```
import pandas as pd

# Load the dataset
file_path = '/content/mobile_year_2022_quarter_03.csv'
data = pd.read_csv(file_path)

# Display the first few rows to inspect the structure
print("Preview of the Dataset:")
print(data.head())

# Check basic information about the dataset
print("\nDataset Info:")
print(data.info())
```

```
Preview of the Dataset:
                             Name Number of Records Devices Tests  \
0                      Antarctica                 1       1     1
1     Falkland Islands (Malvinas)                 1       1     1
2  British Indian Ocean Territory                 1       1     2
3                  Norfolk Island                 5       5    14
4                        Kiribati                 5       5    14

  Avg. Avg U Kbps Avg. Avg D Kbps Avg Lat Ms Avg. Pop2005  Rank Upload  \
0           4,568             103      1,263            0          223
1           1,057           4,131        811        2,975          231
2             863           2,878        749            0          232
3           3,467          29,444        576            0          226
4           2,770           4,071        560       92,003          228

   Rank Download  Rank Latency
0            233             1
1            229             2
2            231             3
3            130             4
4            230             5

Dataset Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 233 entries, 0 to 232
Data columns (total 11 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Name               233 non-null    object
 1   Number of Records  233 non-null    object
 2   Devices            233 non-null    object
 3   Tests              233 non-null    object
 4   Avg. Avg U Kbps    233 non-null    object
 5   Avg. Avg D Kbps    233 non-null    object
 6   Avg Lat Ms         233 non-null    object
 7   Avg. Pop2005       233 non-null    object
 8   Rank Upload        233 non-null    int64
 9   Rank Download      233 non-null    int64
 10  Rank Latency       233 non-null    int64
dtypes: int64(3), object(8)
memory usage: 20.1+ KB
None
```

```
# Standardize column names for ease of use (lowercase and replace spaces with underscores)
data.columns = data.columns.str.strip().str.replace(' ', '_').str.lower()

print("\nStandardized Column Names:")
print(data.columns)
```

```
Standardized Column Names:
Index(['name', 'number_of_records', 'devices', 'tests', 'avg._avg_u_kbps',
       'avg._avg_d_kbps', 'avg_lat_ms', 'avg._pop2005', 'rank_upload',
       'rank_download', 'rank_latency'],
      dtype='object')
```

```
# Check for missing values
print("\nMissing Values Before Cleaning:")
print(data.isnull().sum())

# Fill numeric columns with their mean value if missing
numeric_cols = data.select_dtypes(include=['float64', 'int64']).columns
data[numeric_cols] = data[numeric_cols].fillna(data[numeric_cols].mean())

# Recheck missing values
print("\nMissing Values After Cleaning:")
print(data.isnull().sum())
```

```
Missing Values Before Cleaning:
name                  0
number_of_records     0
devices               0
tests                 0
avg._avg_u_kbps       0
avg._avg_d_kbps       0
avg_lat_ms            0
avg._pop2005          0
rank_upload           0
rank_download         0
rank_latency          0
dtype: int64

Missing Values After Cleaning:
name                  0
number_of_records     0
devices               0
tests                 0
avg._avg_u_kbps       0
avg._avg_d_kbps       0
avg_lat_ms            0
avg._pop2005          0
rank_upload           0
rank_download         0
rank_latency          0
dtype: int64
```

```
# Convert rank columns to integers (if applicable)
rank_cols = ['rank_upload', 'rank_download', 'rank_latency']

for col in rank_cols:
    if col in data.columns:
        data[col] = pd.to_numeric(data[col], errors='coerce').fillna(0).astype(int)

print("\nData Types After Adjustment:")
print(data.dtypes)
```

```
Data Types After Adjustment:
name                  object
number_of_records     object
devices               object
tests                 object
avg._avg_u_kbps       object
avg._avg_d_kbps       object
avg_lat_ms            object
avg._pop2005          object
rank_upload            int64
rank_download          int64
rank_latency           int64
dtype: object
```

```
# Remove duplicates from the dataset
data = data.drop_duplicates()

print("\nNumber of Records After Removing Duplicates:", len(data))
```

```
Number of Records After Removing Duplicates: 233
```

```
# Clip numeric columns at the 99th percentile to handle outliers
outlier_cols = ['avg_u_kbps', 'avg_d_kbps', 'avg_lat_msavg']

for col in outlier_cols:
    if col in data.columns:
        upper_limit = data[col].quantile(0.99)
        data[col] = data[col].clip(upper=upper_limit)

print("\nOutliers Handled for Columns:", outlier_cols)
```

```
Outliers Handled for Columns: ['avg_u_kbps', 'avg_d_kbps', 'avg_lat_msavg']
```

```
# Filter data for India
india_data = data[data['name'].str.lower() == 'india']

print("\nIndia's Data:")
print(india_data)
```

```
India's Data:
     name number_of_records    devices      tests avg._avg_u_kbps  \
65  India           406,467  1,307,405  2,766,067           6,434

   avg._avg_d_kbps avg_lat_ms   avg._pop2005  rank_upload  rank_download  \
65          19,306         51  1,134,403,141          216            183

    rank_latency
65            63
```

```
# Calculate global averages for numeric columns
global_averages = data.mean(numeric_only=True)

print("\nGlobal Averages:")
print(global_averages)
```

```
Global Averages:
rank_upload      116.995708
rank_download    117.000000
rank_latency     114.648069
dtype: float64
```

```
# Create a comparison DataFrame
comparison = pd.DataFrame({
    'India': india_data.mean(numeric_only=True),
    'Global Average': global_averages
}).dropna()

print("\nComparison of India vs Global Averages:")
print(comparison)
```

```
Comparison of India vs Global Averages:
               India  Global Average
rank_upload    216.0      116.995708
rank_download  183.0      117.000000
rank_latency    63.0      114.648069
```

Start coding or generate with AI.

```
# Display all column names in the dataset to verify the correct names
print("\nColumns in the dataset:")
print(data.columns)
```

```
Columns in the dataset:
Index(['name', 'number_of_records', 'devices', 'tests', 'avg._avg_u_kbps',
       'avg._avg_d_kbps', 'avg_lat_ms', 'avg._pop2005', 'rank_upload',
       'rank_download', 'rank_latency'],
      dtype='object')
```

```
# Clean and convert columns to numeric values
data['avg._avg_u_kbps'] = data['avg._avg_u_kbps'].str.replace(',', '').astype(float)
data['avg._avg_d_kbps'] = data['avg._avg_d_kbps'].str.replace(',', '').astype(float)
data['avg_lat_ms'] = data['avg_lat_ms'].str.replace(',', '').astype(float)

# Verify the conversion
print(data[['avg._avg_u_kbps', 'avg._avg_d_kbps', 'avg_lat_ms']].head())
```

```
   avg._avg_u_kbps  avg._avg_d_kbps  avg_lat_ms
0           4568.0            103.0      1263.0
1           1057.0           4131.0       811.0
2            863.0           2878.0       749.0
3           3467.0          29444.0       576.0
4           2770.0           4071.0       560.0
```

```
# Group data by country and calculate mean for relevant metrics
country_metrics = data.groupby('name')[['avg._avg_u_kbps', 'avg._avg_d_kbps', 'avg_lat_ms']].mean()

# Display the aggregated metrics
print("\nCountry Metrics (Mean Values):")
print(country_metrics.head())
```

```
Country Metrics (Mean Values):
                avg._avg_u_kbps  avg._avg_d_kbps  avg_lat_ms
name
Afghanistan              4257.0           7313.0        52.0
Albania                 11916.0          38413.0        44.0
Algeria                 11656.0          19646.0        41.0
American Samoa           7275.0          18894.0        35.0
Andorra                 20018.0         103177.0        46.0
```

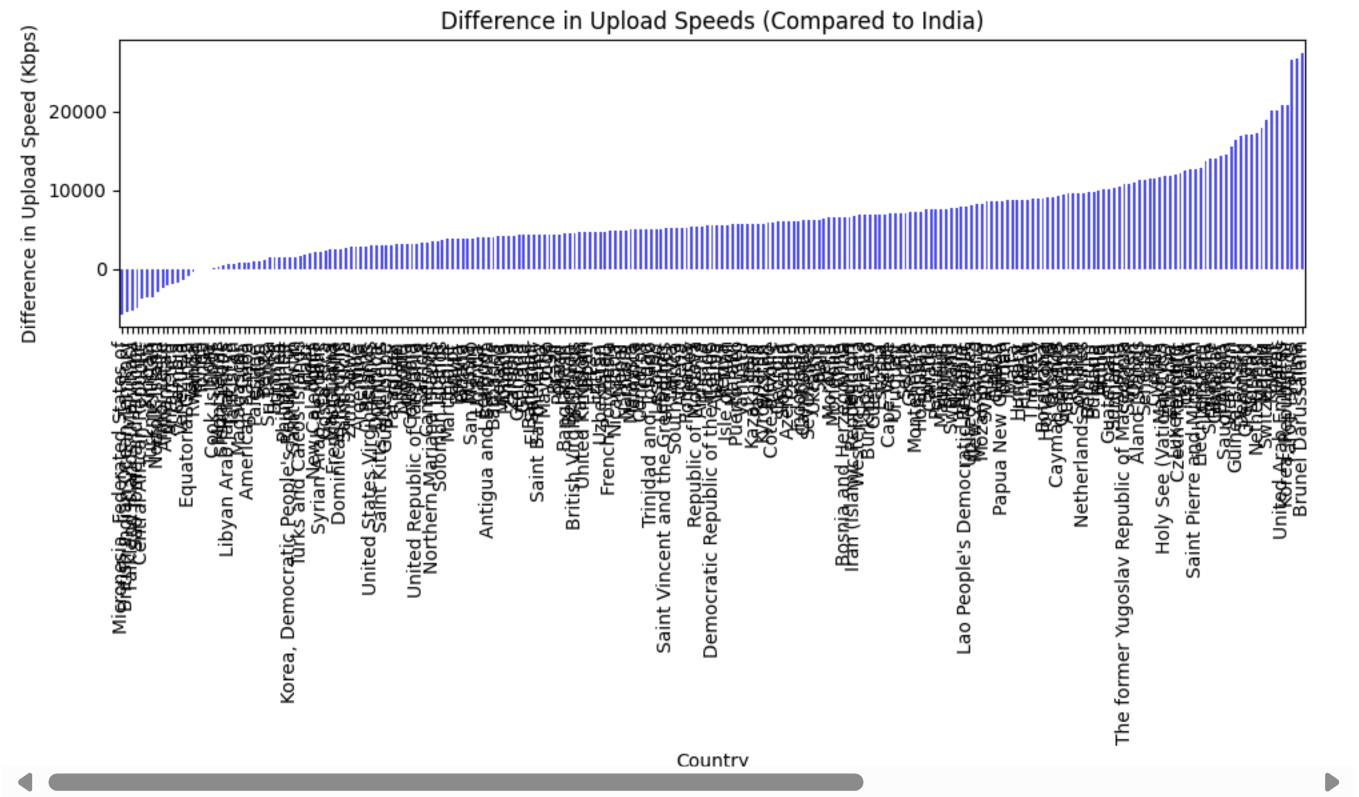
```
# Extract India's metrics for comparison
india_metrics = country_metrics.loc['India']

print("\nIndia's Metrics:")
print(india_metrics)
```

```
India's Metrics:
avg._avg_u_kbps     6434.0
avg._avg_d_kbps    19306.0
avg_lat_ms            51.0
Name: India, dtype: float64
```

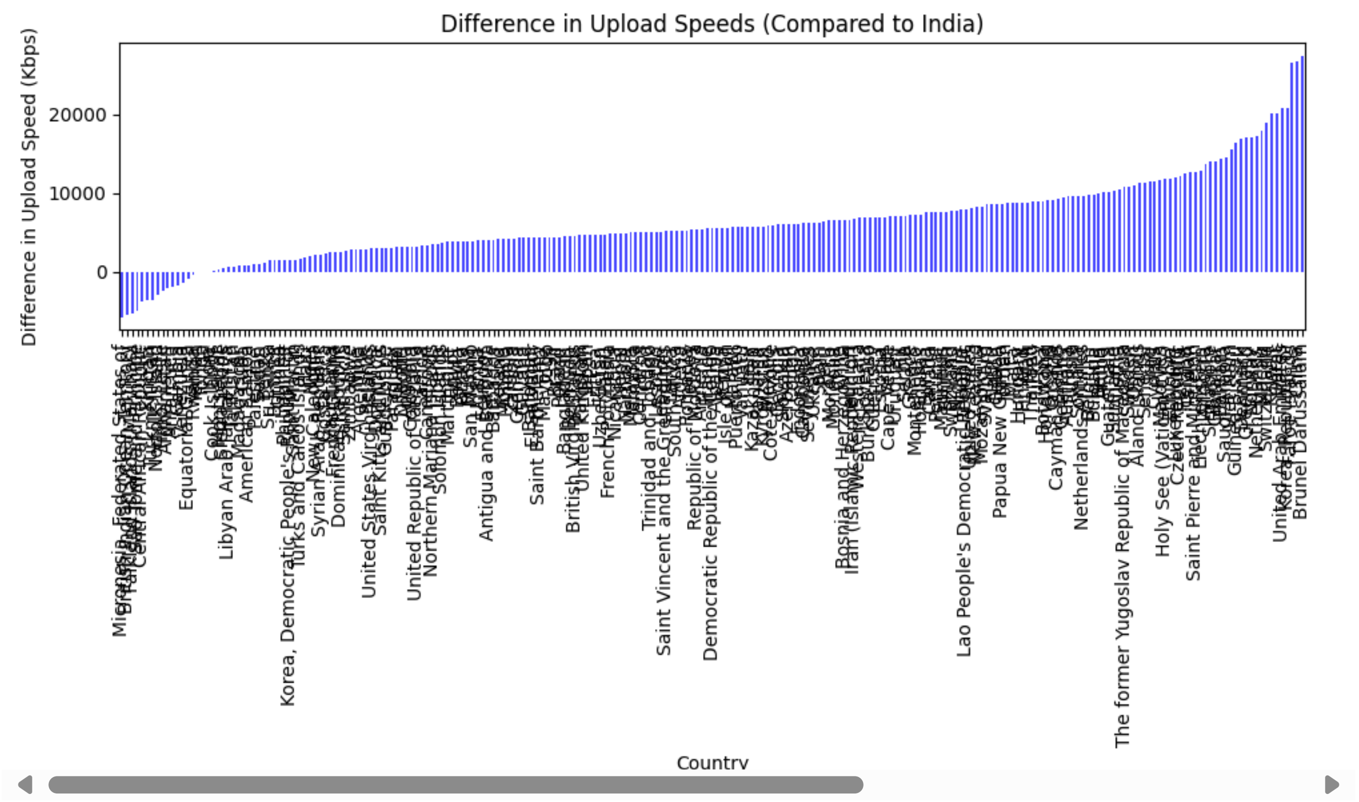
```
# Add a column for the difference compared to India
comparison_df = country_metrics.copy()
comparison_df['Difference_Upload'] = comparison_df['avg._avg_u_kbps'] - india_metrics['avg._avg_u_kbps']
comparison_df['Difference_Download'] = comparison_df['avg._avg_d_kbps'] - india_metrics['avg._avg_d_kbps']
comparison_df['Difference_Latency'] = comparison_df['avg_lat_ms'] - india_metrics['avg_lat_ms']

# Sort data for visualization (optional)
comparison_df = comparison_df.sort_values(by='Difference_Download', ascending=False)

print("\nComparison DataFrame:")
print(comparison_df.head())
```

```
Comparison DataFrame:
                      avg._avg_u_kbps  avg._avg_d_kbps  avg_lat_ms  \
name
Korea, Republic of            27181.0         248599.0        34.0
United Arab Emirates          27173.0         210427.0        35.0
Kuwait                        26482.0         195254.0        21.0
Qatar                         26532.0         195085.0        29.0
China                         33173.0         161880.0        32.0

                      Difference_Upload  Difference_Download  \
name
Korea, Republic of              20747.0             229293.0
United Arab Emirates            20739.0             191121.0
Kuwait                          20048.0             175948.0
Qatar                           20098.0             175779.0
China                           26739.0             142574.0

                      Difference_Latency
name
Korea, Republic of                 -17.0
United Arab Emirates               -16.0
Kuwait                             -30.0
Qatar                              -22.0
China                              -19.0
```

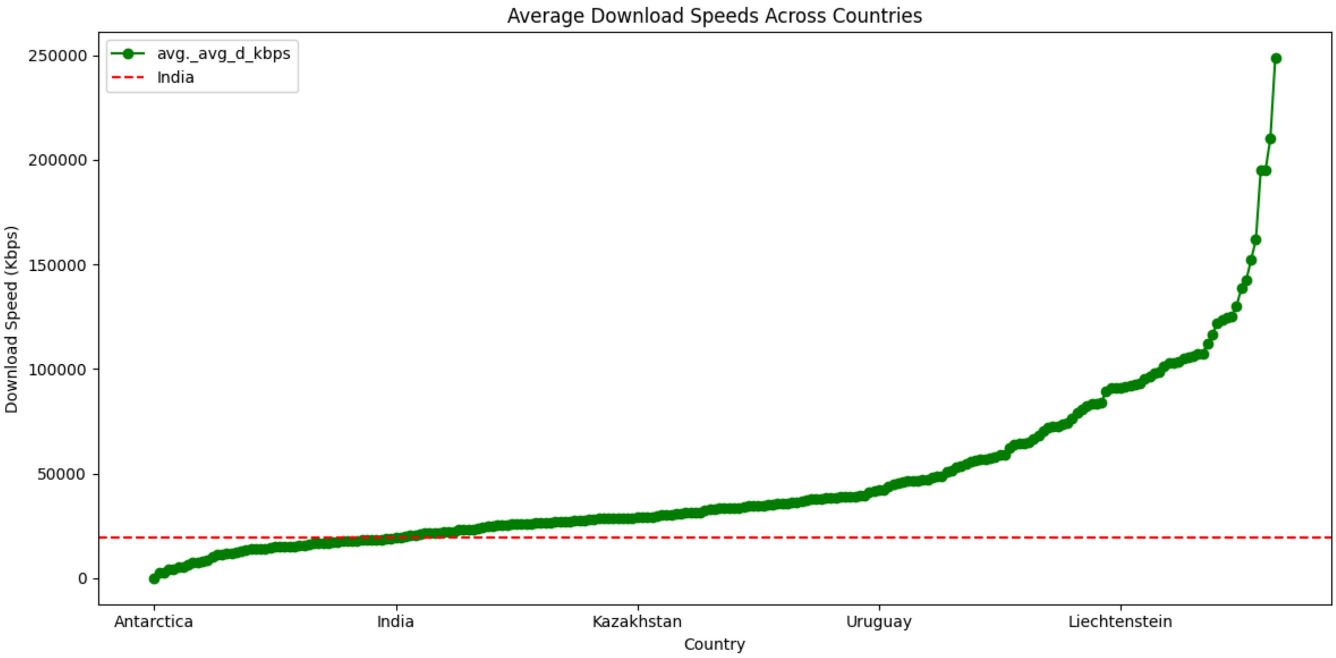
```
# Bar plot of upload speed comparison
import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
comparison_df['Difference_Upload'].sort_values().plot(kind='bar', color='blue', alpha=0.7)
plt.title('Difference in Upload Speeds (Compared to India)')
plt.ylabel('Difference in Upload Speed (Kbps)')
plt.xlabel('Country')
plt.tight_layout()
plt.show()
```



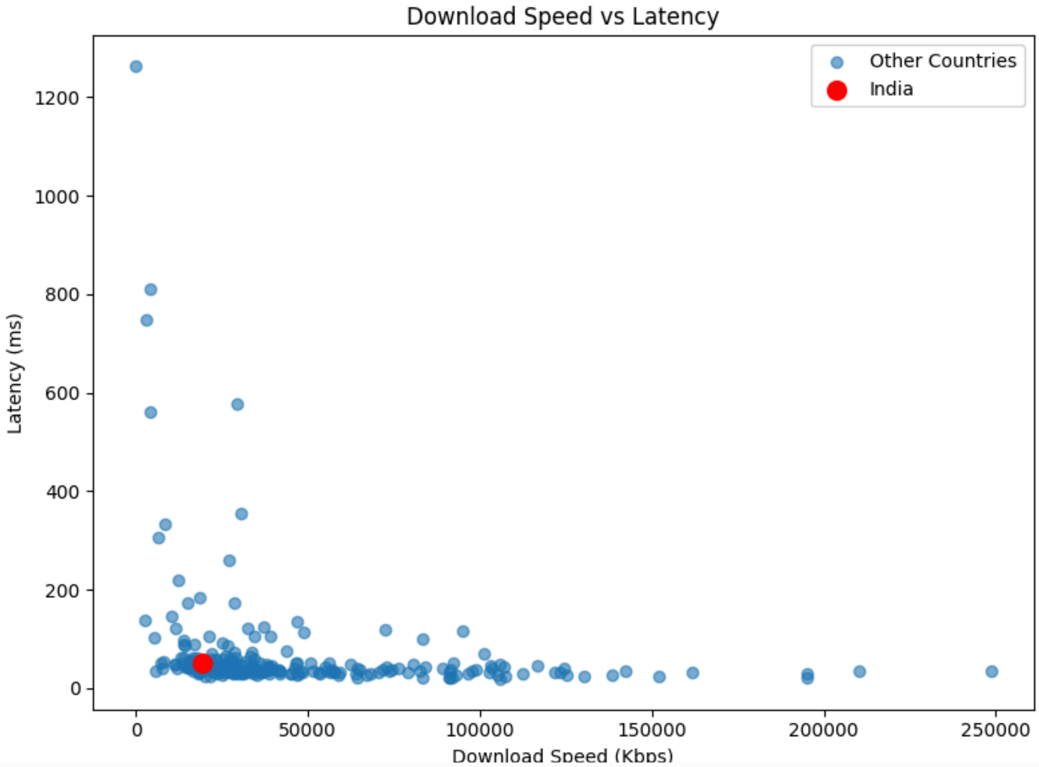
```
import matplotlib.pyplot as plt

# Bar plot of upload speed comparison
plt.figure(figsize=(10, 6))
comparison_df['Difference_Upload'].sort_values().plot(kind='bar', color='blue', alpha=0.7)
plt.title('Difference in Upload Speeds (Compared to India)')
plt.ylabel('Difference in Upload Speed (Kbps)')
plt.xlabel('Country')
plt.tight_layout()
plt.show()
```



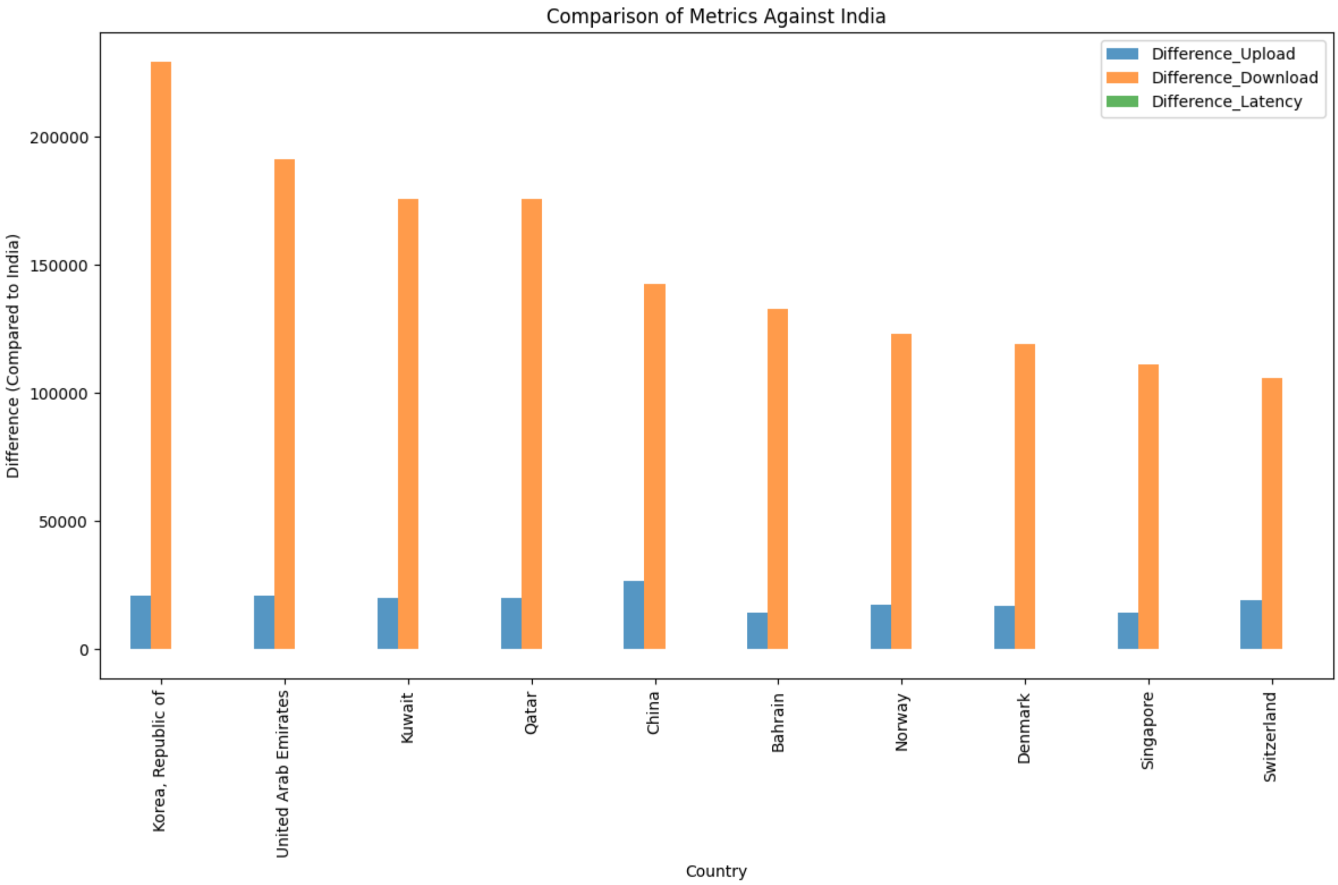
```
# Line plot for download speeds
plt.figure(figsize=(12, 6))
country_metrics['avg._avg_d_kbps'].sort_values().plot(kind='line', marker='o', color='green')
plt.axhline(y=india_metrics['avg._avg_d_kbps'], color='red', linestyle='--', label="India")
plt.title('Average Download Speeds Across Countries')
plt.ylabel('Download Speed (Kbps)')
plt.xlabel('Country')
plt.legend()
plt.tight_layout()
plt.show()
```



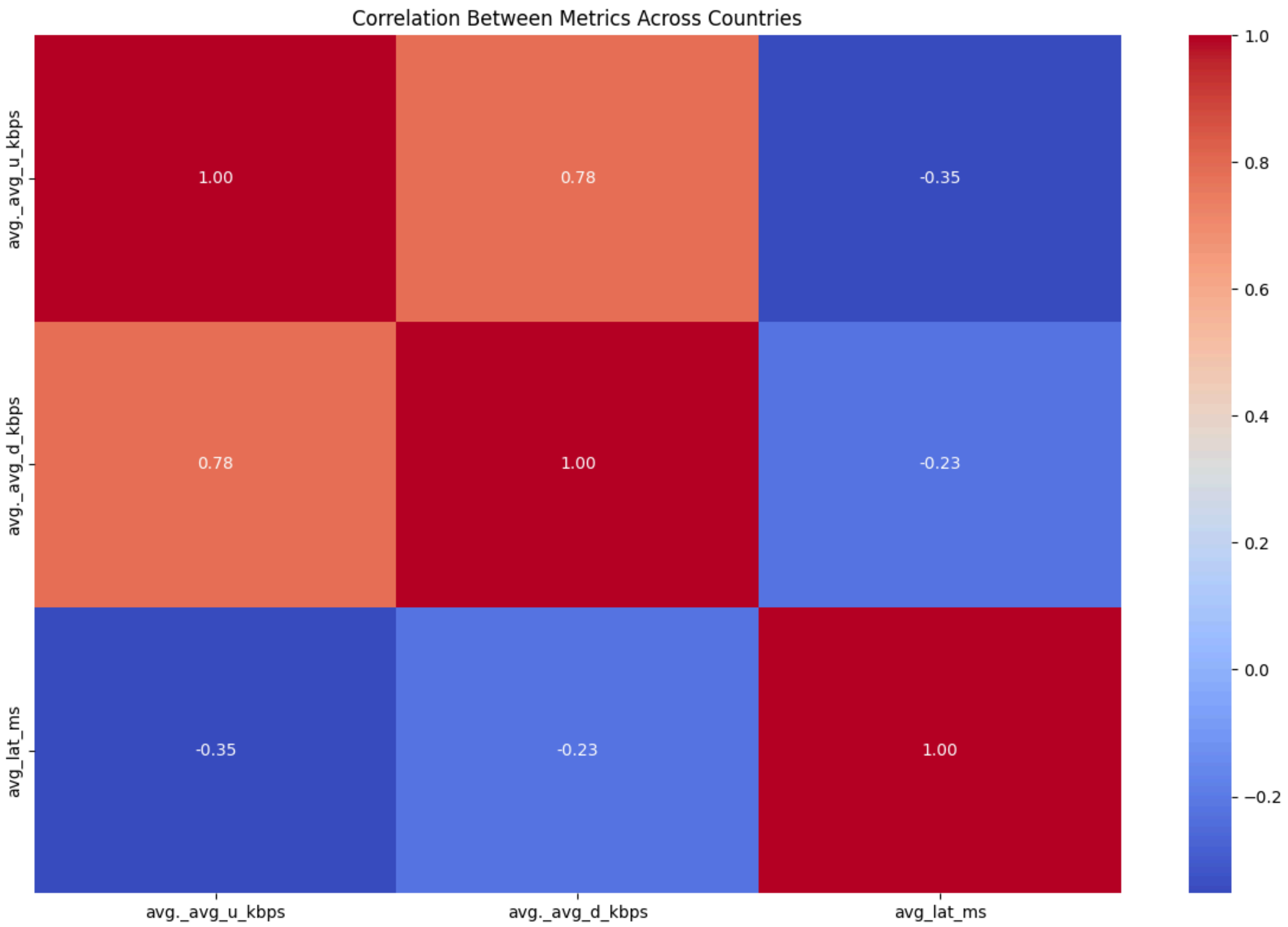
```
# Scatter plot: Download speed vs Latency
plt.figure(figsize=(8, 6))
plt.scatter(country_metrics['avg._avg_d_kbps'], country_metrics['avg_lat_ms'], alpha=0.6, label='Other Countries')
plt.scatter(india_metrics['avg._avg_d_kbps'], india_metrics['avg_lat_ms'], color='red', label='India', s=100)
plt.title('Download Speed vs Latency')
plt.xlabel('Download Speed (Kbps)')
plt.ylabel('Latency (ms)')
plt.legend()
plt.tight_layout()
plt.show()
```



```
# Bar chart to compare upload, download, and latency differences
comparison_df[['Difference_Upload', 'Difference_Download', 'Difference_Latency']].head(10).plot(
    kind='bar', figsize=(12, 8), alpha=0.75
)
plt.title('Comparison of Metrics Against India')
plt.ylabel('Difference (Compared to India)')
plt.xlabel('Country')
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```



```
import seaborn as sns

# Create a heatmap for the correlations between metrics
plt.figure(figsize=(12, 8))
sns.heatmap(
    country_metrics.corr(),
    annot=True,
    cmap='coolwarm',
    fmt='.2f'
)
plt.title('Correlation Between Metrics Across Countries')
plt.tight_layout()
plt.show()
```



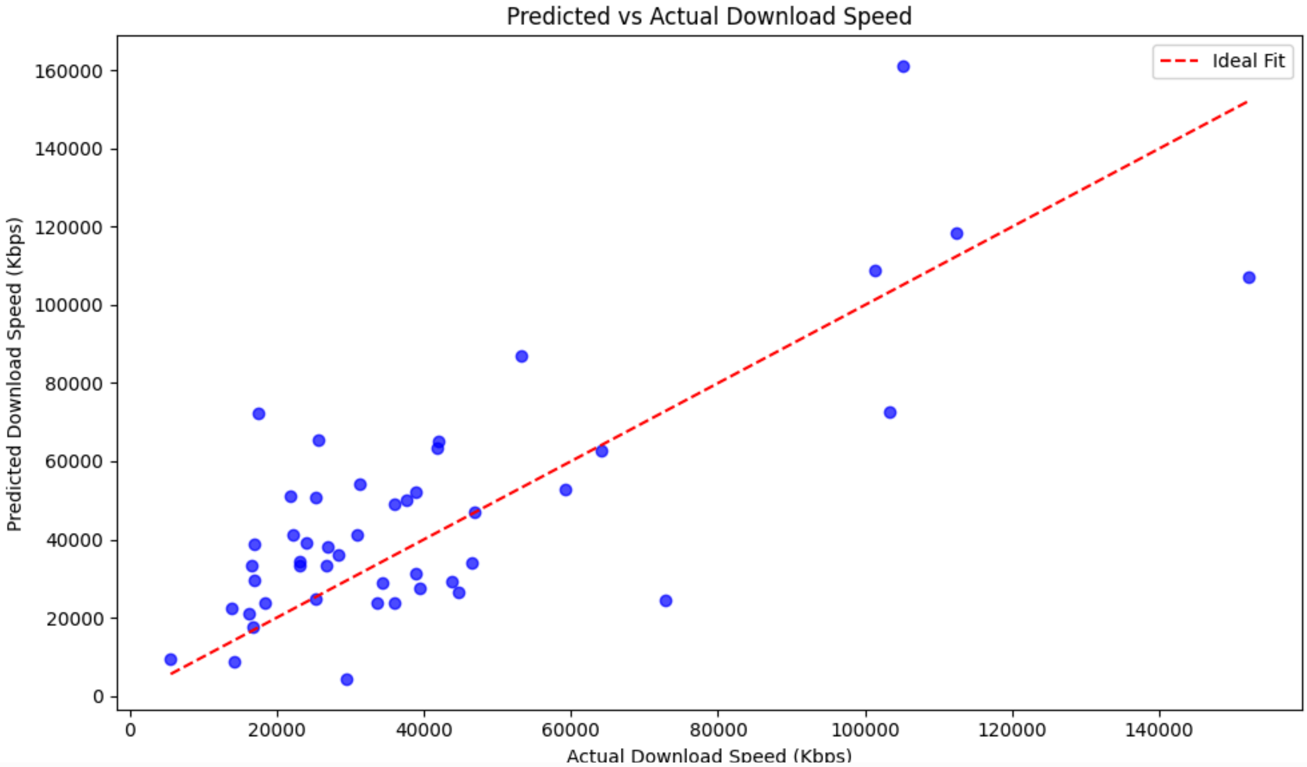
```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler

# Select features and target variable
X = country_metrics[['avg._avg_u_kbps', 'avg_lat_ms']]  # Upload speed and latency as features
y = country_metrics['avg._avg_d_kbps']  # Download speed as target variable

# 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)

# Standardize the features (important for most models)
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

```
from sklearn.ensemble import RandomForestRegressor

# Initialize and train the Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train_scaled, y_train)
```

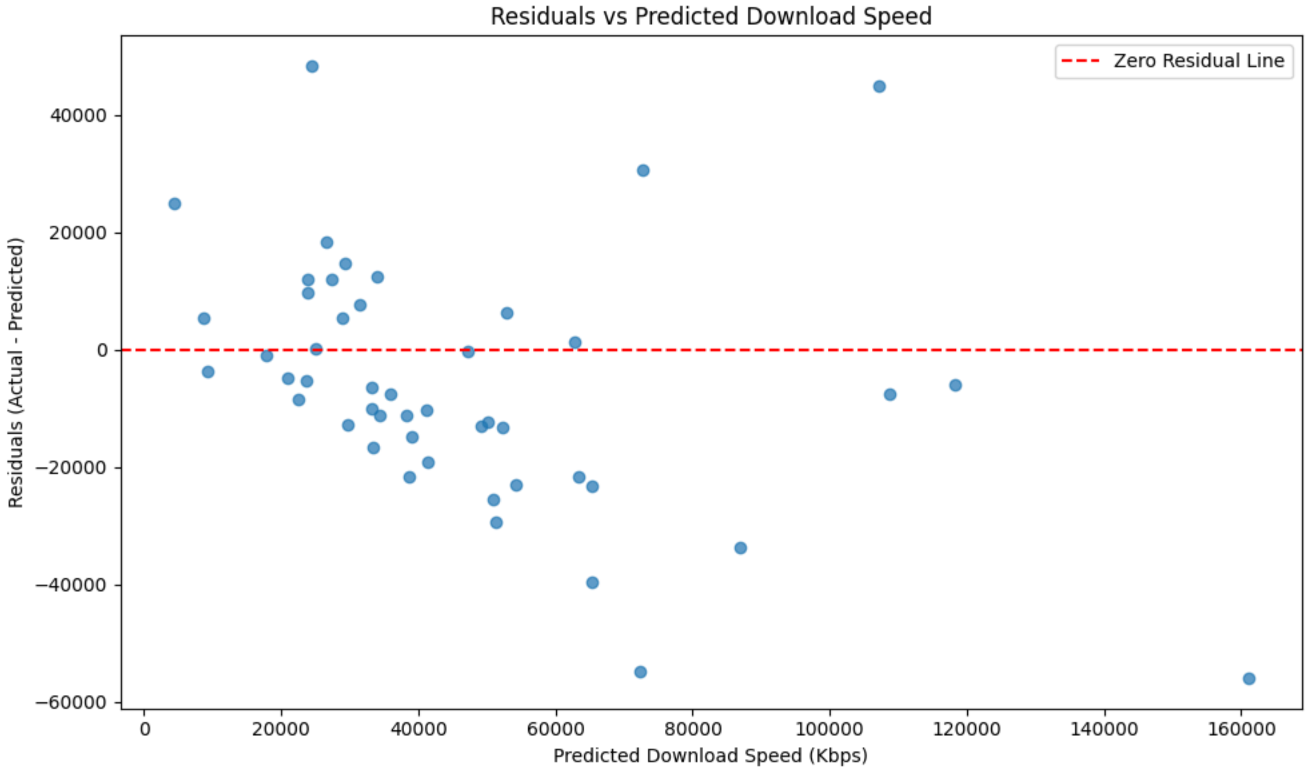
```
▾      RandomForestRegressor      ⓘ ?
RandomForestRegressor(random_state=42)
```

Start coding or generate with AI.

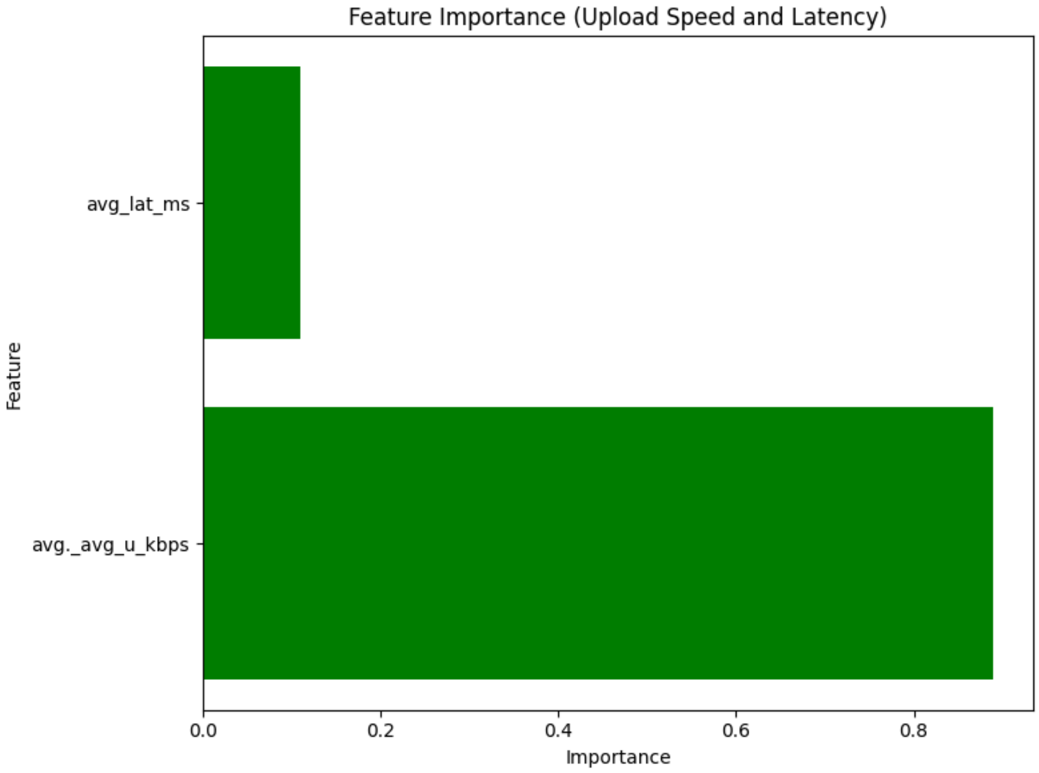
```
plt.figure(figsize=(10, 6))
plt.scatter(y_test, y_pred, color='blue', alpha=0.7)
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)], color='red', linestyle='--', label='Ideal Fit')
plt.title('Predicted vs Actual Download Speed')
plt.xlabel('Actual Download Speed (Kbps)')
plt.ylabel('Predicted Download Speed (Kbps)')
plt.legend()
plt.tight_layout()
plt.tight_layout()
plt.show()
```



```
# Calculate residuals
residuals = y_test - y_pred

# Plot residuals
plt.figure(figsize=(10, 6))
plt.scatter(y_pred, residuals, alpha=0.7)
plt.axhline(y=0, color='red', linestyle='--', label="Zero Residual Line")
plt.title('Residuals vs Predicted Download Speed')
plt.xlabel('Predicted Download Speed (Kbps)')
plt.ylabel('Residuals (Actual - Predicted)')
plt.legend()
plt.tight_layout()
plt.show()
```



```
# Get feature importances from the trained model
importances = model.feature_importances_

# Plot the feature importances
plt.figure(figsize=(8, 6))
plt.barh(X.columns, importances, color='green')
plt.title('Feature Importance (Upload Speed and Latency)')
plt.xlabel('Importance')
plt.ylabel('Feature')
plt.tight_layout()
plt.show()
```



Start coding or generate with AI.