordered=True)

# Group by month and count unique order IDs
orders_by_month = df.groupby('month_name', observed=False)['order_id'].nunique()

# Create figure
plt.figure(figsize=(10, 5))

# Fill and plot line chart
plt.fill_between(orders_by_month.index, orders_by_month.values, color="orange", alpha=0.6)
plt.plot(orders_by_month.index, orders_by_month.values, color="black", linewidth=2, marker='o')

# Add title and labels
plt.title("Total Orders by Month", fontsize=14, fontweight='bold')
plt.xlabel("Month")
plt.ylabel("Number of Orders")
plt.xticks(rotation=45)

# Add value labels above points
for i, val in enumerate(orders_by_month.values):
    plt.text(i, val + 20, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

# Adjust layout and show
plt.tight_layout()
plt.show()
```



% of Sales Category

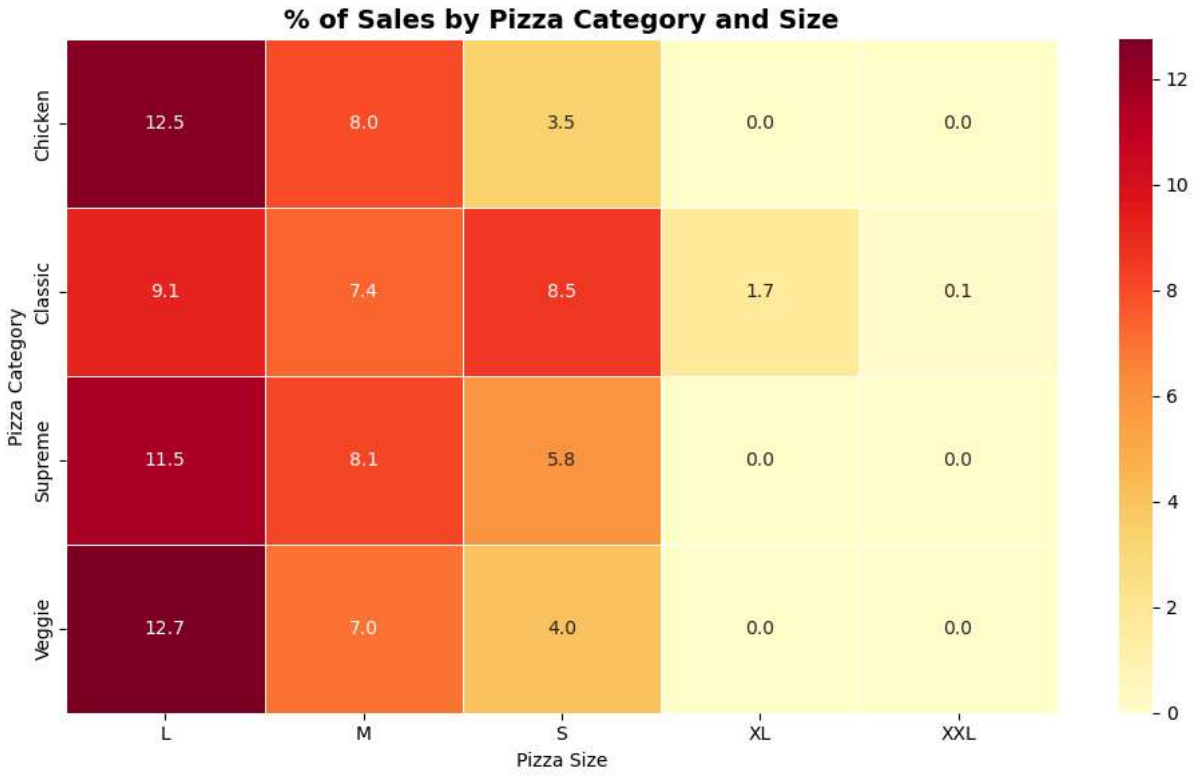
```
category_sales = df.groupby('pizza_category')['total_price'].sum()
category_pct = category_sales /category_sales.sum() *100
plt.figure(figsize =(7,7))
colors = plt.get_cmap('tab20').colors
plt.pie(category_pct,labels = category_pct.index,autopct = '%1.1f%%',startangle = 90, colors = colors ,wedgeprops = {'edgecolor
plt.title("Percentage of Sales by Pizza Category")
plt.show()
```



% Sales by Pizza Size & Category

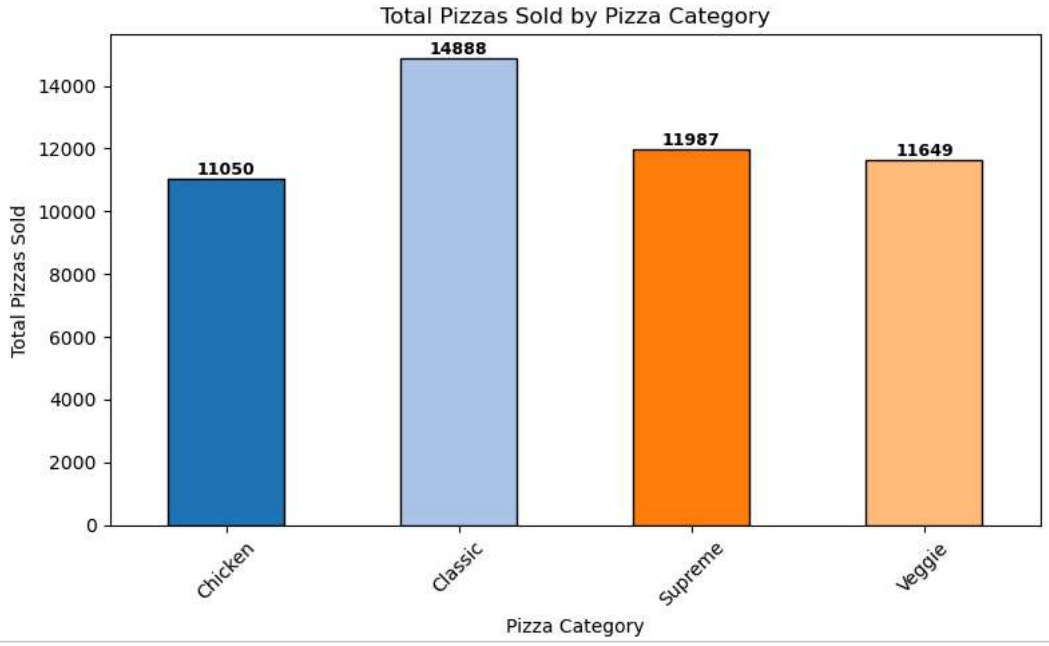
```
import pandas as pd
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Create pivot table
sales_pivot = df.pivot_table(
    index='pizza_category',
    columns='pizza_size',
    values='total_price',
    aggfunc='sum',
    fill_value=0
)

# Calculate percentage of total sales
sales_pct = sales_pivot / sales_pivot.sum().sum() * 100

# Plot heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(sales_pct, annot=True, fmt=".1f", cmap="YlOrRd", linewidths=0.5)

# Add titles and labels
plt.title("% of Sales by Pizza Category and Size", fontsize=14, fontweight='bold')
plt.ylabel("Pizza Category")
plt.xlabel("Pizza Size")

# Show the plot
plt.tight_layout()
plt.show()
```



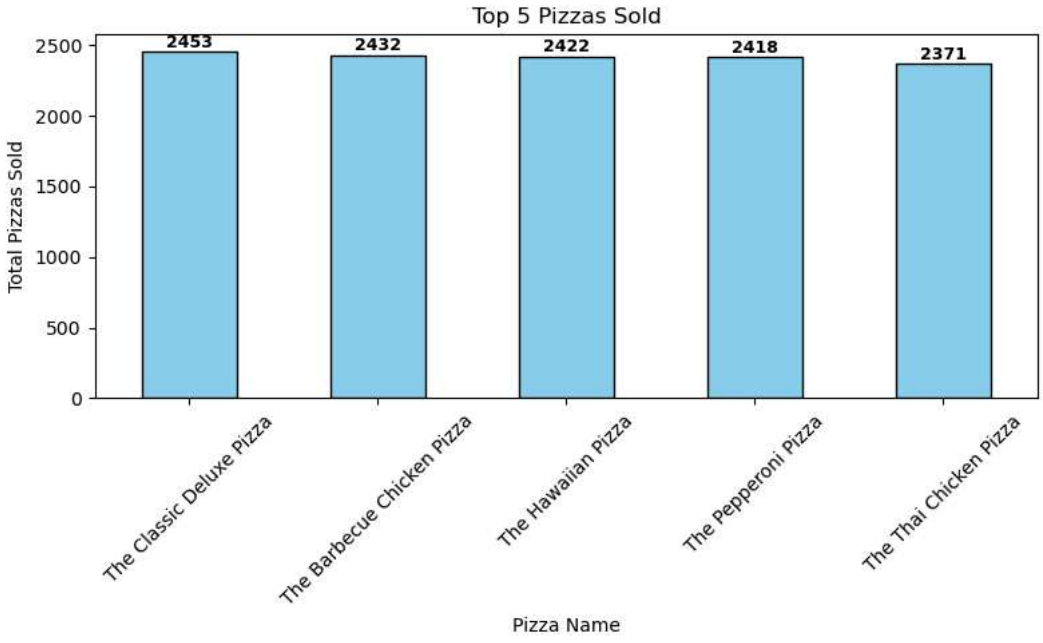
Total Pizzas Sold by Pizza category

```
pizzas_by_category = df.groupby('pizza_category')['quantity'].sum()
colors = list(plt.get_cmap('tab20').colors)
colors = colors[:len(pizzas_by_category)]
ax = pizzas_by_category.plot(kind='bar', figsize=(8,5), color=colors, edgecolor='black')
plt.title("Total Pizzas Sold by Pizza Category")
plt.xlabel("Pizza Category")
plt.ylabel("Total Pizzas Sold")
plt.xticks(rotation=45)
for i, val in enumerate(pizzas_by_category):
    plt.text(i, val + 5, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')
plt.tight_layout()
plt.show()
```



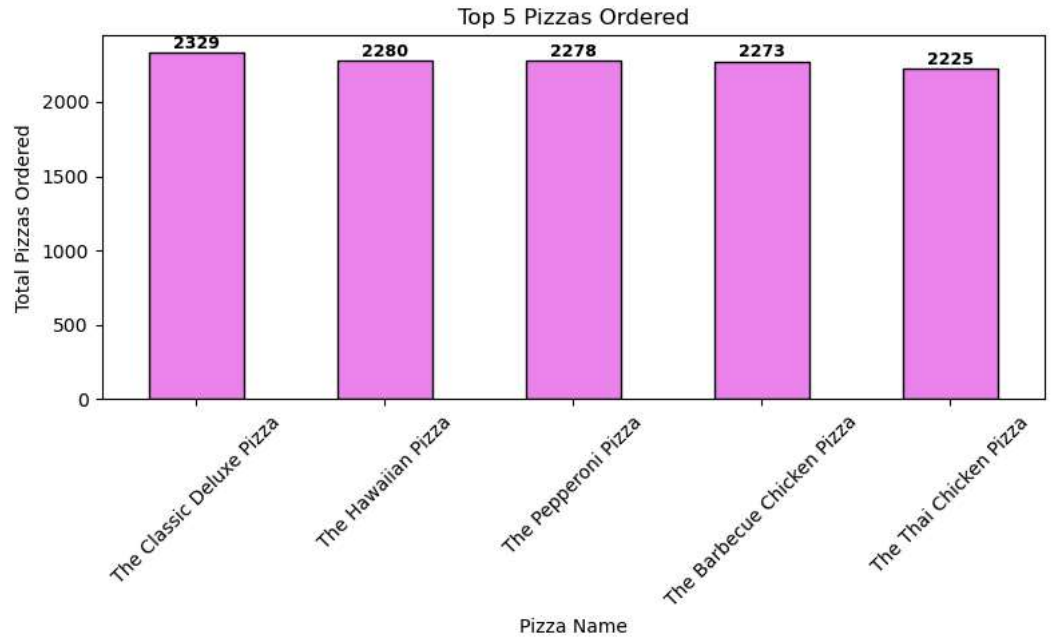
Top 5 Best Selling Pizzas -Total Quantity

```
pizzas_by_name = df.groupby('pizza_name')['quantity'].sum()
top5 = pizzas_by_name.sort_values(ascending=False).head(5)

ax = top5.plot(kind='bar', figsize=(8,5), color='skyblue', edgecolor='black')

plt.title("Top 5 Pizzas Sold")
plt.xlabel("Pizza Name")
plt.ylabel("Total Pizzas Sold")
plt.xticks(rotation=45)

for i, val in enumerate(top5):
    plt.text(i, val + 2, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```



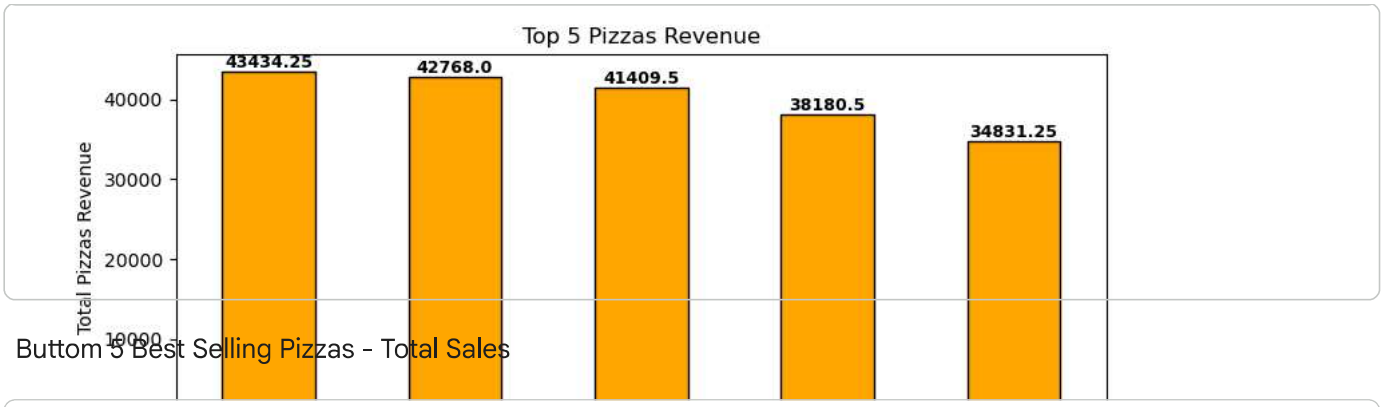
Top 5 Best Selling Pizzas - Total Order

```
pizzas_by_name = df.groupby('pizza_name')['order_id'].nunique()
top5 = pizzas_by_name.sort_values(ascending=False).head(5)

ax = top5.plot(kind='bar', figsize=(8,5), color='violet', edgecolor='black')

plt.title("Top 5 Pizzas Ordered")
plt.xlabel("Pizza Name")
plt.ylabel("Total Pizzas Ordered")
plt.xticks(rotation=45)

for i, val in enumerate(top5):
    plt.text(i, val + 2, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```



Top 5 Best Selling Pizzas - Total Sales

```
pizzas_by_name = df.groupby('pizza_name')['total_price'].sum()
top5 = pizzas_by_name.sort_values(ascending=False).head(5)

ax = top5.plot(kind='bar', figsize=(8,5), color='orange', edgecolor='black')

plt.title("Top 5 Pizzas Revenue")
plt.xlabel("Pizza Name")
plt.ylabel("Total Pizzas Revenue")
plt.xticks(rotation=45)

for i, val in enumerate(top5):
    plt.text(i, val + 2, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```



Buttom 5 Best Selling Pizzas - Total Sales

```
pizzas_by_name = df.groupby('pizza_name')['total_price'].sum()
bottom5 = pizzas_by_name.sort_values(ascending=True).head(5)

ax = bottom5.plot(kind='bar', figsize=(8,5), color='brown', edgecolor='black')

plt.title("Bottom 5 Pizzas Sold")
plt.xlabel("Pizza Name")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)

for i, val in enumerate(bottom5):
    plt.text(i, val + 2, str(val), ha='center', va='bottom', fontsize=9, fontweight='bold')

plt.tight_layout()
plt.show()
```