In [2]:
```python
import pandas as pd
import numpy as np
```

In [3]:
```python
# Read data
df = pd.read_excel("Online Retail.xlsx")
```

In [4]:
```python
# Save data (to CSV for example)
df.to_csv("Online_Retail.csv", index=False)
```

In [9]:
```python
# Top 5 rows
df.head()
```

Out[9]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [10]:
```python
# last 5 rows
df.tail()
```

Out[10]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France
541905	581587	22899	CHILDREN'S APRON DOLLY GIRL	6	2011-12-09 12:50:00	2.10	12680.0	France
541906	581587	23254	CHILDRENS CUTLERY DOLLY GIRL	4	2011-12-09 12:50:00	4.15	12680.0	France
541907	581587	23255	CHILDRENS CUTLERY CIRCUS PARADE	4	2011-12-09 12:50:00	4.15	12680.0	France
541908	581587	22138	BAKING SET 9 PIECE RETROSPOT	3	2011-12-09 12:50:00	4.95	12680.0	France

In [12]:
```python
# Select one column (eg:'CustomerID')
df['CustomerID']
```

Out[12]:
```
0         17850.0
1         17850.0
2         17850.0
3         17850.0
4         17850.0
           ...
541904    12680.0
541905    12680.0
541906    12680.0
541907    12680.0
541908    12680.0
Name: CustomerID, Length: 541909, dtype: float64
```

In [74]:
```python
# Drop missing CustomerIDs
df = df.dropna(subset=['CustomerID'])
```

In [76]:
```python
# Check dimension
df.shape
```

Out[76]:
```
(406829, 11)
```

In [77]:
```python
# first 1000 rows
df.iloc[:1000]
```

Out[77]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	Profit	HighValue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30	13.770	1
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	18.306	1
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00	19.800	1
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	18.306	1
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34	18.306	1
...	...	...	...	...	...	...	...	...	...	...	...
996	536520	22100	SKULLS SQUARE TISSUE BOX	1	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	1.25	1.125	0
997	536520	22096	PINK PAISLEY SQUARE TISSUE BOX	1	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	1.25	1.125	0
998	536520	22583	PACK OF 6 HANDBAG GIFT BOXES	1	2010-12-01 12:43:00	2.55	14729.0	United Kingdom	2.55	2.295	0
999	536520	21358	TOAST ITS - HAPPY BIRTHDAY	2	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	2.50	2.250	0
1000	536520	21123	SET/10 IVORY POLKADOT PARTY CANDLES	1	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	1.25	1.125	0

1000 rows × 11 columns

In [78]:
```python
# Get rows from 1001-2000
df.iloc[1000:2000]
```

Out[78]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice	Profit	HighValue
1001	536520	21124	SET/10 BLUE POLKADOT PARTY CANDLES	1	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	1.25	1.125	0
1002	536520	21122	SET/10 PINK POLKADOT PARTY CANDLES	1	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	1.25	1.125	0
1003	536520	84378	SET OF 3 HEART COOKIE CUTTERS	1	2010-12-01 12:43:00	1.25	14729.0	United Kingdom	1.25	1.125	0
1004	536520	21985	PACK OF 12 HEARTS DESIGN TISSUES	12	2010-12-01 12:43:00	0.29	14729.0	United Kingdom	3.48	3.132	0
1005	536520	21427	SKULLS STORAGE BOX SMALL	2	2010-12-01 12:43:00	2.10	14729.0	United Kingdom	4.20	3.780	0
...	...	...	...	...	...	...	...	...	...	...	...
3135	536602	21068	VINTAGE BILLBOARD LOVE/HATE MUG	6	2010-12-02 08:34:00	1.06	17850.0	United Kingdom	6.36	5.724	0
3136	536602	82483	WOOD 2 DRAWER CABINET WHITE FINISH	4	2010-12-02 08:34:00	4.95	17850.0	United Kingdom	19.80	17.820	1
3137	536602	22411	JUMBO SHOPPER VINTAGE RED PAISLEY	6	2010-12-02 08:34:00	1.65	17850.0	United Kingdom	9.90	8.910	1
3138	536602	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-02 08:34:00	3.39	17850.0	United Kingdom	20.34	18.306	1
3139	536602	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-02 08:34:00	3.39	17850.0	United Kingdom	20.34	18.306	1

1000 rows × 11 columns

In [79]:
```python
# Check datatypes
df.dtypes
```

Out[79]:
```
InvoiceNo              object
StockCode              object
Description            object
Quantity                int64
InvoiceDate    datetime64[ns]
UnitPrice             float64
CustomerID            float64
Country                object
TotalPrice            float64
Profit                float64
HighValue               int32
dtype: object
```

In [19]:
```python
# Frequency table of a column (Country)
df['Country'].value_counts()
```

Out[19]:
```
United Kingdom          495478
Germany                   9495
France                    8557
EIRE                      8196
Spain                     2533
Netherlands               2371
Belgium                   2069
Switzerland               2002
Portugal                  1519
Australia                 1259
Norway                    1086
Italy                      803
Channel Islands            758
Finland                    695
Cyprus                     622
Sweden                     462
Unspecified                446
Austria                    401
Denmark                    389
Japan                      358
Poland                     341
Israel                     297
USA                        291
Hong Kong                  288
Singapore                  229
Iceland                    182
Canada                     151
Greece                     146
Malta                      127
United Arab Emirates        68
European Community          61
RSA                         58
Lebanon                     45
Lithuania                   35
Brazil                      32
Czech Republic              30
Bahrain                     19
Saudi Arabia                10
Name: Country, dtype: int64
```

In [20]:
```python
# Unique values and count
df['Country'].unique(), df['Country'].nunique()
```

Out[20]:
```
(array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
        'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
        'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
        'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
        'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore',
        'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
        'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
        'European Community', 'Malta', 'RSA'], dtype=object),
 38)
```

In [21]:
```python
# Data summary
df.describe()
```

Out[21]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

In [24]:
```python
# loc, iloc
df.loc[0:5, ['InvoiceNo','CustomerID']] ,df.iloc[0:5, 0:2]
```

Out[24]:
```
(  InvoiceNo  CustomerID
 0    536365     17850.0
 1    536365     17850.0
 2    536365     17850.0
 3    536365     17850.0
 4    536365     17850.0
 5    536365     17850.0,
   InvoiceNo StockCode
 0    536365    85123A
 1    536365     71053
 2    536365    84406B
 3    536365    84029G
 4    536365    84029E)
```

In [26]:
```python
# Subset with condition (eg: Quantity > 10)
df[df['Quantity'] > 10]
```

Out[26]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
9	536367	84879	ASSORTED COLOUR BIRD ORNAMENT	32	2010-12-01 08:34:00	1.69	13047.0	United Kingdom
26	536370	22728	ALARM CLOCK BAKELIKE PINK	24	2010-12-01 08:45:00	3.75	12583.0	France
27	536370	22727	ALARM CLOCK BAKELIKE RED	24	2010-12-01 08:45:00	3.75	12583.0	France
28	536370	22726	ALARM CLOCK BAKELIKE GREEN	12	2010-12-01 08:45:00	3.75	12583.0	France
29	536370	21724	PANDA AND BUNNIES STICKER SHEET	12	2010-12-01 08:45:00	0.85	12583.0	France
...	...	...	...	...	...	...	...	...
541894	581587	22631	CIRCUS PARADE LUNCH BOX	12	2011-12-09 12:50:00	1.95	12680.0	France
541895	581587	22556	PLASTERS IN TIN CIRCUS PARADE	12	2011-12-09 12:50:00	1.65	12680.0	France
541896	581587	22555	PLASTERS IN TIN STRONGMAN	12	2011-12-09 12:50:00	1.65	12680.0	France
541902	581587	22629	SPACEBOY LUNCH BOX	12	2011-12-09 12:50:00	1.95	12680.0	France
541904	581587	22613	PACK OF 20 SPACEBOY NAPKINS	12	2011-12-09 12:50:00	0.85	12680.0	France

132631 rows × 8 columns

In [80]:
```python
# Remove negative or zero values
df = df[df['Quantity'] > 0]
df = df[df['UnitPrice'] > 0]
```

In [82]:
```python
# Remove cancelled invoices (InvoiceNo starting with 'C')
df = df[~df['InvoiceNo'].astype(str).str.startswith('C')]
```

In [81]:
```python
# Check duplicates
df.duplicated().sum()
```

Out[81]:
```
5192
```

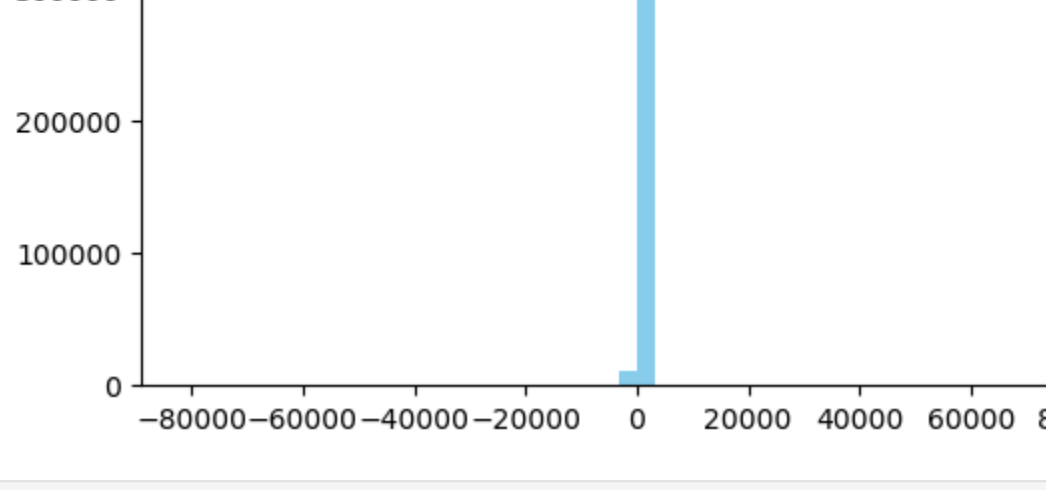
In [83]:
```python
# Remove duplicates
df = df.drop_duplicates()
```

In [84]:
```python
# Check missing values
df.isnull().sum()
```

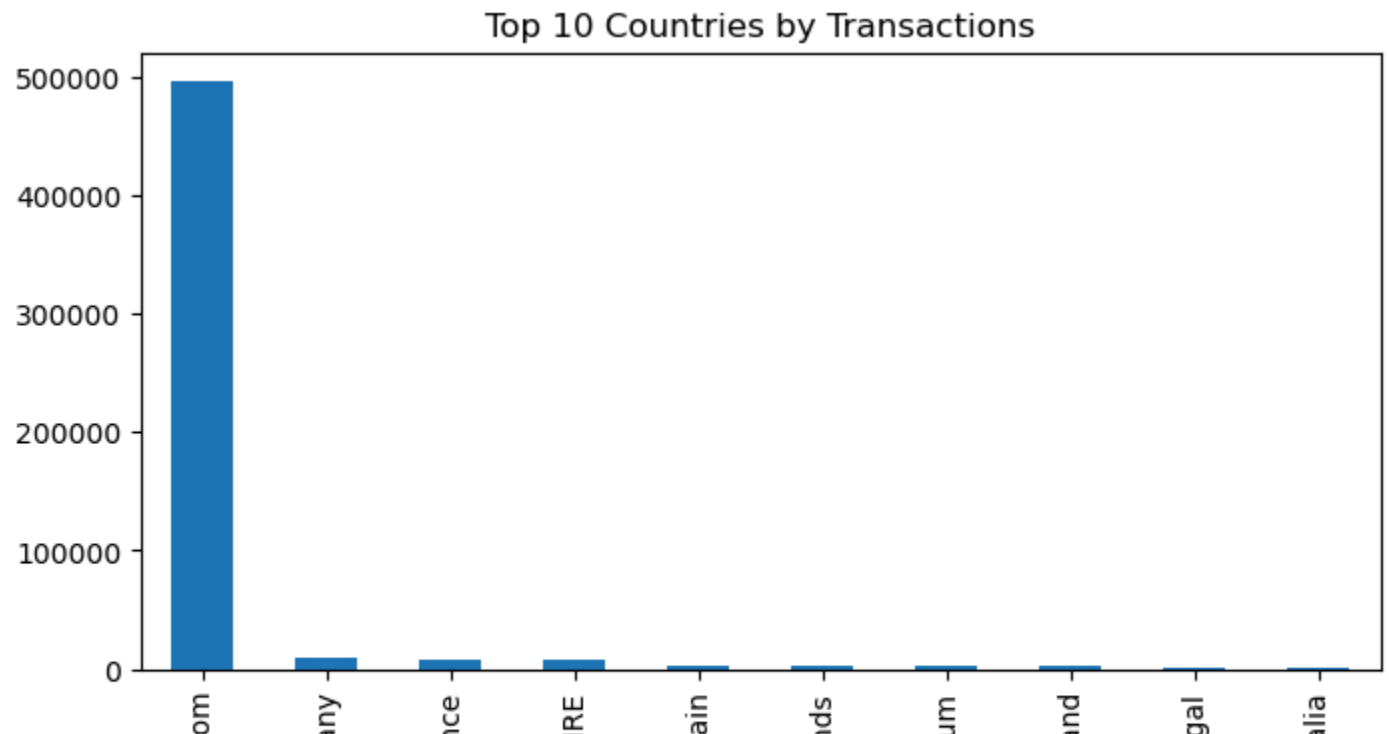
Out[84]:
```
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
TotalPrice     0
Profit         0
HighValue      0
dtype: int64
```

In [35]:
```python
# check different plots
import matplotlib.pyplot as plt
import seaborn as sns

# Histogram for Quantity
plt.hist(df['Quantity'], bins=50, color='skyblue')
plt.title("Distribution of Quantity")
plt.show()
```

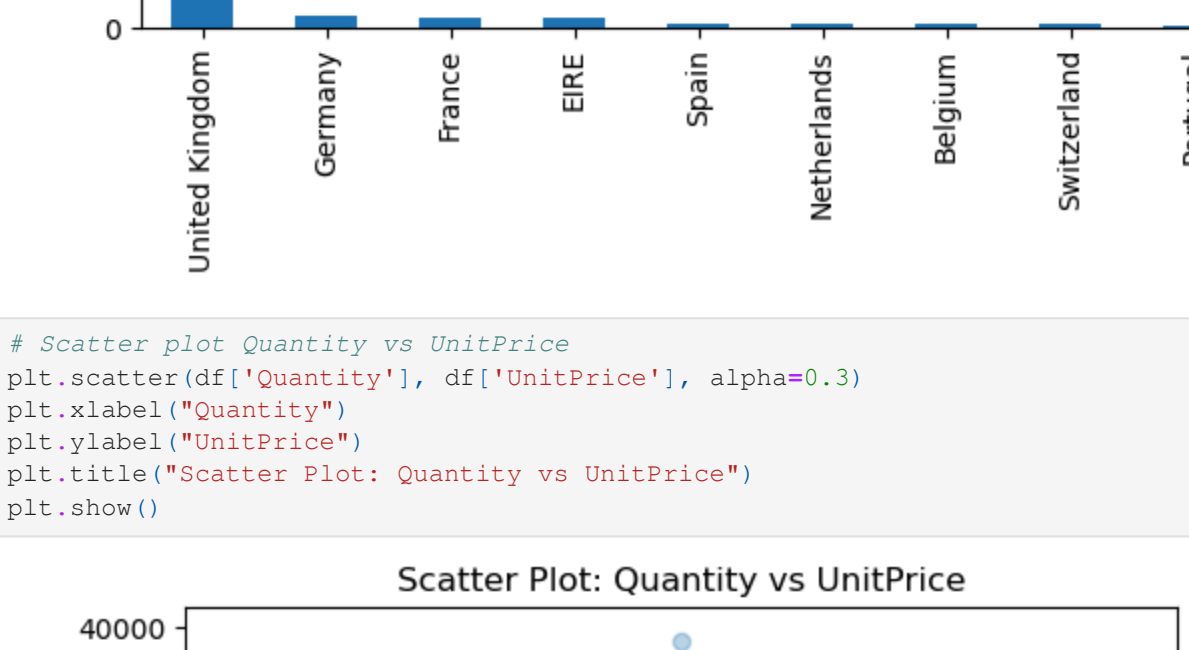
In [36]:
```python
# Boxplot for UnitPrice
sns.boxplot(x=df['UnitPrice'])
plt.title("Boxplot of UnitPrice")
plt.show()
```



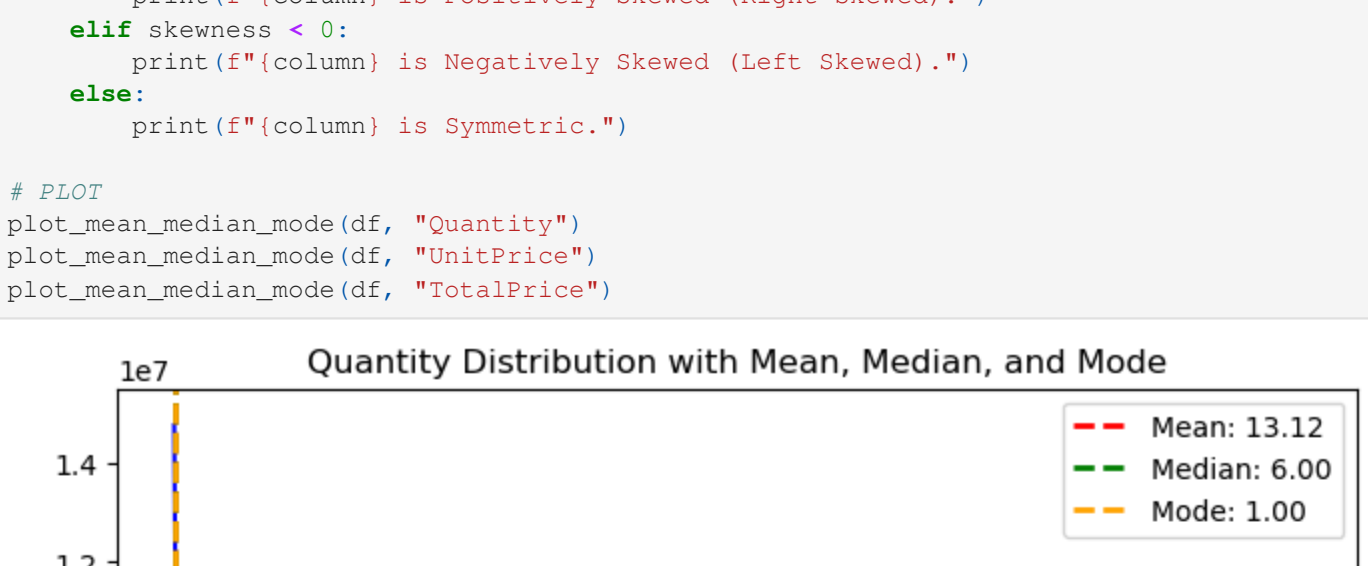
In [37]:
```python
# Bar chart for top 10 countries
df['Country'].value_counts().head(10).plot(kind='bar', figsize=(8,4))
plt.title("Top 10 Countries by Transactions")
plt.show()
```



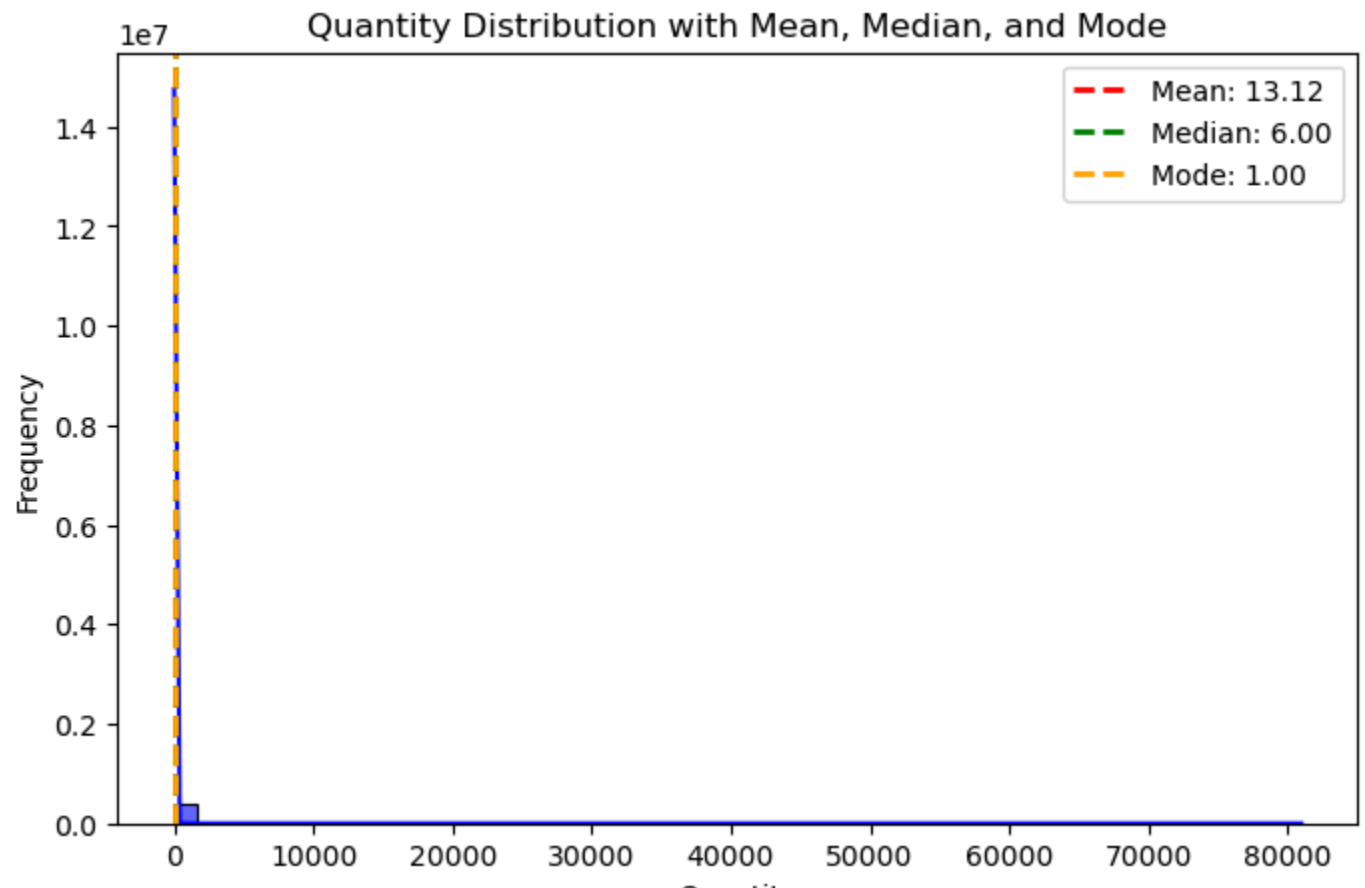
In [38]:
```python
# Scatter plot Quantity vs UnitPrice
plt.scatter(df['Quantity'], df['UnitPrice'], alpha=0.3)
plt.xlabel("Quantity")
plt.ylabel("UnitPrice")
plt.title("Scatter Plot: Quantity vs UnitPrice")
plt.show()
```



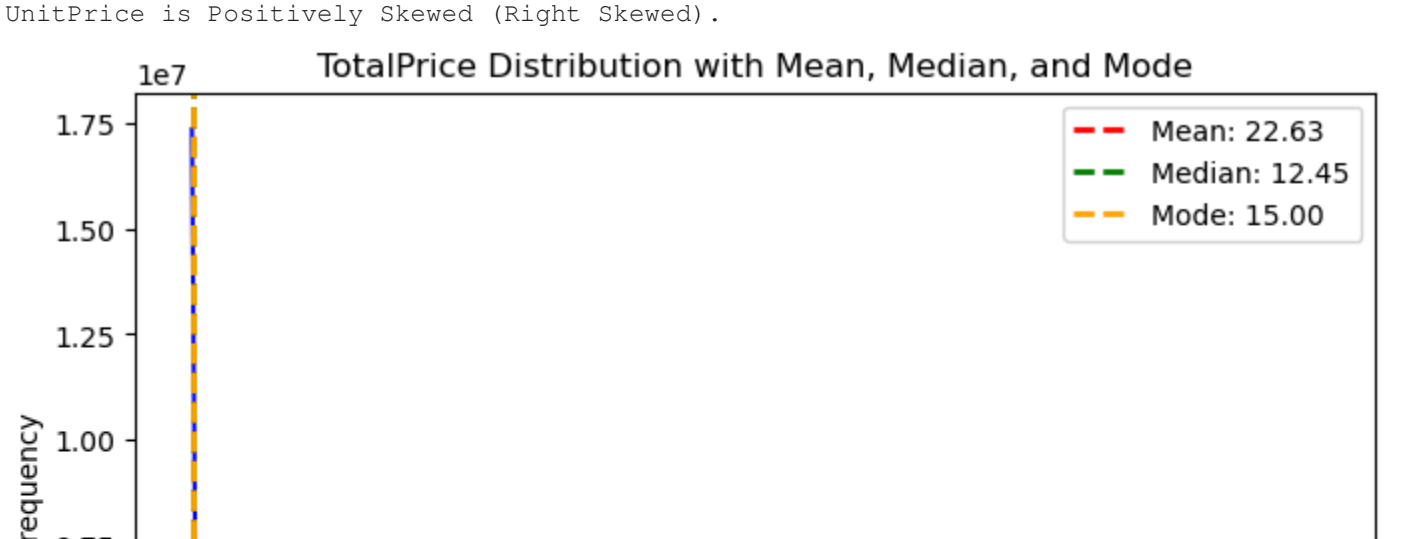
In [88]:
```python
# EDA:

import matplotlib.pyplot as plt
import seaborn as sns
import statistics as stat
from scipy.stats import skew

def plot_mean_median_mode(df, column):
    data = df[column].dropna()

    mean = data.mean()
    median = data.median()
    mode = stat.mode(data)
    skewness = skew(data)

    plt.figure(figsize=(8,5))
    sns.histplot(data, bins=50, kde=True, color='blue', alpha=0.6)

    # mean, median, mode
    plt.axvline(mean, color='red', linestyle='dashed', linewidth=2, label=f'Mean: {mean:.2f}')
    plt.axvline(median, color='green', linestyle='dashed', linewidth=2, label=f'Median: {median:.2f}')
    plt.axvline(mode, color='orange', linestyle='dashed', linewidth=2, label=f'Mode: {mode:.2f}')

    plt.title(f"{column} Distribution with Mean, Median, and Mode")
    plt.xlabel(column)
    plt.ylabel("Frequency")
    plt.legend()
    plt.show()

    # skewness
    print(f"\nColumn: {column}")
    print(f"Skewness = {skewness:.2f}")
    if skewness > 0:
        print(f"{column} is Positively Skewed (Right Skewed).")
    elif skewness < 0:
        print(f"{column} is Negatively Skewed (Left Skewed).")
    else:
        print(f"{column} is Symmetric.")

# PLOT
plot_mean_median_mode(df, "Quantity")
plot_mean_median_mode(df, "UnitPrice")
plot_mean_median_mode(df, "TotalPrice")
```




```
Column: Quantity
Skewness = 407.34
Quantity is Positively Skewed (Right Skewed).
```




```
Column: UnitPrice
Skewness = 202.74
UnitPrice is Positively Skewed (Right Skewed).
```

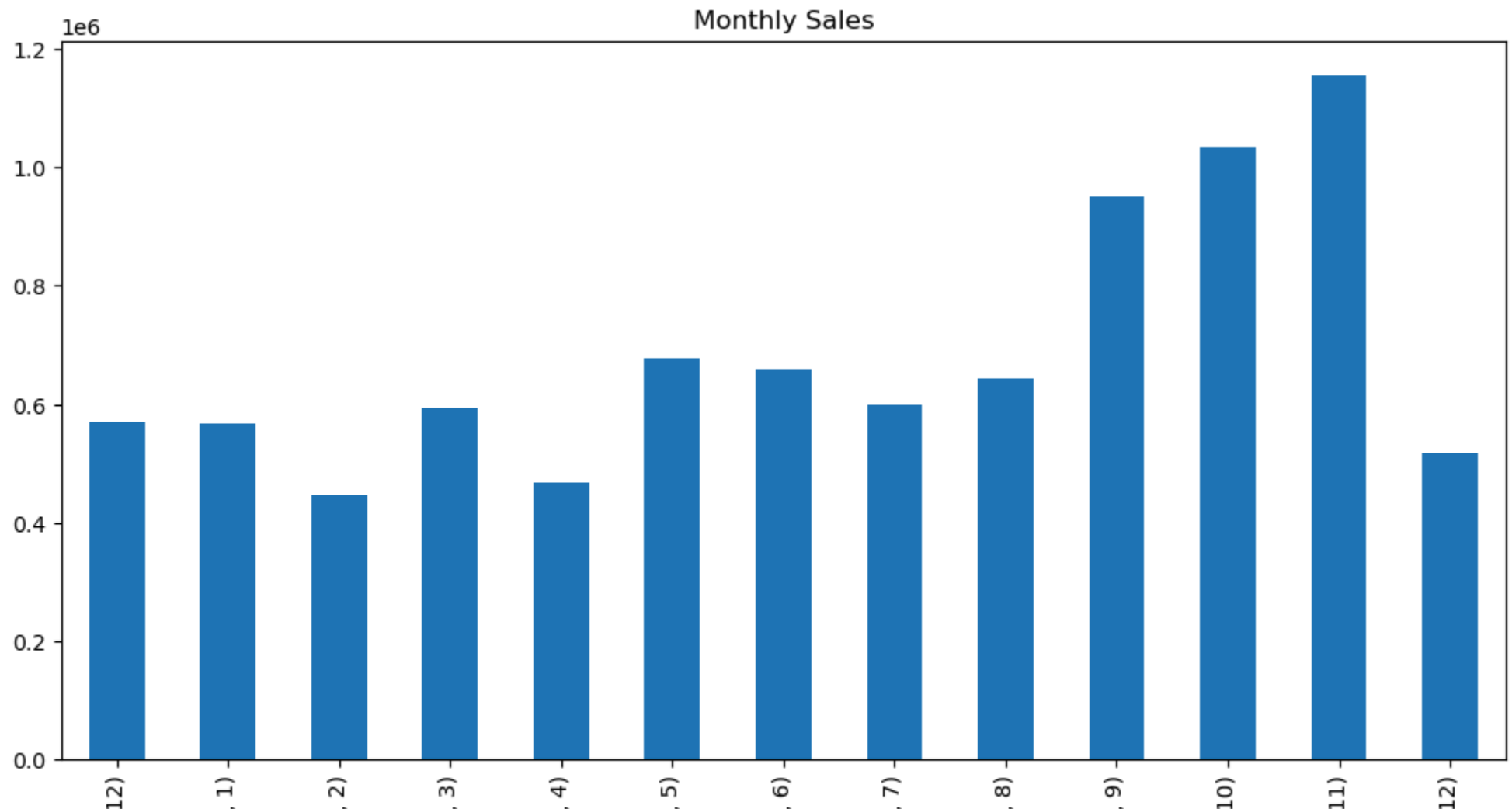


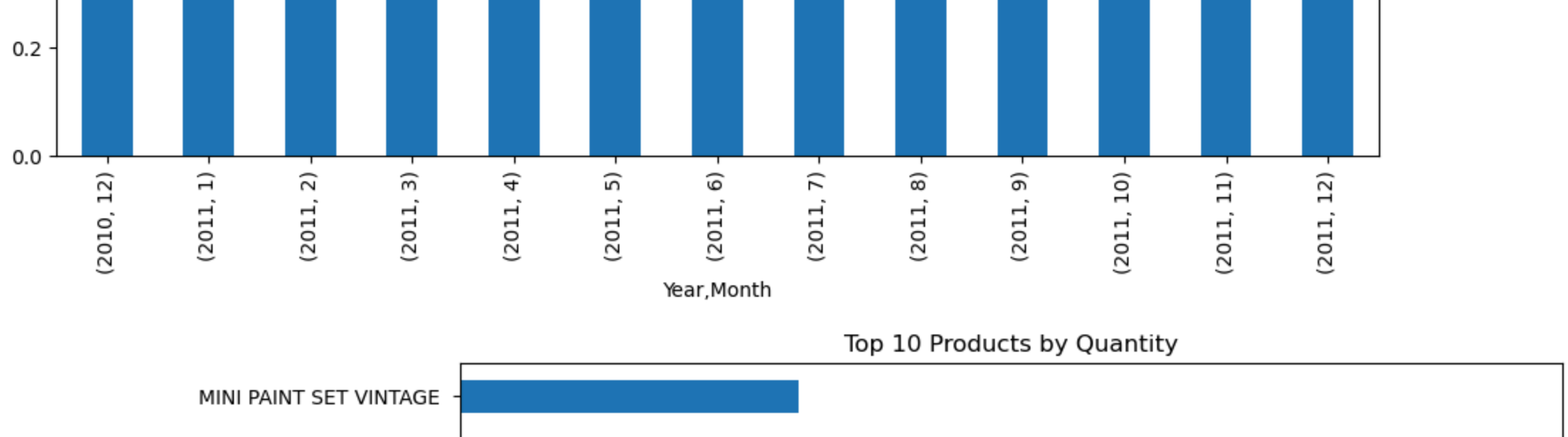

```
Column: TotalPrice
Skewness = 448.52
TotalPrice is Positively Skewed (Right Skewed).
```

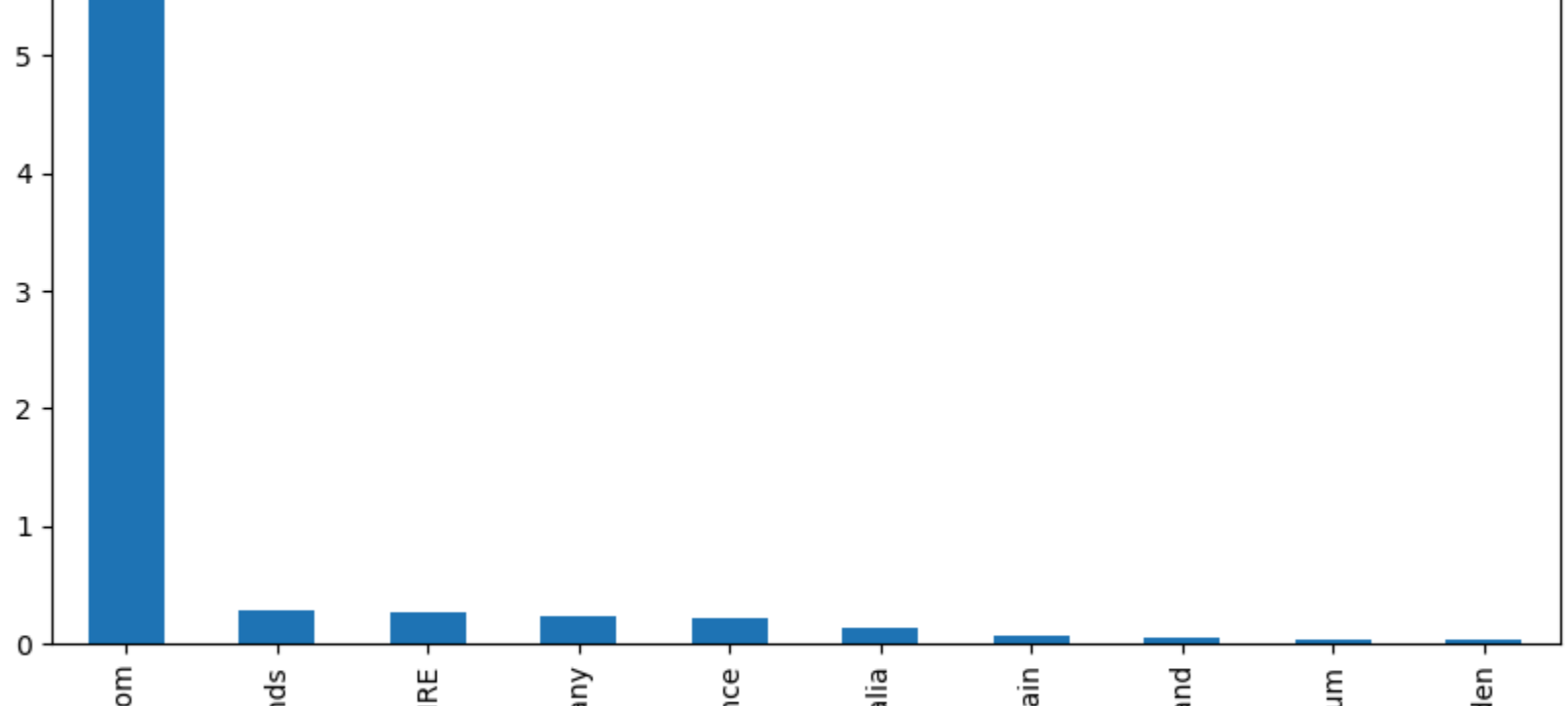
In [65]:
```python
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```

In [87]:
```python
# Feature Engineering:

# Total Price
df['TotalPrice'] = df['Quantity'] * df['UnitPrice']

# Convert InvoiceDate to datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

# Extract time-based features
df['Year'] = df['InvoiceDate'].dt.year
df['Month'] = df['InvoiceDate'].dt.month
df['Weekday'] = df['InvoiceDate'].dt.day_name()
df['Hour'] = df['InvoiceDate'].dt.hour
```

In [89]:
```python
# Sales Analysis:

# Monthly Sales
monthly_sales = df.groupby(['Year','Month'])['TotalPrice'].sum()
monthly_sales.plot(kind='bar', figsize=(12,6), title="Monthly Sales")
plt.show()

# Top 10 Products
top_products = df.groupby('Description')['Quantity'].sum().nlargest(10)
top_products.plot(kind='barh', figsize=(10,6), title="Top 10 Products by Quantity")
plt.show()

# Top 10 Countries
top_countries = df.groupby('Country')['TotalPrice'].sum().nlargest(10)
top_countries.plot(kind='bar', figsize=(10,6), title="Top 10 Countries by Sales")
plt.show()
```







In [67]:
```python
# Binary target (1=high value, 0=low value)
df['HighValue'] = (df['TotalPrice'] > df['TotalPrice'].median()).astype(int)
```

In [92]:
```python
# Data Preprocessing:

# features selection
X = df[['Quantity', 'UnitPrice']]
y = df['HighValue']

# Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Standardize features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

print("Data Preprocessing Done")
```

```
Data Preprocessing Done
```

In [93]:
```python
# Correlation heatmap
plt.figure(figsize=(8,5))
sns.heatmap(df[['Quantity','UnitPrice','TotalPrice']].corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```

In [71]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import classification_report, confusion_matrix

# Train model
model = LogisticRegression()
model.fit(X_train_scaled, y_train)
```

Out[71]:
```
▾ LogisticRegression
LogisticRegression()
```

In [72]:
```python
# Prediction
y_pred = model.predict(X_test_scaled)

# Evaluation
print("Classification Report:\n", classification_report(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
```

```
Classification Report:
               precision    recall  f1-score   support

           0       0.73      0.88      0.79     54252
           1       0.84      0.67      0.75     54130

    accuracy                           0.77    108382
   macro avg       0.78      0.77      0.77    108382
weighted avg       0.78      0.77      0.77    108382

Confusion Matrix:
 [[47486  6766]
 [17849 36281]]
```

In [73]:
```python
# Cross-validation
cv_scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
print("Cross-validation scores:", cv_scores)
print("Average Accuracy:", cv_scores.mean())
```

```
Cross-validation scores: [0.77041391 0.77156723 0.77487037 0.77479655 0.76815124]
Average Accuracy: 0.7719598607725346
```