PROGRAM:

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from statsmodels.tsa.seasonal import seasonal_decompose

# Load your tourism dataset
# Example columns: ['date', 'region', 'tourists', 'spending']
df = pd.read_csv('tourism_data.csv', parse_dates=['date'])

# Basic overview
print(df.head())
print(df.describe())

# Time Series Analysis (aggregated by month)
df['month'] = df['date'].dt.to_period('M')
monthly_data = df.groupby('month')['tourists'].sum().reset_index()
monthly_data['month'] = monthly_data['month'].dt.to_timestamp()

# Decompose time series
```

```
result = seasonal_decompose(monthly_data['tourists'], model='additive', period=12)

result.plot()

plt.title('Tourism Seasonality & Trend')

plt.show()


# Clustering Regions Based on Average Tourists and Spending

region_stats = df.groupby('region')[['tourists', 'spending']].mean()

scaler = StandardScaler()

scaled_data = scaler.fit_transform(region_stats)


kmeans = KMeans(n_clusters=3, random_state=42)

region_stats['cluster'] = kmeans.fit_predict(scaled_data)


# Plot Clusters

sns.scatterplot(data=region_stats, x='tourists', y='spending', hue='cluster', palette='viridis')

plt.title('Tourism Region Clusters')

plt.xlabel('Average Tourists')

plt.ylabel('Average Spending')

plt.show()


# Decision-making insights

for cluster in region_stats['cluster'].unique():

    print(f"\nCluster {cluster}:")
```

print(region_stats[region_stats['cluster'] == cluster])

 # Save clustered data

region_stats.to_csv('clustered_regions.csv')

OUTPUT:

1. Data Loading and Exploration

```
python
CopyEdit
import pandas as pd

df = pd.read_csv('tourism_data.csv', parse_dates=['date'])
print(df.head())
print(df.describe())
```

- **Purpose**: Loads a CSV file containing tourism data, parsing the 'date' column as datetime objects. It then displays the first few rows and a statistical summary of the dataset.

📆 2. Time Series Analysis

```
python
CopyEdit
df['month'] = df['date'].dt.to_period('M')
monthly_data = df.groupby('month')['tourists'].sum().reset_index()
monthly_data['month'] = monthly_data['month'].dt.to_timestamp()
```

- **Purpose**: Aggregates the number of tourists by month, converting the 'month' period to a timestamp for further analysis.

🔍 3. Seasonal Decomposition

```
python
CopyEdit
from statsmodels.tsa.seasonal import seasonal_decompose
import matplotlib.pyplot as plt
```

```
result = seasonal_decompose(monthly_data['tourists'], model='additive',
period=12)
result.plot()
plt.title('Tourism Seasonality & Trend')
plt.show()
```

- **Purpose**: Performs seasonal decomposition of the time series data to extract and visualize the trend, seasonal, and residual components.

🌍 4. Regional Clustering

```
python
CopyEdit
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

region_stats = df.groupby('region')[['tourists', 'spending']].mean()
scaler = StandardScaler()
scaled_data = scaler.fit_transform(region_stats)

kmeans = KMeans(n_clusters=3, random_state=42)
region_stats['cluster'] = kmeans.fit_predict(scaled_data)
```

- **Purpose**: Clusters regions based on the average number of tourists and spending using K-Means clustering. The data is standardized before clustering to ensure fair weighting of features.

📈 5. Visualization of Clusters

```
python
CopyEdit
import seaborn as sns

sns.scatterplot(data=region_stats, x='tourists', y='spending',
hue='cluster', palette='viridis')
plt.title('Tourism Region Clusters')
plt.xlabel('Average Tourists')
plt.ylabel('Average Spending')
plt.show()
```

- **Purpose**: Visualizes the clustering results, showing how regions group based on tourist numbers and spending patterns.

🧠 6. Insights by Cluster

```
python
CopyEdit
for cluster in region_stats['cluster'].unique():
    print(f"\nCluster {cluster}:")
```

```
print(region_stats[region_stats['cluster'] == cluster])
```

- **Purpose**: Provides a detailed view of each cluster, helping to interpret the characteristics of regions within each group.

💾 7. Saving the Results

```
python
CopyEdit
region_stats.to_csv('clustered_regions.csv')
```

- **Purpose**: Saves the clustered regional data to a CSV file for further analysis or reporting.

□ Potential Enhancements

- **Seasonal Model Selection**: Experiment with 'multiplicative' models in seasonal decomposition if the data exhibits exponential growth patterns.
- **Clustering Validation**: Use methods like the elbow method or silhouette score to determine the optimal number of clusters.
- **Advanced Clustering Techniques**: Explore hierarchical clustering or DBSCAN for potentially better clustering results, especially if the data has non-linear relationships.
- **Time Series Forecasting**: Incorporate forecasting models like ARIMA or Prophet to predict future tourism trends.
- **Interactive Dashboards**: Utilize libraries like Plotly or Dash to create interactive visualizations for more dynamic data exploration.