Practical 2

You have been given a web analytics data which comprises of the following features:

User_ID: Unique identifier for each user.

Session_ID: Unique session identifier for tracking user activity.

Page_Views: The number of pages viewed during the session.

Session_Duration: The duration of the session (in minutes).

Bounce_Rate: Whether the user bounced (1 for bounce, 0 for no bounce).

Device: The type of device used (Desktop, Mobile, Tablet).

Country: The country where the user is located.

Traffic_Source: The source of traffic (Organic, Direct, Referral, Social Media, Paid).

Conversions: Whether the session resulted in a conversion (1 for yes, 0 for no).

1) Analyze Traffic Sources

Using the dataset, determine which traffic source (Organic, Direct, Referral, Social Media, or Paid) drives the most traffic (page views and sessions) and has the highest conversion rate.

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('web_analytics_data.csv')

# Grouping by Traffic_Source to find total Page_Views and Sessions
traffic_analysis = df.groupby('Traffic_Source').agg(
    total_page_views=('Page_Views', 'sum'),
    total_sessions=('Session_ID', 'count'),
    total_conversions=('Conversions', 'sum')
).reset_index()

# Calculating Conversion Rate for each traffic source
traffic_analysis['conversion_rate'] = traffic_analysis['total_conversions'] / traffic_analysis['total_sessions'] * 100

# Sorting by total page views and conversion rate
traffic_analysis_sorted = traffic_analysis.sort_values(by=['total_page_views', 'conversion_rate'], ascending=False)

# Displaying the traffic analysis
print(traffic_analysis_sorted)
```

Output:

```
  Traffic_Source  total_page_views  total_sessions  total_conversions
0         Direct              1739             232                 36
4   Social Media              1505             192                 32
2           Paid              1497             210                 28
1        Organic              1387             196                 28
3       Referral              1176             170                 17

   conversion_rate
0        15.517241
4        16.666667
2        13.333333
1        14.285714
3        10.000000
```

2) Device-Based Bounce Rate Analysis

Calculate and compare the bounce rates for each device type (Desktop, Mobile, Tablet). Which device has the highest bounce rate?

```python
import pandas as pd

# Load the dataset
df = pd.read_csv('web_analytics_data.csv')

# Calculate bounce rate for each device type
# Bounce rate = (Number of sessions with bounce / Total sessions) * 100
bounce_rates = df.groupby('Device')['Bounce_Rate'].mean() * 100

# Display the bounce rates for each device
print("Bounce rates for each device type:")
print(bounce_rates)

# Find the device with the highest bounce rate
highest_bounce_device = bounce_rates.idxmax()
highest_bounce_rate = bounce_rates.max()

print(f"\nThe device with the highest bounce rate is: {highest_bounce_device} ({highest_bounce_rate:.2f}%)")
```

Output:

```
Bounce rates for each device type:
Device
Desktop    29.878049
Mobile     30.182927
Tablet     32.848837
Name: Bounce_Rate, dtype: float64

The device with the highest bounce rate is: Tablet (32.85%)
```

Group by Device: The code groups the dataset by the Device column (Desktop, Mobile, Tablet) and calculates the average bounce rate for each group.

Highest Bounce Rate: The device type with the highest bounce rate is identified using idxmax() to find the maximum value.

3) Country-Based User Behavior

Analyze the average session duration and page views per session for users from different countries (USA, UK, Canada, Germany, France). Which country shows the most engagement?

```python
import pandas as pd

# Load your dataset
df = pd.read_csv('web_analytics_data.csv')

# Calculate average session duration and page views per session by country
engagement_by_country = df.groupby('Country').agg({
    'Session_Duration': 'mean',
    'Page_Views': 'mean'
}).reset_index()

# Rename the columns for clarity
engagement_by_country.columns = ['Country', 'Avg_Session_Duration', 'Avg_Page_Views']

# Add an engagement score by combining both metrics (equal weighting for simplicity)
engagement_by_country['Engagement_Score'] = (engagement_by_country['Avg_Session_Duration'] + engagement_by_country['Avg_Page_Views']) / 2

# Find the country with the highest engagement score
most_engaged_country = engagement_by_country.loc[engagement_by_country['Engagement_Score'].idxmax()]

# Display the average session duration and page views per session by country
print("Average Session Duration and Page Views per Session by Country:")
print(engagement_by_country)

# Display the country with the highest engagement score
print("\nCountry with the Highest Engagement:")
print(most_engaged_country)
```

Output:

```
Average Session Duration and Page Views per Session by Country:
   Country  Avg_Session_Duration  Avg_Page_Views  Engagement_Score
0   Canada              9.494039        7.591133          8.542586
1   France             10.160637        7.514706          8.837672
2  Germany             10.727678        7.194313          8.960995
3       UK             10.073655        7.319797          8.696726
4      USA             10.574108        6.864865          8.719486

Country with the Highest Engagement:
Country                 Germany
Avg_Session_Duration  10.727678
Avg_Page_Views         7.194313
Engagement_Score       8.960995
Name: 2, dtype: object
```

Group by Country: The code groups the dataset by Country and calculates the average session duration and page views for each country.

Engagement Score: An engagement score is calculated by averaging the session duration and page views, assuming equal importance for both.

Most Engaged Country: The country with the highest engagement score is identified.

4) Conversion Rate Analysis

Determine the overall conversion rate and compare it across traffic sources. Identify which traffic source has the highest conversion rate.

```
import pandas as pd

# Load your dataset
df = pd.read_csv('web_analytics_data.csv')

# Calculate the overall conversion rate
overall_conversion_rate = df['Conversions'].mean()

# Calculate conversion rate for each traffic source
conversion_rate_by_source = df.groupby('Traffic_Source')['Conversions'].mean().reset_index()

# Rename columns for clarity
conversion_rate_by_source.columns = ['Traffic_Source', 'Conversion_Rate']

# Find the traffic source with the highest conversion rate
highest_conversion_rate_source = conversion_rate_by_source.loc[conversion_rate_by_source['Conversion_Rate'].idxmax()]

# Display overall conversion rate
print(f"Overall Conversion Rate: {overall_conversion_rate:.2%}")

# Display conversion rate by traffic source
print("\nConversion Rate by Traffic Source:")
print(conversion_rate_by_source)

# Display the traffic source with the highest conversion rate
print("\nTraffic Source with the Highest Conversion Rate:")
print(highest_conversion_rate_source)
```

Output:

```
Overall Conversion Rate: 14.10%

Conversion Rate by Traffic Source:
  Traffic_Source  Conversion_Rate
0         Direct         0.155172
1        Organic         0.142857
2           Paid         0.133333
3       Referral         0.100000
4   Social Media         0.166667

Traffic Source with the Highest Conversion Rate:
Traffic_Source     Social Media
Conversion_Rate        0.166667
Name: 4, dtype: object
```

Overall Conversion Rate: The mean of the Conversions column is calculated to determine the overall conversion rate.

Conversion Rate by Traffic Source: The code groups the data by Traffic_Source and calculates the average conversion rate for each.

Highest Conversion Rate Source: The traffic source with the highest conversion rate is identified using idxmax().

5) Predictive Modeling

Use logistic regression to predict whether a session will result in a conversion based on features like session duration, page views, and traffic source.

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.metrics import accuracy_score

# Load the dataset
df = pd.read_csv('web_analytics_data.csv')

# Features for the logistic regression model
features = ['Session_Duration', 'Page_Views', 'Traffic_Source']

# One-hot encode the categorical variable 'Traffic_Source'
encoder = OneHotEncoder(drop='first')
traffic_source_encoded = encoder.fit_transform(df[['Traffic_Source']]).toarray() # Convert to dense array
traffic_source_encoded_df = pd.DataFrame(traffic_source_encoded, columns=encoder.get_feature_names_out(['Traffic_Source']))

# Combine encoded 'Traffic_Source' with other features
X = pd.concat([df[['Session_Duration', 'Page_Views']], traffic_source_encoded_df], axis=1)
```

```
# Target variable (Conversion: 1 or 0)
y = df['Conversions']

# Split data into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Initialize and train the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model's accuracy
accuracy = accuracy_score(y_test, y_pred)

# Display the accuracy of the model
print(f"Model Accuracy: {accuracy:.2%}")
```

```
Model Accuracy: 87.00%
```

One-Hot Encoding: The Traffic_Source categorical variable is one-hot encoded, turning it into multiple binary columns for logistic regression.

Model Training: A logistic regression model is trained using Session_Duration, Page_Views, and Traffic_Source features.

Model Evaluation: The model's performance is evaluated using accuracy on the test set.