PIZZA SALES ANALYSIS

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
```

IMPORTIG RAW DATA

In [2]:
```
df = pd.read_csv(r"C:\Users\ashis\Downloads\Pizza Sales - Python Project\pizza_sales.csv")
```

In [3]:
```
df
```

Out[3]:

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category	pizza
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	Mu
2	3	2	five_cheese_l	1	01-01-2015	11:57:40	18.50	18.50	L	Veggie	Ch
3	4	2	ital_supr_l	1	01-01-2015	11:57:40	20.75	20.75	L	Supreme	Cala T
4	5	2	mexicana_m	1	01-01-2015	11:57:40	16.00	16.00	M	Veggie	T Pepp Pep
...	...	...	...	...	...	...	...	...	...	...	
48615	48616	21348	ckn_alfredo_m	1	31-12-2015	21:23:10	16.75	16.75	M	Chicken	Mu
48616	48617	21348	four_cheese_l	1	31-12-2015	21:23:10	17.95	17.95	L	Veggie	R Pic
48617	48618	21348	napolitana_s	1	31-12-2015	21:23:10	12.00	12.00	S	Classic	Anc
48618	48619	21349	mexicana_l	1	31-12-2015	22:09:54	20.25	20.25	L	Veggie	T Pepp Pep
48619	48620	21350	bbq_ckn_s	1	31-12-2015	23:02:05	12.75	12.75	S	Chicken	P

48620 rows × 12 columns

META DATA OF RAW DATA

In [5]:
```
df.head()
```

Out[5]:

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category	pizza_ing
0	1	1	hawaiian_m	1	01-01-2015	11:38:36	13.25	13.25	M	Classic	Sli P M
1	2	2	classic_dlx_m	1	01-01-2015	11:57:40	16.00	16.00	M	Classic	P Mushroo Oni P
2	3	2	five_cheese_l	1	01-01-2015	11:57:40	18.50	18.50	L	Veggie	M F Cheese
3	4	2	ital_supr_l	1	01-01-2015	11:57:40	20.75	20.75	L	Supreme	Calabres C Tomat
4	5	2	mexicana_m	1	01-01-2015	11:57:40	16.00	16.00	M	Veggie	Tomat Peppers, Peppers,

In [6]:
```
df.tail()
```

Out[6]:

	pizza_id	order_id	pizza_name_id	quantity	order_date	order_time	unit_price	total_price	pizza_size	pizza_category	pizz
48615	48616	21348	ckn_alfredo_m	1	31-12-2015	21:23:10	16.75	16.75	M	Chicken	M
48616	48617	21348	four_cheese_l	1	31-12-2015	21:23:10	17.95	17.95	L	Veggie	R Pic
48617	48618	21348	napolitana_s	1	31-12-2015	21:23:10	12.00	12.00	S	Classic	Anc
48618	48619	21349	mexicana_l	1	31-12-2015	22:09:54	20.25	20.25	L	Veggie	T Pepp Pep
48619	48620	21350	bbq_ckn_s	1	31-12-2015	23:02:05	12.75	12.75	S	Chicken	P

In [7]:
```
print("The metadata of the dataset :",df.shape)
```

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

In [9]:
```
print("The rows of the dataset :",df.shape[0])
```

```
The rows of the dataset : 48620
```

In [10]:
```
print("The rows of the dataset :",df.shape[1])
```

```
The rows of the dataset : 12
```

In [11]:
```
df.columns
```

Out[11]:
```
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')
```

In [13]:
```
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
```

Data Types in raw data

In [14]:
```
df.dtypes
```

Out[14]:
```
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
```

In [15]:
```
df.describe()
```

Out[15]:

	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

In [20]:
```
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
```

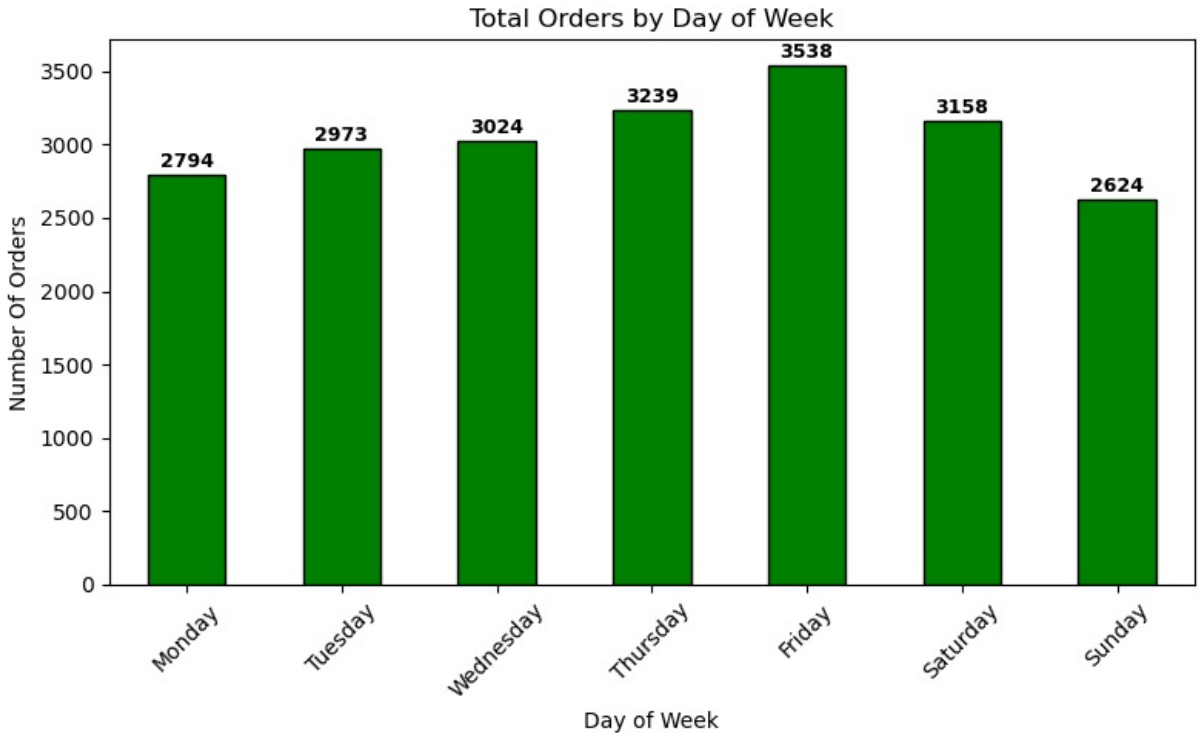
CHARTS

In [23]:
```
ingredient = (
                df['pizza_ingredients']
                .str.split(',')
                .explode()
                .str.strip()
                .value_counts()
                .reset_index()
                .rename(columns = {'index' : 'counts', 'pizza_ingredients' : 'Ingredients'})
)
print(ingredient.head(10))
```

```
        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
```

DAILY TREND - TOTAL ORDERS

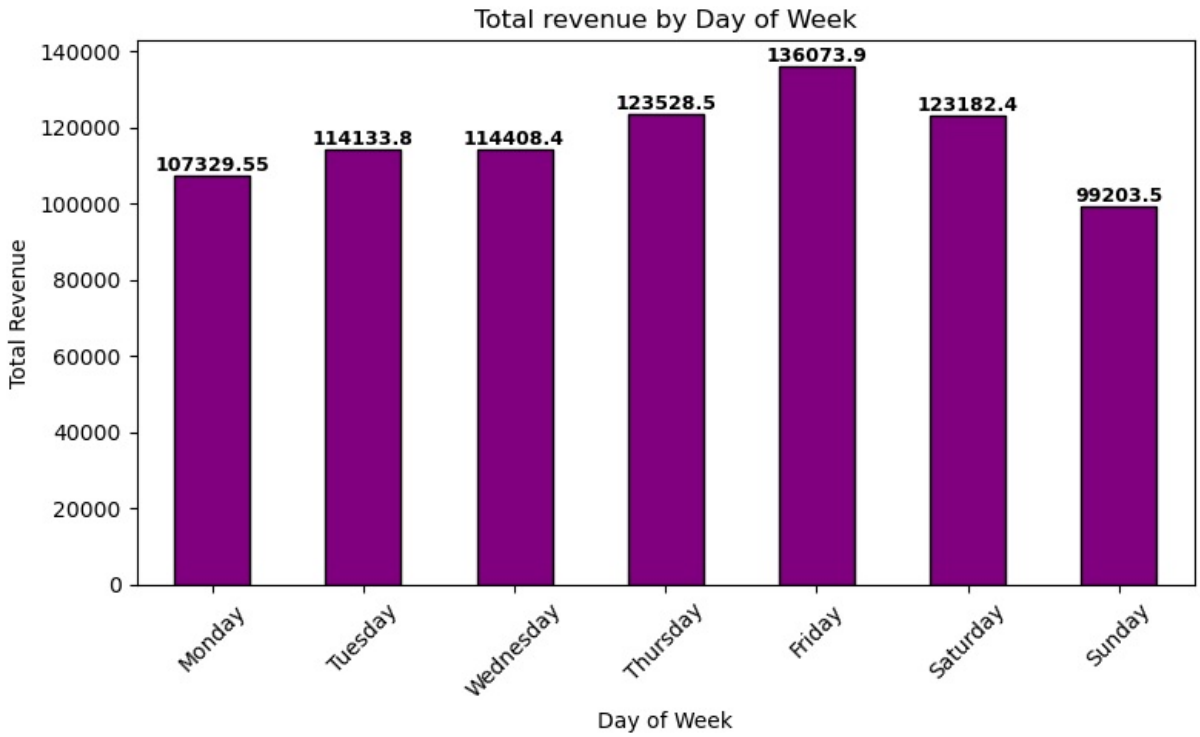
In [25]:
```
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)
df['order_day'] = df['order_date'].dt.day_name()

weekday_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
df['day_name'] = pd.Categorical(df['order_day'], categories=weekday_order, ordered=True)

orders_by_day = df.groupby('day_name', observed=False)['order_id'].nunique()

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)

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()
```



DAILY TREND - TOTAL REVENUE

In [29]:
```
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)
df['order_day'] = df['order_date'].dt.day_name()
```

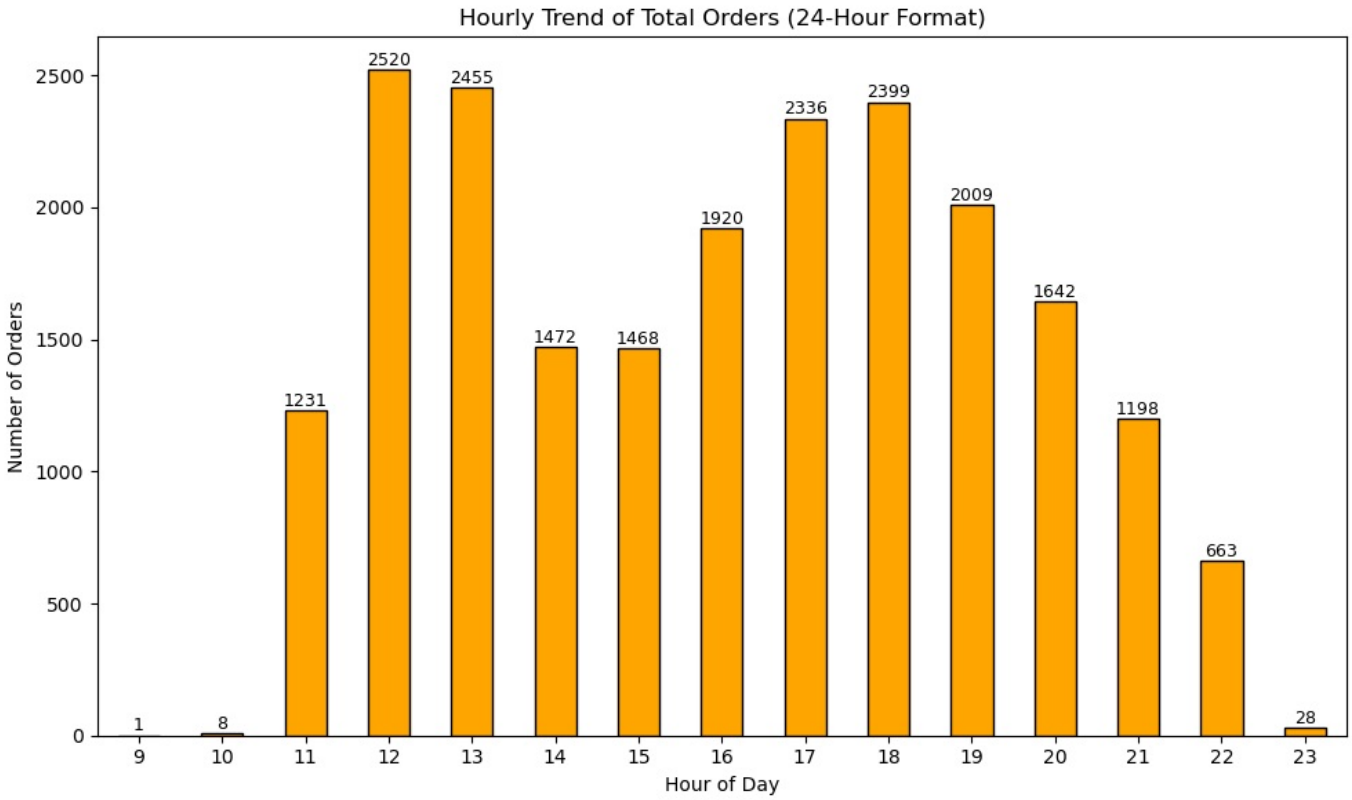
```
weekday_order = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]
df['day_name'] = pd.Categorical(df['order_day'], categories=weekday_order, ordered=True)

orders_by_day = df.groupby('day_name', observed=False)['total_price'].sum()

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)

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()
```



HOURLY TREND - TOTAL ORDERS

In [31]:

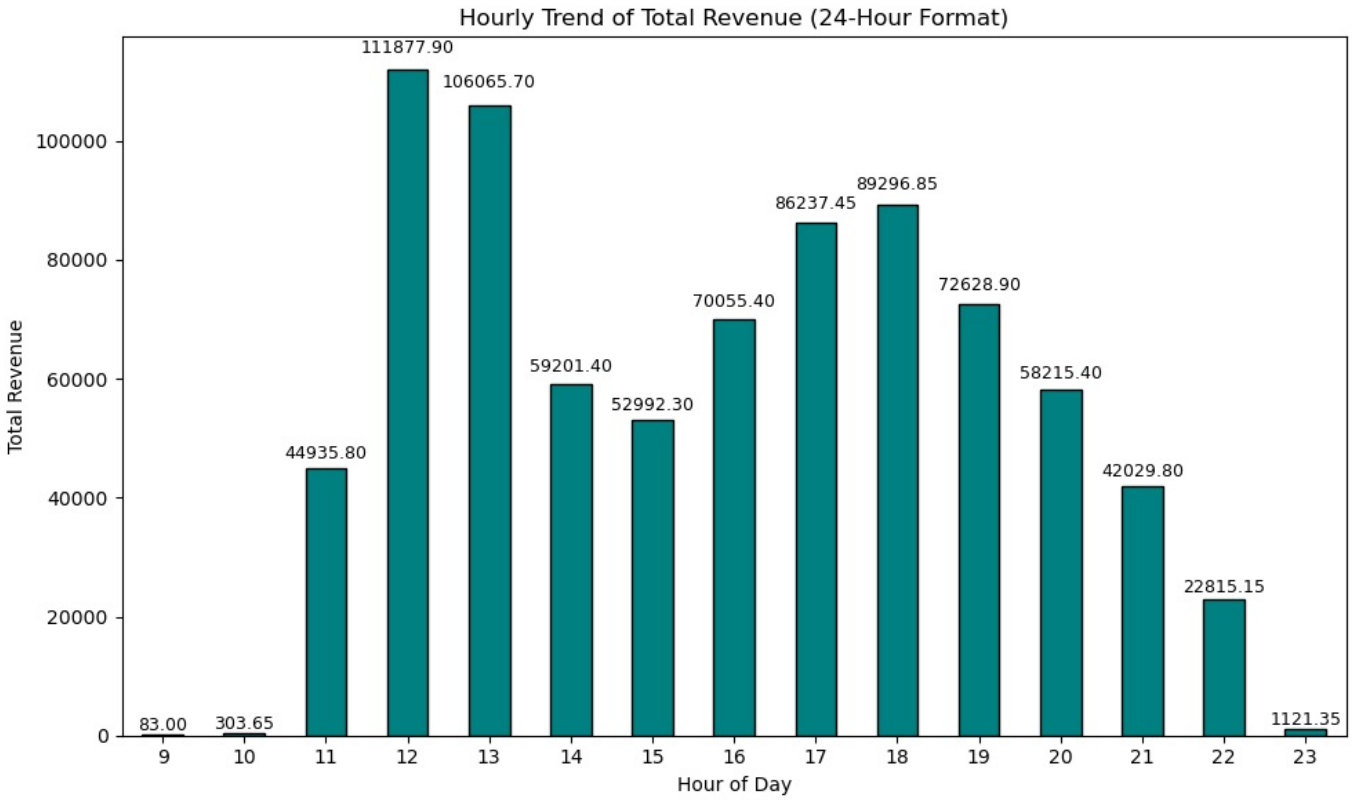
```
# Convert 'order_time' to datetime format
df['order_time'] = pd.to_datetime(df['order_time'], format='%H:%M:%S')

# Extract hour, minute, and second
df['order_hour'] = df['order_time'].dt.hour
df['order_minute'] = df['order_time'].dt.minute
df['order_second'] = df['order_time'].dt.second

# Group by hour and count unique orders
orders_by_hour = df.groupby('order_hour')['order_id'].nunique()

# Plotting
import matplotlib.pyplot as plt

ax = orders_by_hour.plot(kind='bar', figsize=(10,6), color='orange', edgecolor='black')
plt.title("Hourly Trend of Total Orders (24-Hour Format)")
plt.xlabel("Hour of Day")
plt.ylabel("Number of Orders")
plt.xticks(rotation=0)

# Annotate bars
for i, val in enumerate(orders_by_hour):
    plt.text(i, val + 1, str(val), ha='center', va='bottom', fontsize=9)

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



HOURLY TREND - TOTAL REVENUE

In [32]:

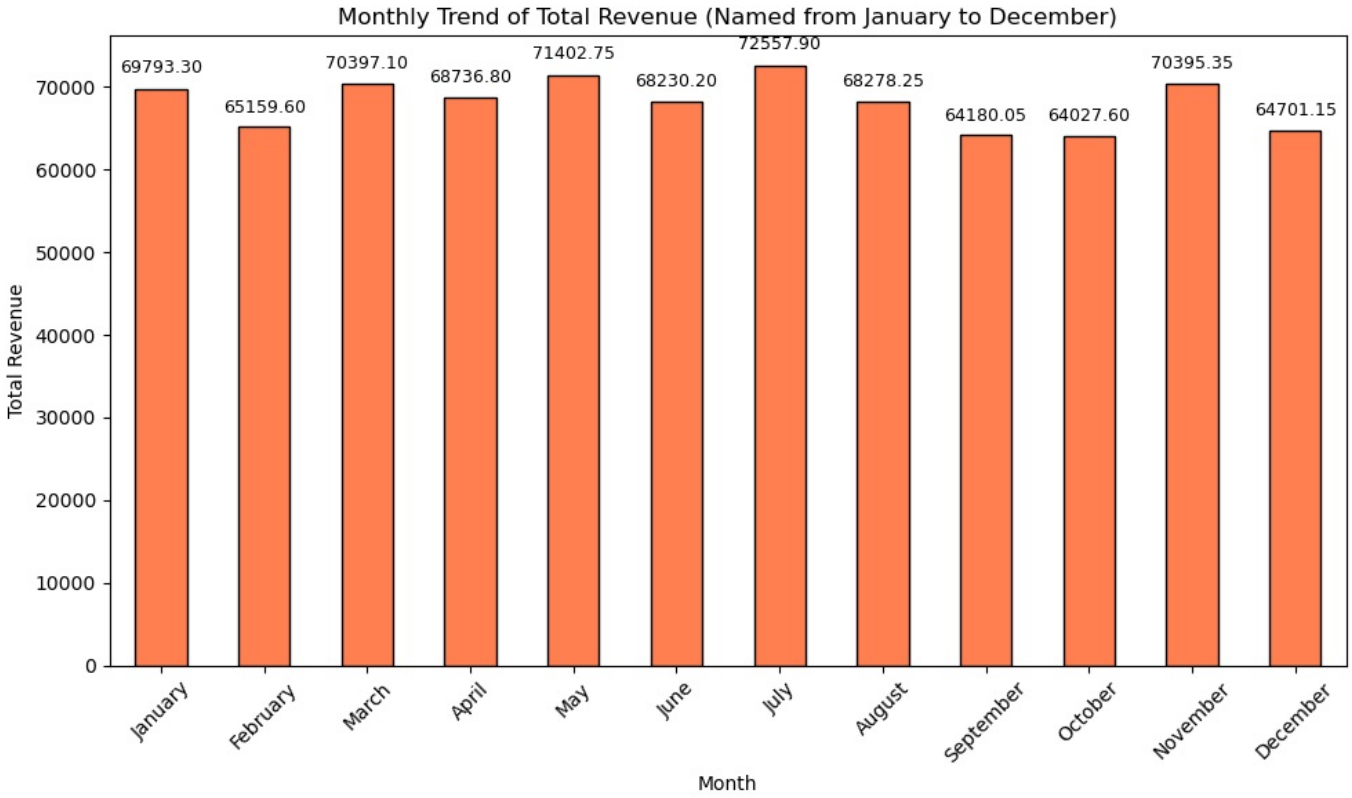
```
# Convert 'order_time' to datetime format
df['order_time'] = pd.to_datetime(df['order_time'], format='%H:%M:%S')

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

# Group by hour and sum total revenue
revenue_by_hour = df.groupby('order_hour')['total_price'].sum()

# Plotting
import matplotlib.pyplot as plt

ax = revenue_by_hour.plot(kind='bar', figsize=(10,6), color='teal', edgecolor='black')
plt.title("Hourly Trend of Total Revenue (24-Hour Format)")
plt.xlabel("Hour of Day")
plt.ylabel("Total Revenue")
plt.xticks(rotation=0)

# Annotate bars
for i, val in enumerate(revenue_by_hour):
    plt.text(i, val + val*0.02, f"{val:.2f}", ha='center', va='bottom', fontsize=9)

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



MONTHLY TREND - TOTAL REVENUE

In [35]:
```python
# Convert 'order_date' to datetime format
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)

# Extract full month name (e.g., "January")
df['order_month_name'] = df['order_date'].dt.strftime('%B')

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

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

# Group by month name and sum revenue
monthly_revenue_named = df.groupby('order_month_name')['total_price'].sum()

# Plotting
import matplotlib.pyplot as plt

ax = monthly_revenue_named.plot(kind='bar', figsize=(10,6), color='coral', edgecolor='black')
plt.title("Monthly Trend of Total Revenue (Named from January to December)")
plt.xlabel("Month")
plt.ylabel("Total Revenue")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(monthly_revenue_named):
    plt.text(i, val + val*0.02, f"{val:.2f}", ha='center', va='bottom', fontsize=9)

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

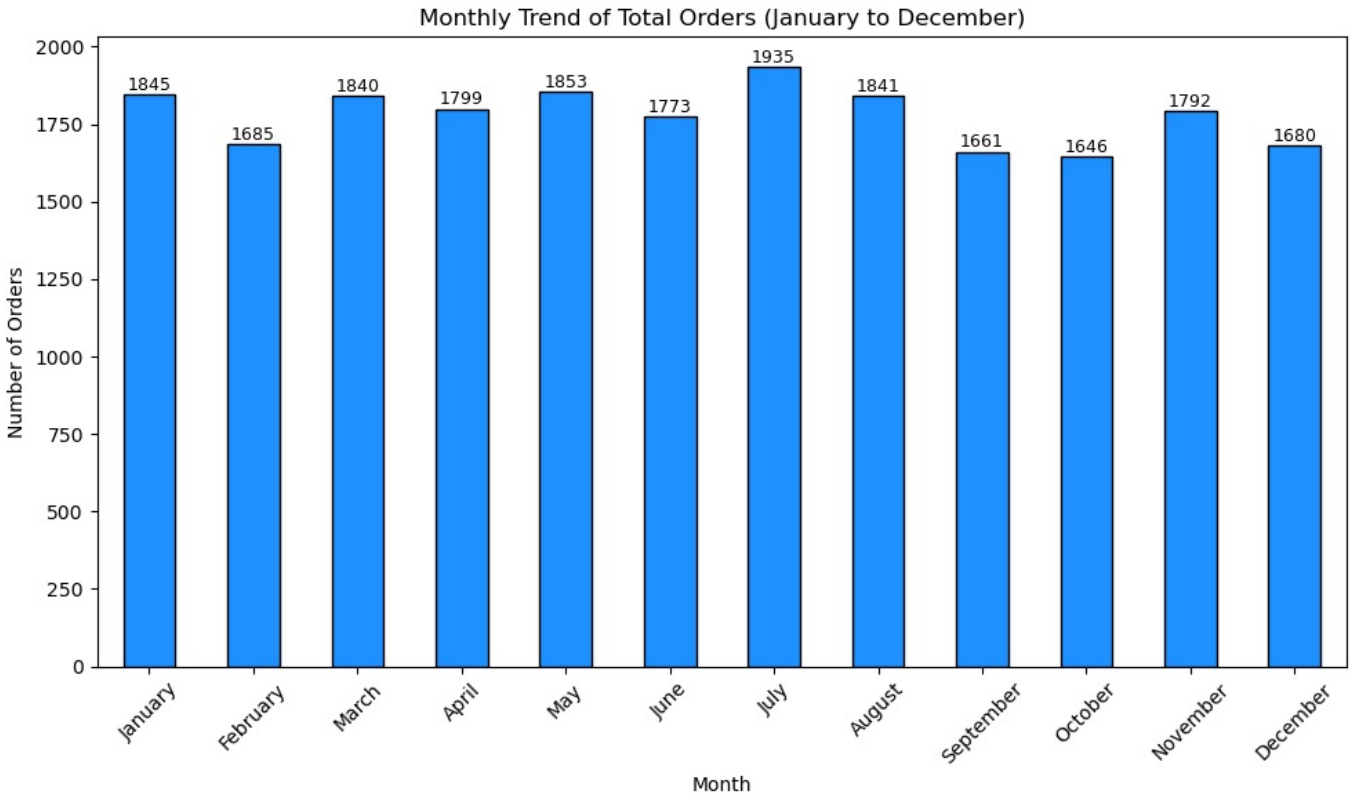
```
C:\Users\ashis\AppData\Local\Temp\ipykernel_17584\1847964869.py:15: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  monthly_revenue_named = df.groupby('order_month_name')['total_price'].sum()
```



MONTHLY TREND - TOTAL ORDERS

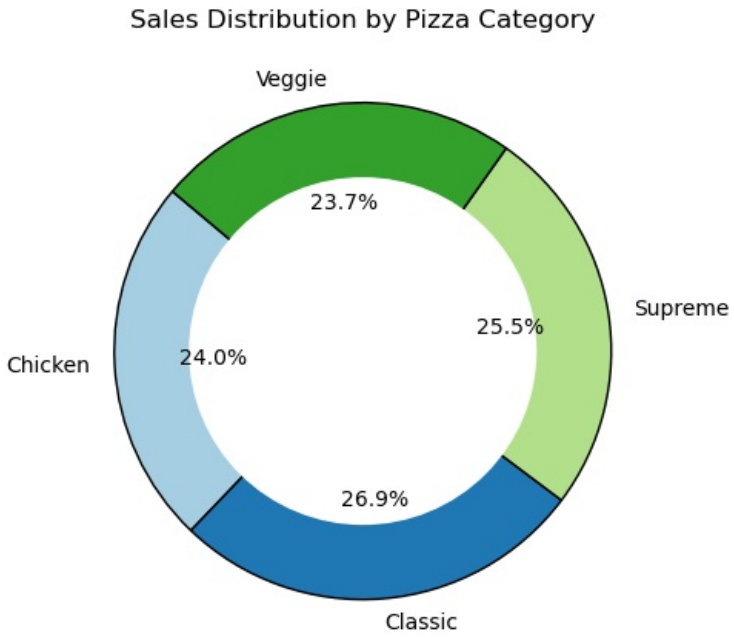
In [36]:
```python
# Convert 'order_date' to datetime format
df['order_date'] = pd.to_datetime(df['order_date'], dayfirst=True)

# Extract full month name (e.g., "January")
df['order_month_name'] = df['order_date'].dt.strftime('%B')

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

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

# Group by month name and count unique orders
monthly_orders_named = df.groupby('order_month_name')['order_id'].nunique()

# Plotting
import matplotlib.pyplot as plt

ax = monthly_orders_named.plot(kind='bar', figsize=(10,6), color='dodgerblue', edgecolor='black')
plt.title("Monthly Trend of Total Orders (January to December)")
plt.xlabel("Month")
plt.ylabel("Number of Orders")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(monthly_orders_named):
    plt.text(i, val + 1, str(val), ha='center', va='bottom', fontsize=9)

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

```
C:\Users\ashis\AppData\Local\Temp\ipykernel_17584\2250929251.py:15: FutureWarning: The default of observed=False is deprecated and will be changed to True in a future version of pandas. Pass observed=False to retain current behavior or observed=True to adopt the future default and silence this warning.
  monthly_orders_named = df.groupby('order_month_name')['order_id'].nunique()
```



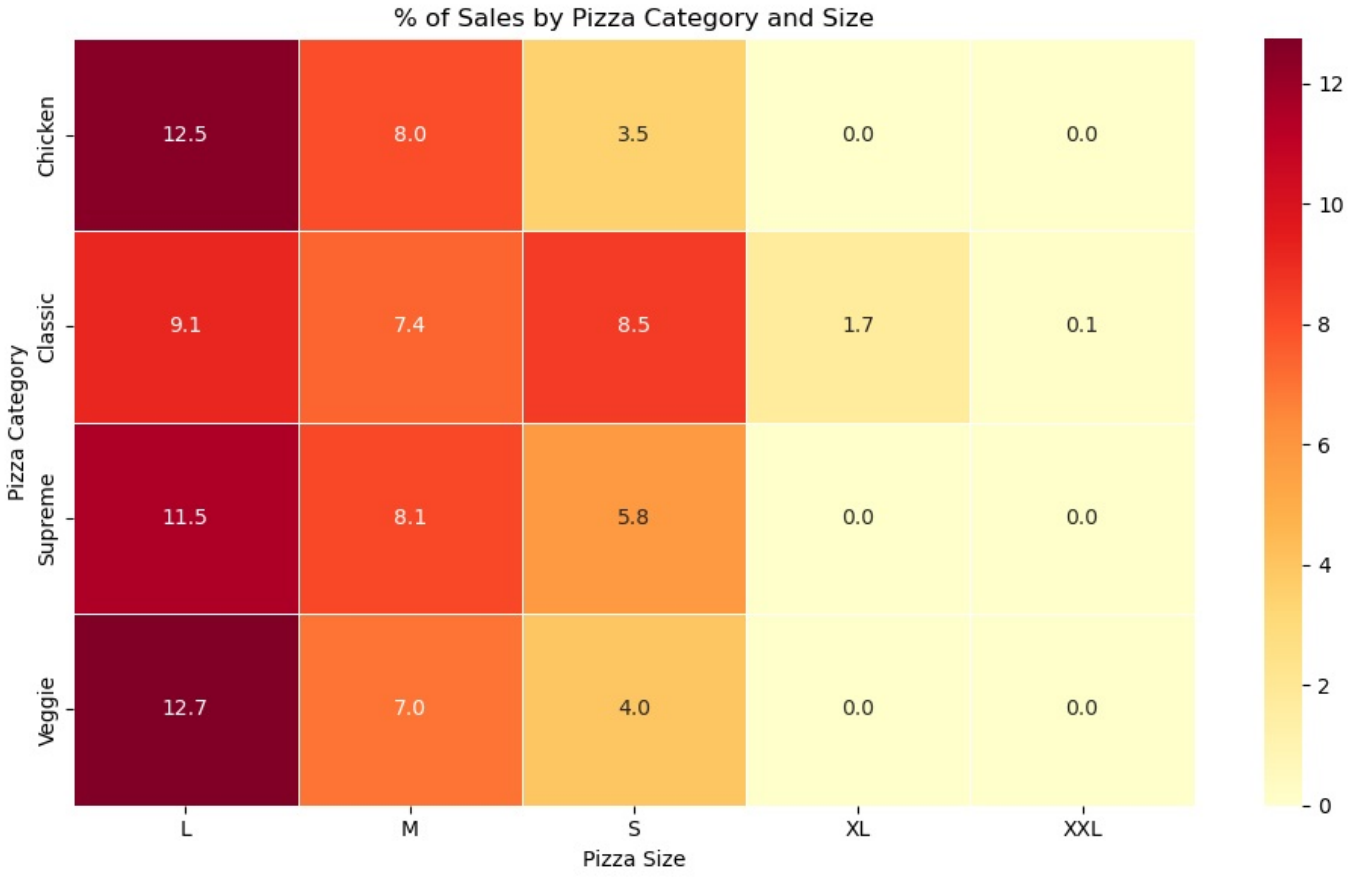
% of Sales by Category

In [39]:
```
# Group by pizza_category and sum total sales
sales_by_category = df.groupby('pizza_category')['total_price'].sum()

# Plotting
import matplotlib.pyplot as plt

plt.figure(figsize=(5,5))
colors = plt.cm.Paired(range(len(sales_by_category)))  # Optional: custom color palette

# Create donut chart by adding a white circle in the center
plt.pie(sales_by_category, labels=sales_by_category.index, autopct='%1.1f%%',
        startangle=140, colors=colors, wedgeprops={'edgecolor': 'black'})

# Add center circle to make it a donut
centre_circle = plt.Circle((0,0), 0.70, fc='white')
plt.gca().add_artist(centre_circle)

plt.title("Sales Distribution by Pizza Category")
plt.tight_layout()
plt.show()
```



% of Sales by Category & pizza size

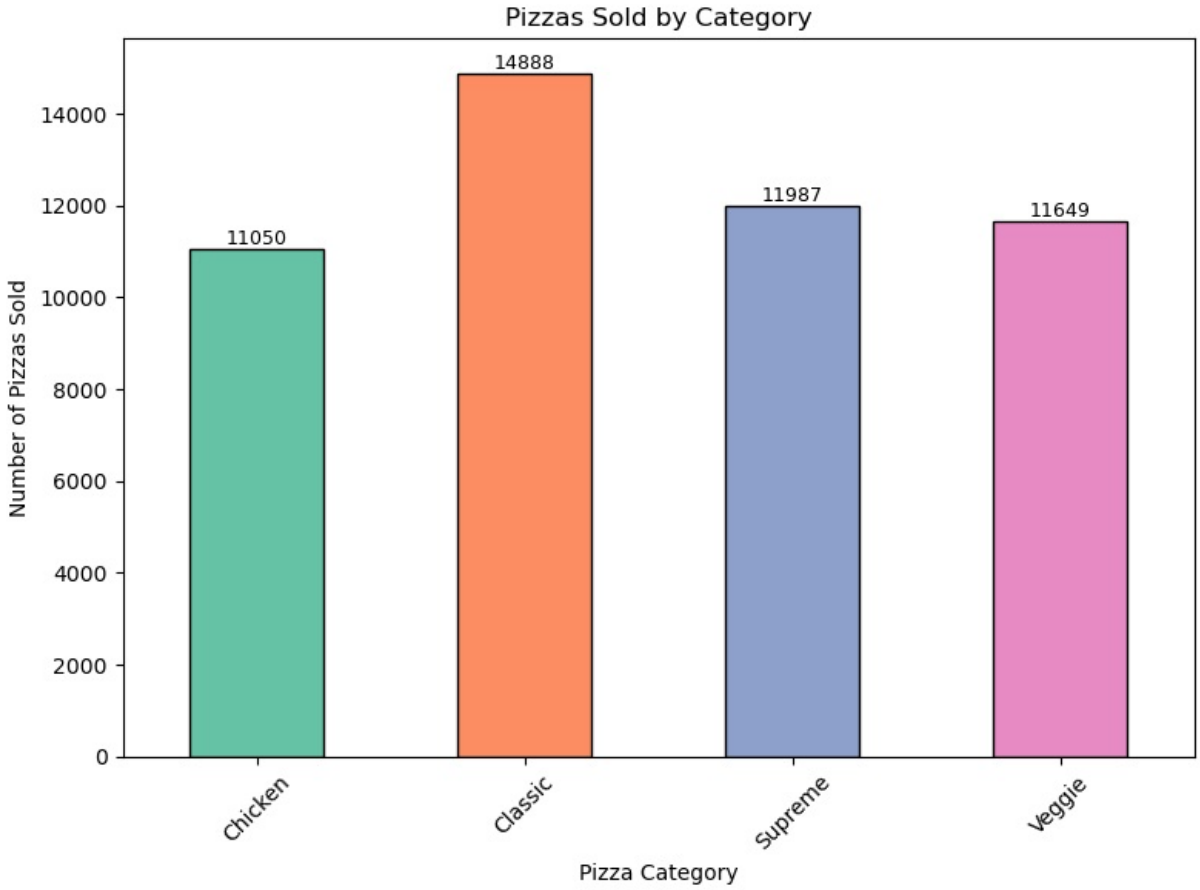
In [46]:
```
import seaborn as sns
import matplotlib.pyplot as plt

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

# Convert absolute sales to percentage of total
sales_pct = (sales_pivot / sales_pivot.values.sum()) * 100

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

plt.title("% of Sales by Pizza Category and Size")
plt.ylabel("Pizza Category")
plt.xlabel("Pizza Size")
plt.tight_layout()
plt.show()
```



total pizza sold by pizza category

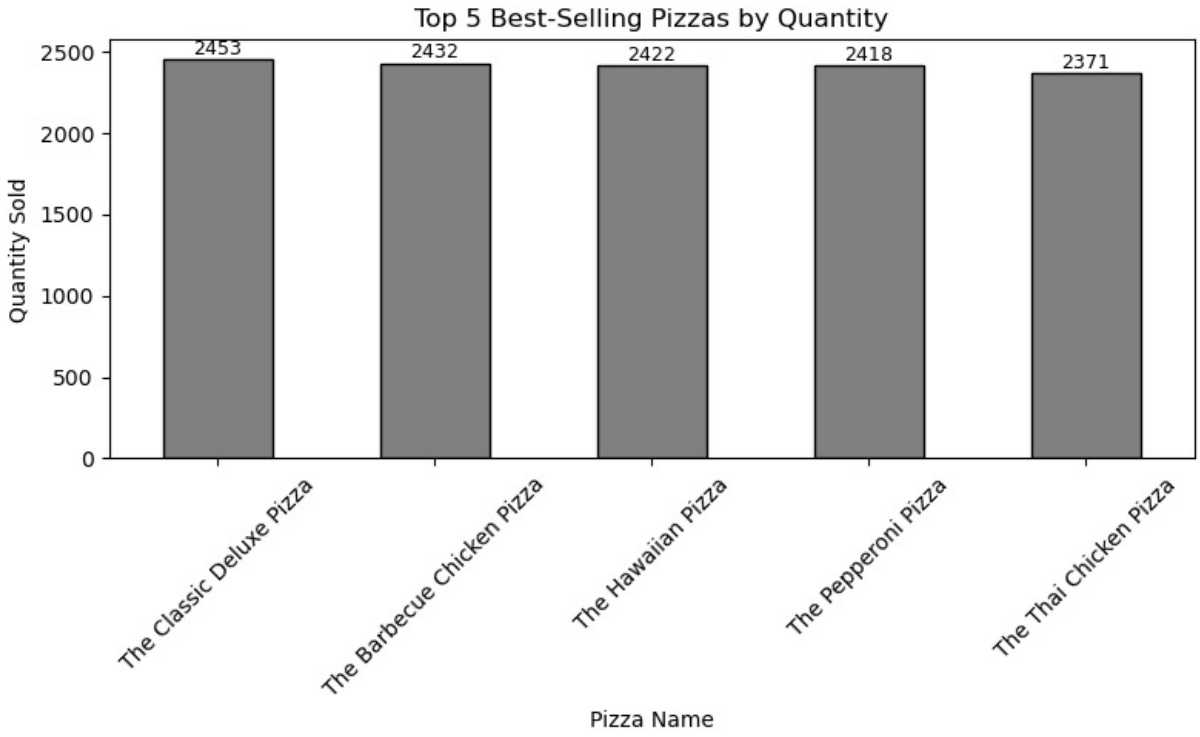
In [49]:
```
# Group by pizza_category and sum of number of pizzas sold
pizzas_sold = df.groupby('pizza_category')['quantity'].sum()

# Plotting
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
colors = plt.cm.Set2(range(len(pizzas_sold)))

pizzas_sold.plot(kind='bar', color=colors, edgecolor='black')

plt.title("Pizzas Sold by Category")
plt.xlabel("Pizza Category")
plt.ylabel("Number of Pizzas Sold")
plt.xticks(rotation=45)
```

```
# Annotate bars
for i, val in enumerate(pizzas_sold):
    plt.text(i, val + 1, str(val), ha='center', va='bottom', fontsize=9)

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



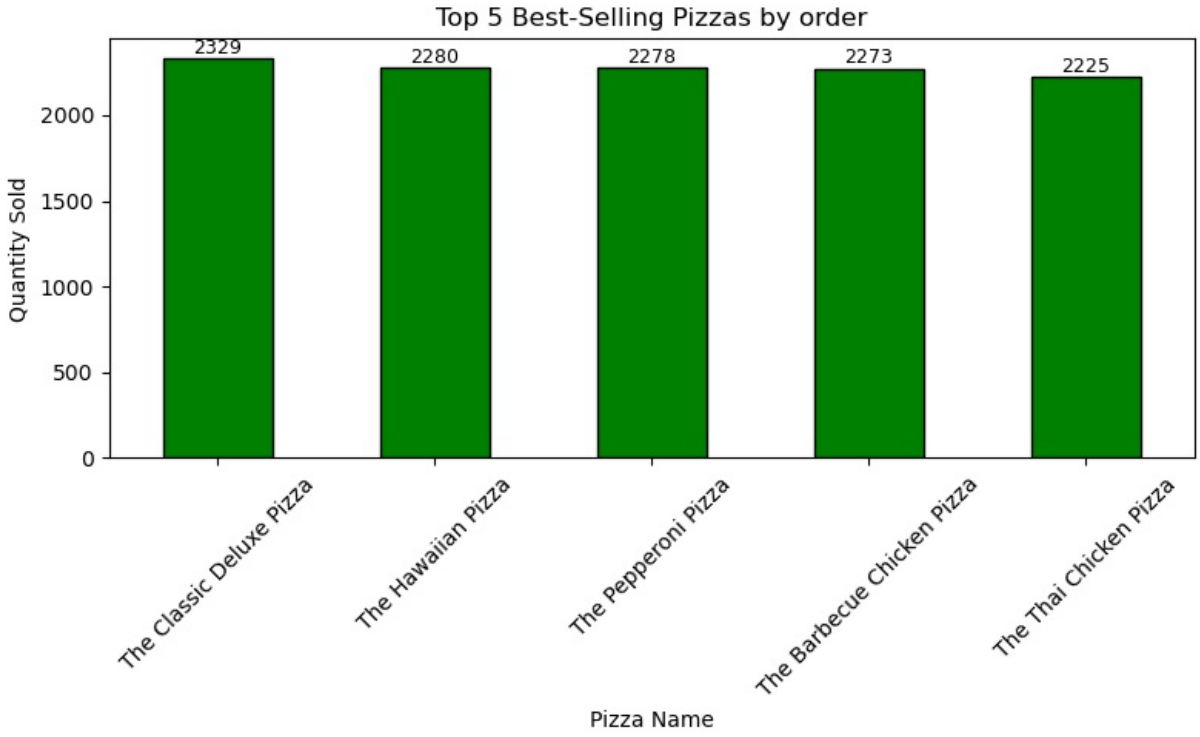
top 5 best selling pizzas - total qty

In [54]:
```
# Group and sort
pizzas_by_name = df.groupby('pizza_name')['quantity'].sum()
top5 = pizzas_by_name.sort_values(ascending=False).head(5)

# Plotting
import matplotlib.pyplot as plt

ax = top5.plot(kind='bar', figsize=(8,5), color='grey', edgecolor='black')
plt.title("Top 5 Best-Selling Pizzas by Quantity")
plt.xlabel("Pizza Name")
plt.ylabel("Quantity Sold")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(top5):
    plt.text(i, val + 0.5, str(val), ha='center', va='bottom', fontsize=9)

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



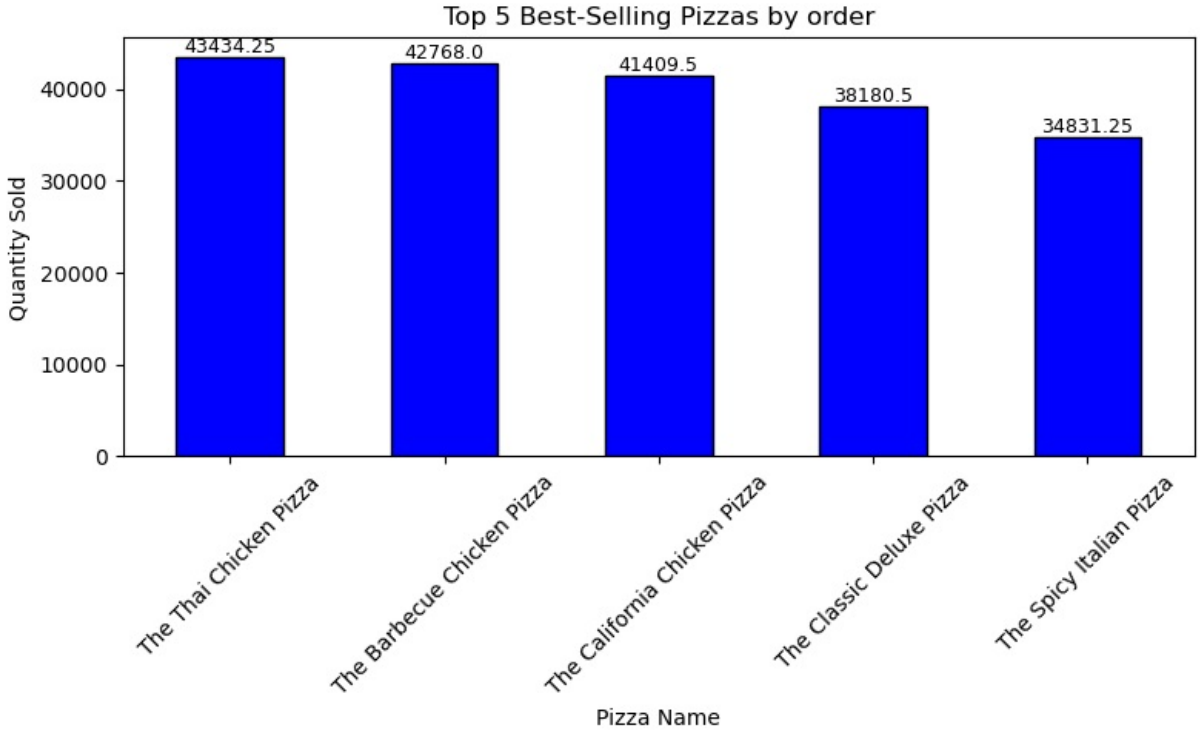
top 5 best selling pizzas - by order

In [58]:
```
# Group and sort
pizzas_by_name = df.groupby('pizza_name')['order_id'].nunique()
top5 = pizzas_by_name.sort_values(ascending=False).head(5)

# Plotting
import matplotlib.pyplot as plt

ax = top5.plot(kind='bar', figsize=(8,5), color='green', edgecolor='black')
plt.title("Top 5 Best-Selling Pizzas by order")
plt.xlabel("Pizza Name")
plt.ylabel("Quantity Sold")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(top5):
    plt.text(i, val + 0.5, str(val), ha='center', va='bottom', fontsize=9)

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



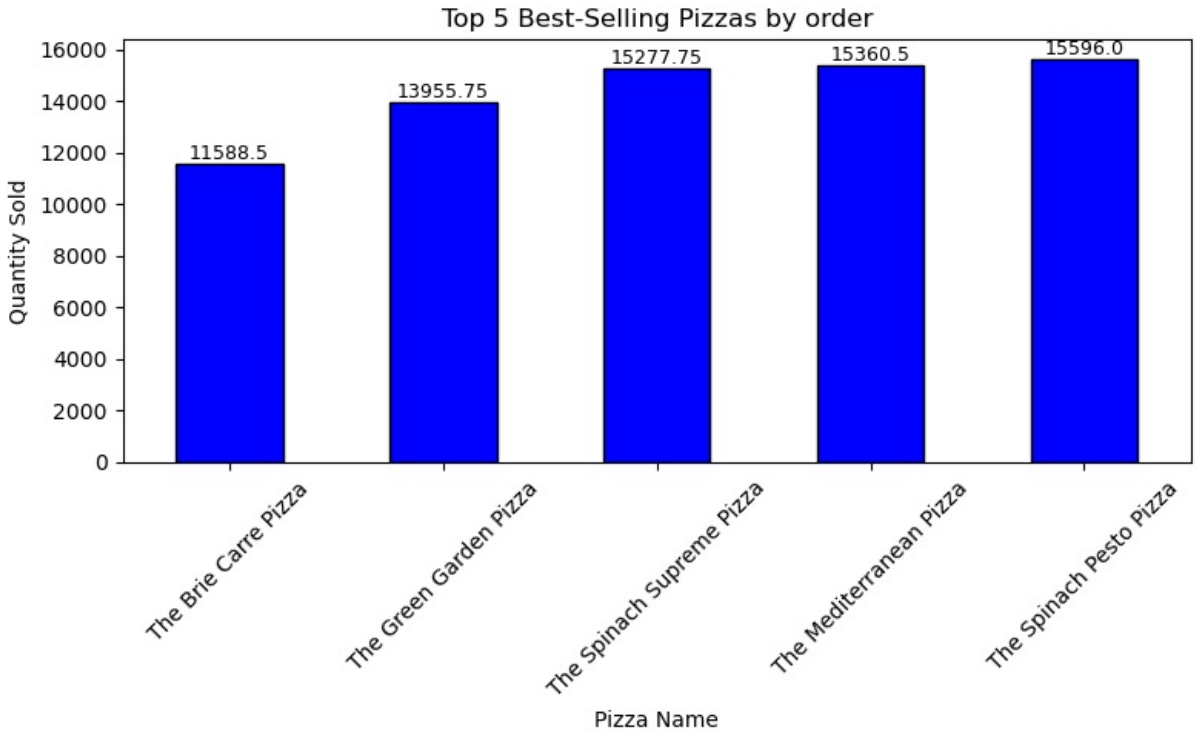
top 5 best selling pizzas - by revenue

In [60]:
```
# Group and sort
pizzas_by_name = df.groupby('pizza_name')['total_price'].sum()
top5 = pizzas_by_name.sort_values(ascending=False).head(5)

# Plotting
import matplotlib.pyplot as plt

ax = top5.plot(kind='bar', figsize=(8,5), color='blue', edgecolor='black')
plt.title("Top 5 Best-Selling Pizzas by order")
plt.xlabel("Pizza Name")
plt.ylabel("Quantity Sold")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(top5):
    plt.text(i, val + 0.5, str(val), ha='center', va='bottom', fontsize=9)

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



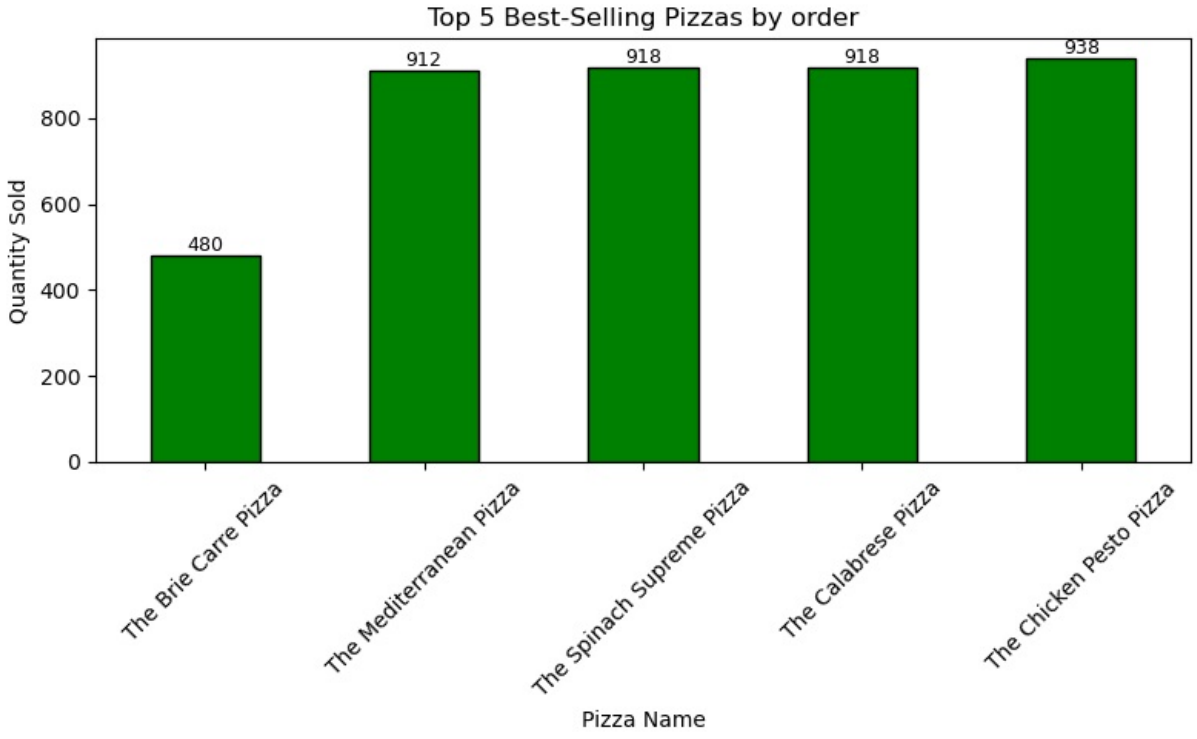
bottom 5 best selling pizzas - by revenue

In [61]:
```
# Group and sort
pizzas_by_name = df.groupby('pizza_name')['total_price'].sum()
top5 = pizzas_by_name.sort_values(ascending=True).head(5)

# Plotting
import matplotlib.pyplot as plt

ax = top5.plot(kind='bar', figsize=(8,5), color='blue', edgecolor='black')
plt.title("Top 5 Best-Selling Pizzas by order")
plt.xlabel("Pizza Name")
plt.ylabel("Quantity Sold")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(top5):
    plt.text(i, val + 0.5, str(val), ha='center', va='bottom', fontsize=9)

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



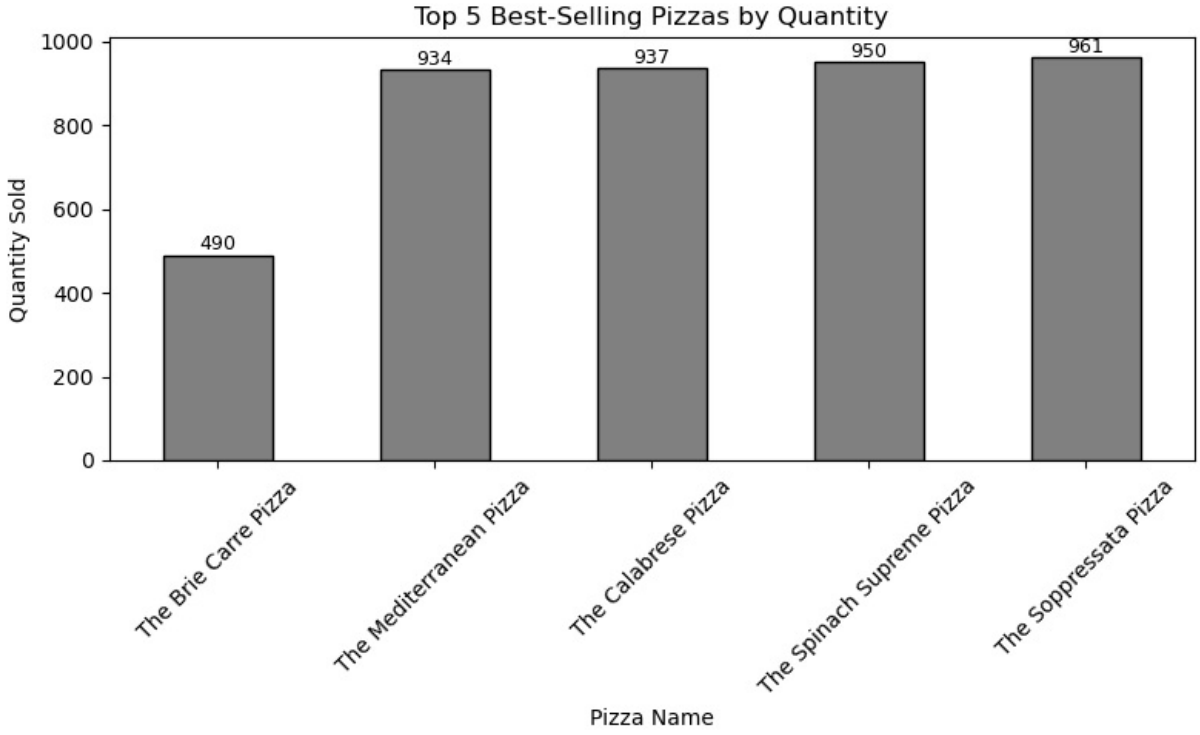
bottom 5 best selling pizzas - by order

In [62]:
```
# Group and sort
pizzas_by_name = df.groupby('pizza_name')['order_id'].nunique()
top5 = pizzas_by_name.sort_values(ascending=True).head(5)

# Plotting
import matplotlib.pyplot as plt

ax = top5.plot(kind='bar', figsize=(8,5), color='green', edgecolor='black')
plt.title("Top 5 Best-Selling Pizzas by order")
plt.xlabel("Pizza Name")
plt.ylabel("Quantity Sold")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(top5):
    plt.text(i, val + 0.5, str(val), ha='center', va='bottom', fontsize=9)

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



bottom 5 best selling pizzas - total qty

In [64]:
```
# Group and sort
pizzas_by_name = df.groupby('pizza_name')['quantity'].sum()
top5 = pizzas_by_name.sort_values(ascending=True).head(5)

# Plotting
import matplotlib.pyplot as plt

ax = top5.plot(kind='bar', figsize=(8,5), color='grey', edgecolor='black')
plt.title("Top 5 Best-Selling Pizzas by Quantity")
plt.xlabel("Pizza Name")
plt.ylabel("Quantity Sold")
plt.xticks(rotation=45)

# Annotate bars
for i, val in enumerate(top5):
    plt.text(i, val + 0.5, str(val), ha='center', va='bottom', fontsize=9)

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



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: