1. Data Analysis and Preprocessing

In [18]:
```
# Import necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
```

In [19]:
```
# Load the dataset
data = pd.read_csv('advertising.csv')  # replace with your actual data file
```

In [20]:
```
# Display the first few rows of the dataset
print(data.head())
```

```
      TV  Radio  Newspaper  Sales
0  230.1   37.8       69.2   22.1
1   44.5   39.3       45.1   10.4
2   17.2   45.9       69.3   12.0
3  151.5   41.3       58.5   16.5
4  180.8   10.8       58.4   17.9
```

In [21]:
```
# Display summary statistics
print(data.describe())
```

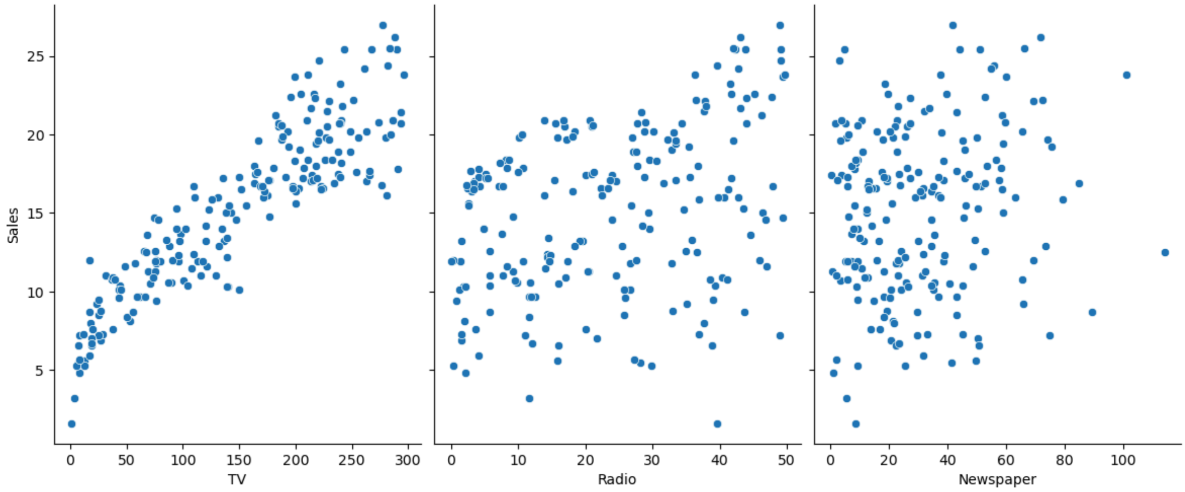
```
               TV       Radio   Newspaper       Sales
count  200.000000  200.000000  200.000000  200.000000
mean   147.042500   23.264000   30.554000   15.130500
std     85.854236   14.846809   21.778621    5.283892
min      0.700000    0.000000    0.300000    1.600000
25%     74.375000    9.975000   12.750000   11.000000
50%    149.750000   22.900000   25.750000   16.000000
75%    218.825000   36.525000   45.100000   19.050000
max    296.400000   49.600000  114.000000   27.000000
```

In [22]:
```
# Check for missing values
print(data.isnull().sum())
```

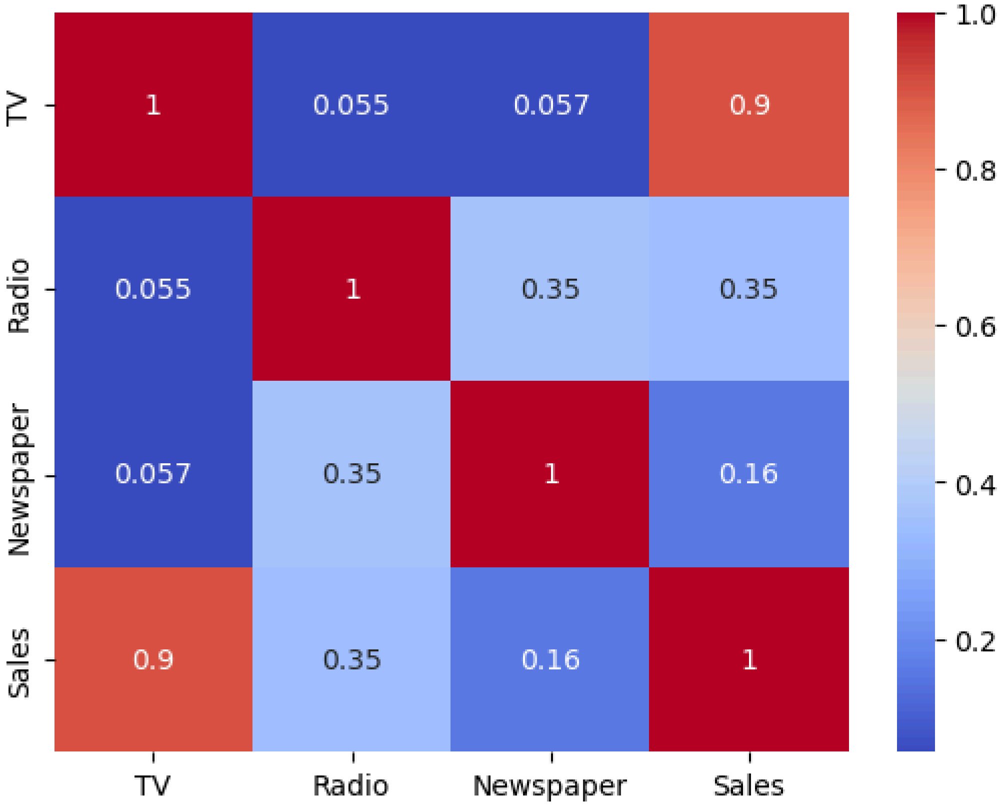
```
TV           0
Radio        0
Newspaper    0
Sales        0
dtype: int64
```

In [23]:
```
# Visualize the relationships between features and target
sns.pairplot(data, x_vars=['TV', 'Radio', 'Newspaper'], y_vars='Sales', height=5, a
plt.show()
```

```
C:\Users\Nishita Bala\anaconda3\Lib\site-packages\seaborn\axisgrid.py:118: UserWar
ning: The figure layout has changed to tight
  self._figure.tight_layout(*args, **kwargs)
```



In [24]:
```
# Correlation matrix
corr_matrix = data.corr()
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm')
plt.show()
```



2. Feature Engineering and Data Splitting

In [25]:
```
# Define features (X) and target (y)
X = data[['TV', 'Radio', 'Newspaper']]
y = data['Sales']
```

In [26]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [27]:
```
# Standardize the features
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

3. Modeling and Evaluation

Linear Regression Model

In [28]:
```
# Train the Linear Regression model
lr_model = LinearRegression()
lr_model.fit(X_train_scaled, y_train)
```

Out[28]:
```
▾ LinearRegression
LinearRegression()
```

In [29]:
```
# Make predictions
y_pred_train = lr_model.predict(X_train_scaled)
y_pred_test = lr_model.predict(X_test_scaled)
```

In [30]:
```
# Evaluate the model
train_rmse = np.sqrt(mean_squared_error(y_train, y_pred_train))
test_rmse = np.sqrt(mean_squared_error(y_test, y_pred_test))
train_r2 = r2_score(y_train, y_pred_train)
test_r2 = r2_score(y_test, y_pred_test)

print(f"Linear Regression Train RMSE: {train_rmse}")
print(f"Linear Regression Test RMSE: {test_rmse}")
print(f"Linear Regression Train R^2: {train_r2}")
print(f"Linear Regression Test R^2: {test_r2}")
```

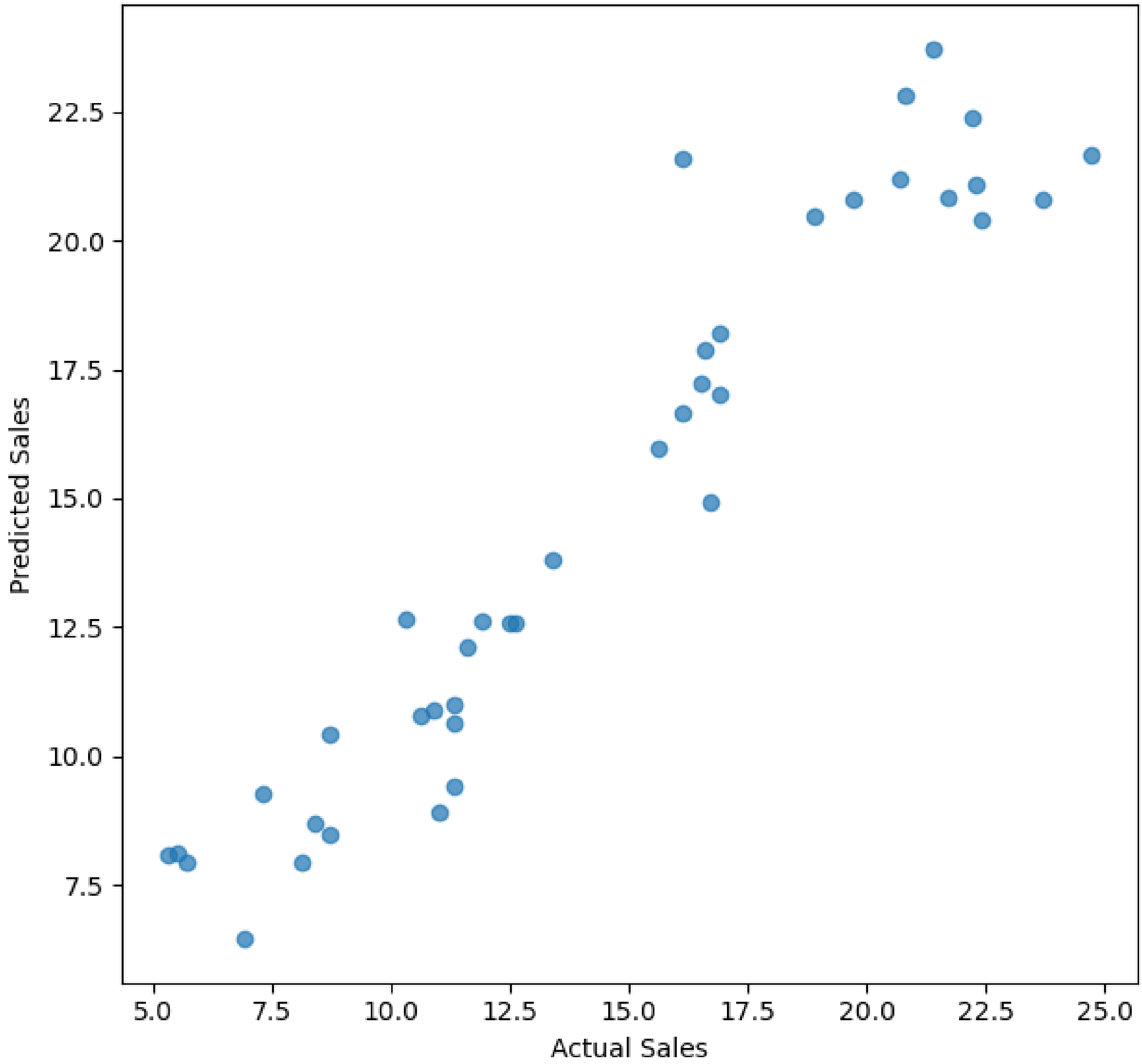
```
Linear Regression Train RMSE: 1.6358920055378559
Linear Regression Test RMSE: 1.7052146229349232
Linear Regression Train R^2: 0.9001416005862131
Linear Regression Test R^2: 0.9059011844150826
```

4. Visualization

In [33]:
```
# Visualize predictions vs actuals
plt.figure(figsize=(12, 6))

plt.subplot(1, 2, 1)
plt.scatter(y_test, y_pred_test, alpha=0.7)
plt.xlabel('Actual Sales')
plt.ylabel('Predicted Sales')

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


Predicted Sales
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0
22.5
25.0
Actual Sales