Data Analytics Project

Team Data Pioneers

Harriete

Team Members

Amaka

Harriete

Joy

Tony

Gideon

Outline

- Team Members
- Project focus
- Plan
- Collect
- Process
- Analyze
- Share
- Project Questions

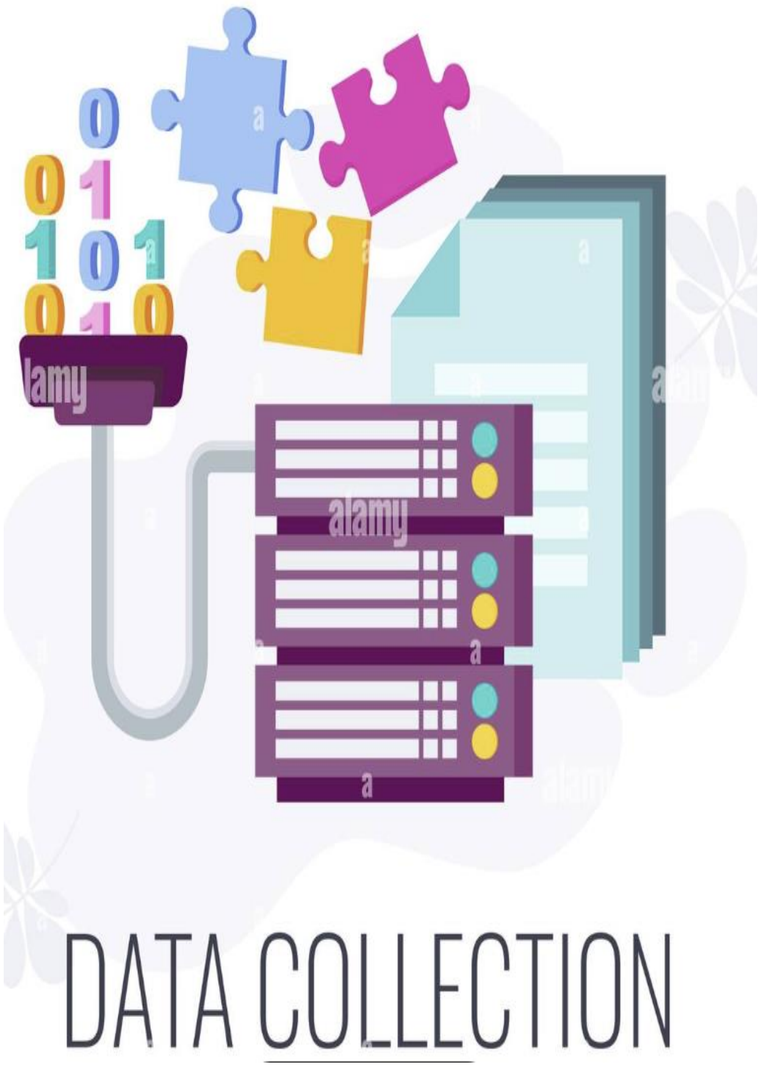
Project focus

Analyse the daily number of webpage visits for several Wikipedia webpages. The dataset includes daily page visit counts for 1,500 Wikipedia pages starting on 2016-01-01 until 2016-12-31.

Dataset source: Kaggle.com

Data Life Cycle

Data Life Cycle



Data Life Cycle

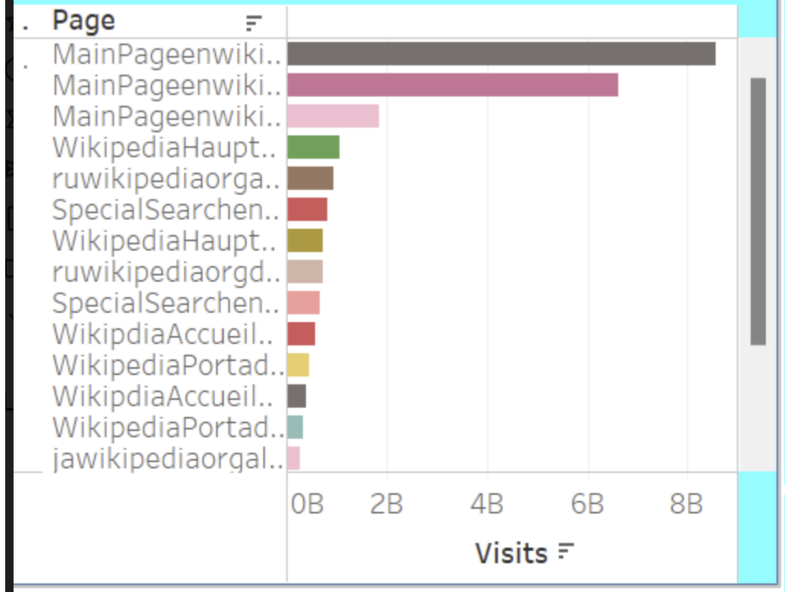
Data Processing

Data Life Cycle

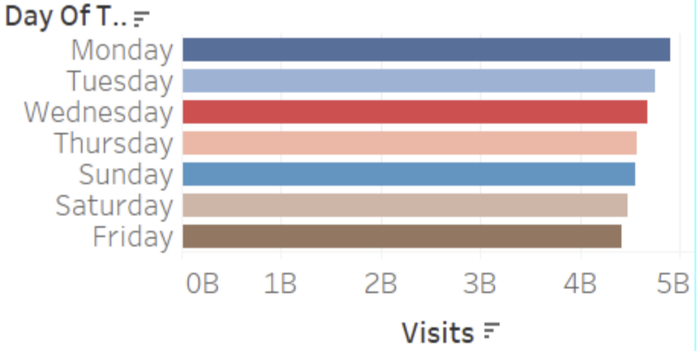
Dashboard

Wikipedia Page Visit Analyses from DataExplorers

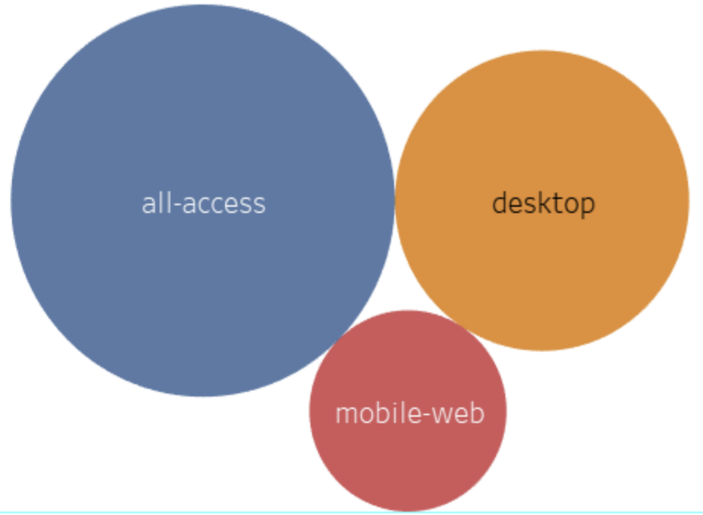
Top 20 Pages Visited



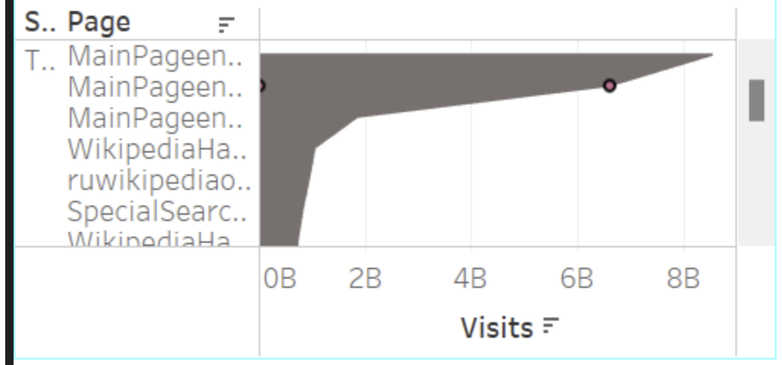
Most Popular Days for Visiting Wikipedia



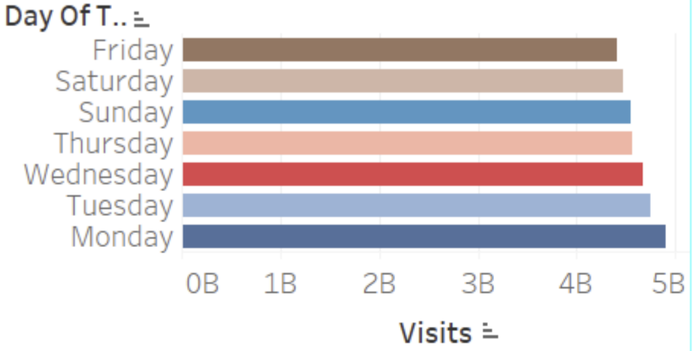
Device Type Used to Visit Wikipedia More Frequently



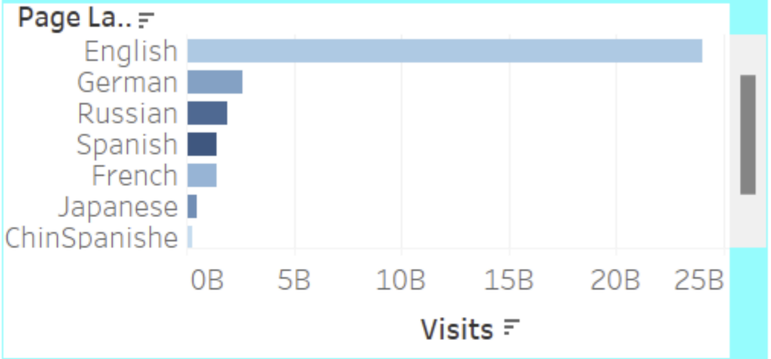
Highest Page Visit



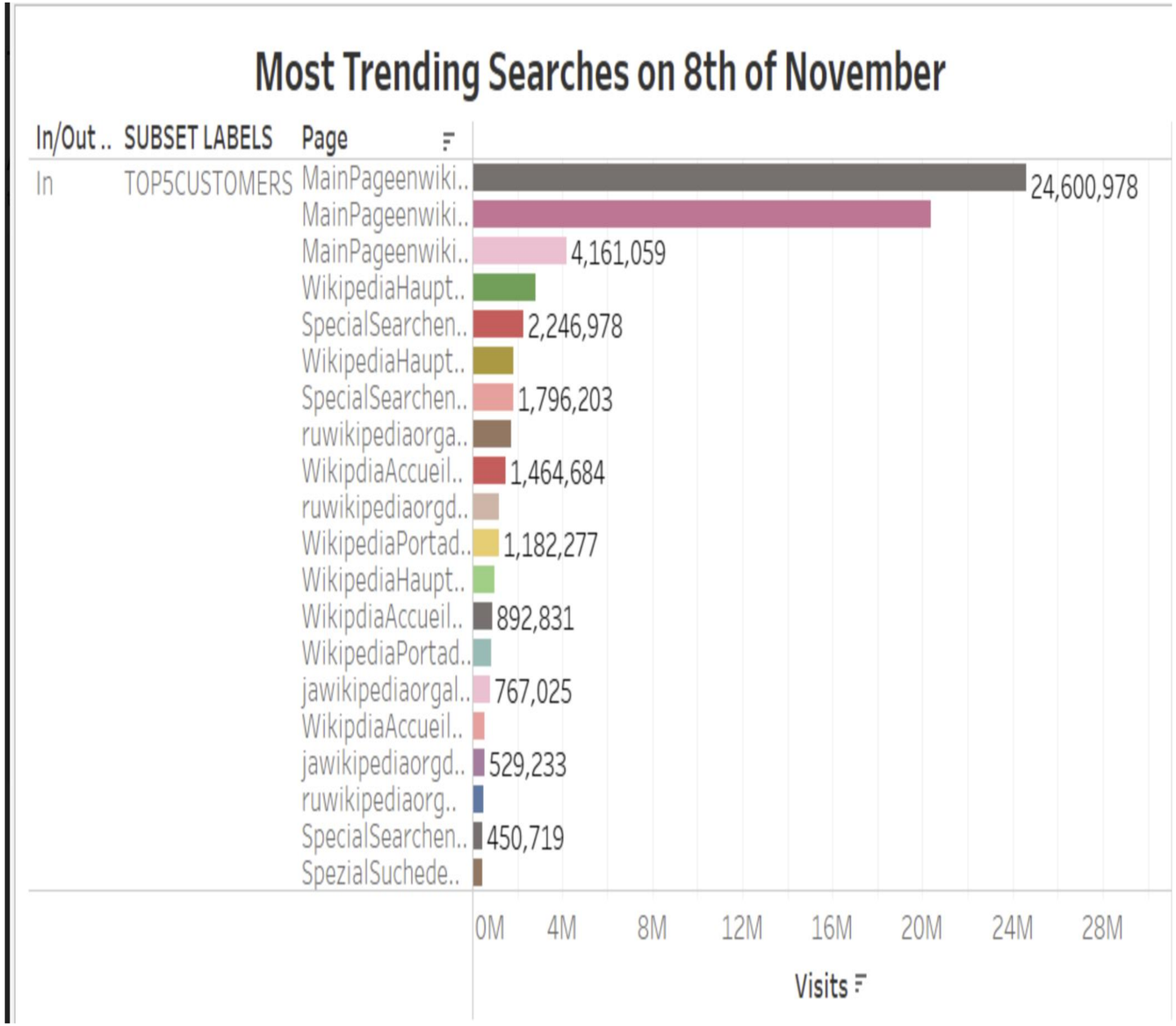
Least Popular Days for Visiting Wikipedia

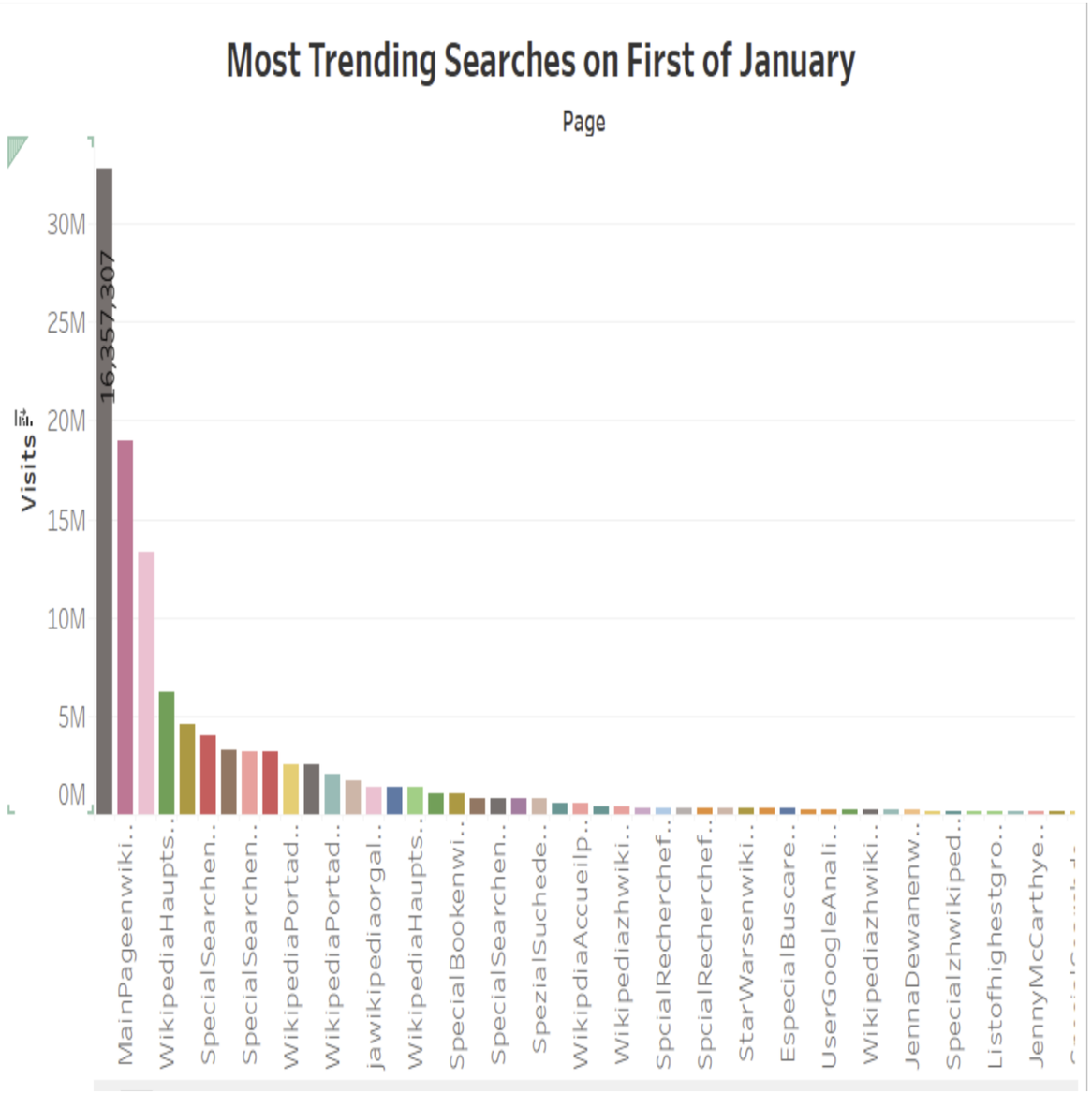


Page Languages

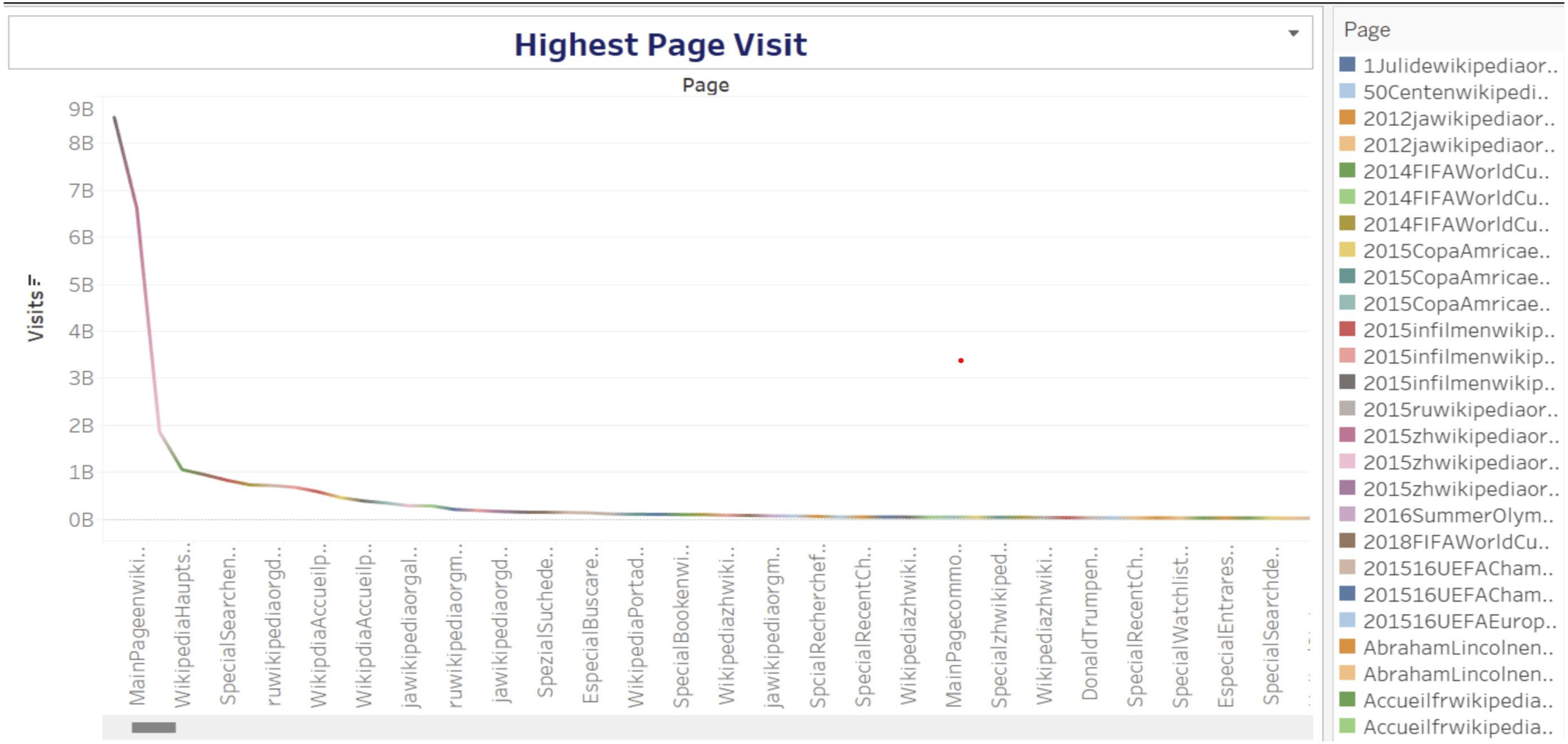


Trending Searches

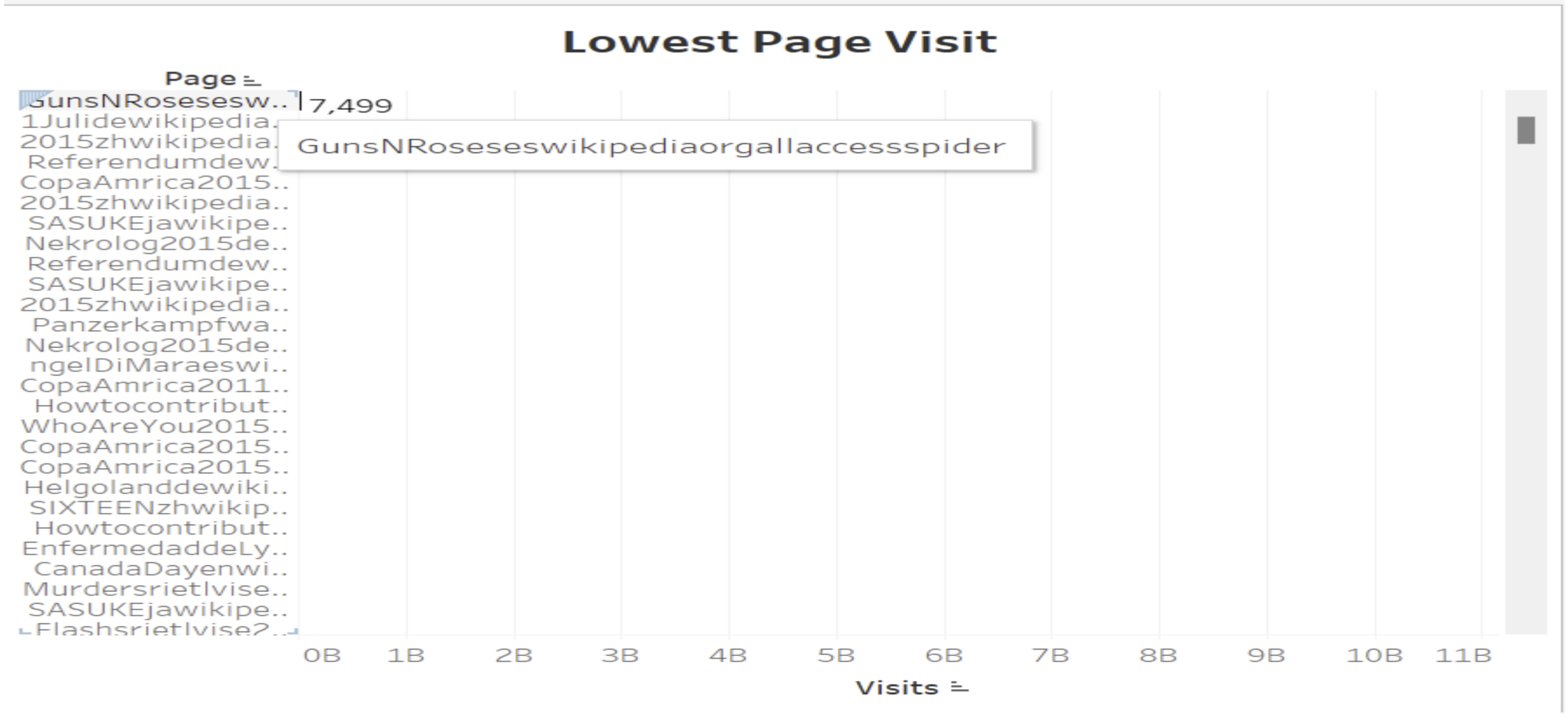




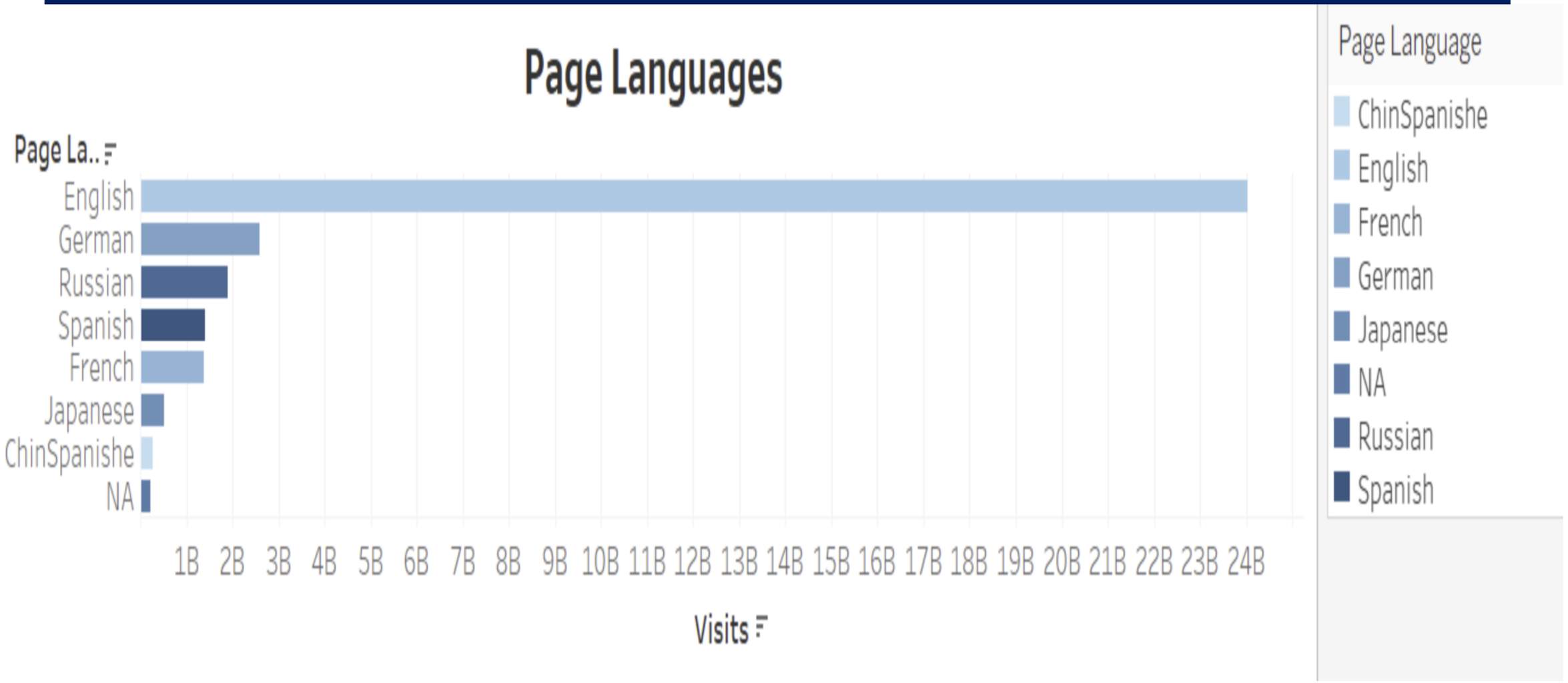
Page with the biggest decline in page visit in 2016



Page with the biggest decline in page visit in 2016

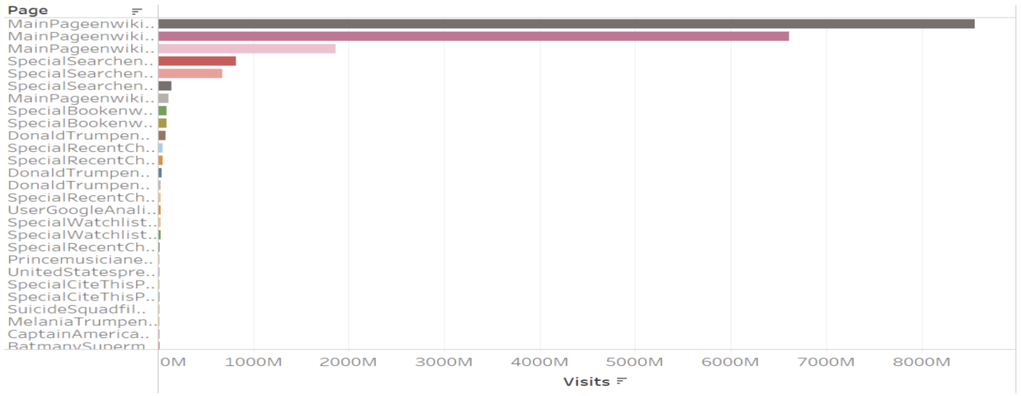


Languages

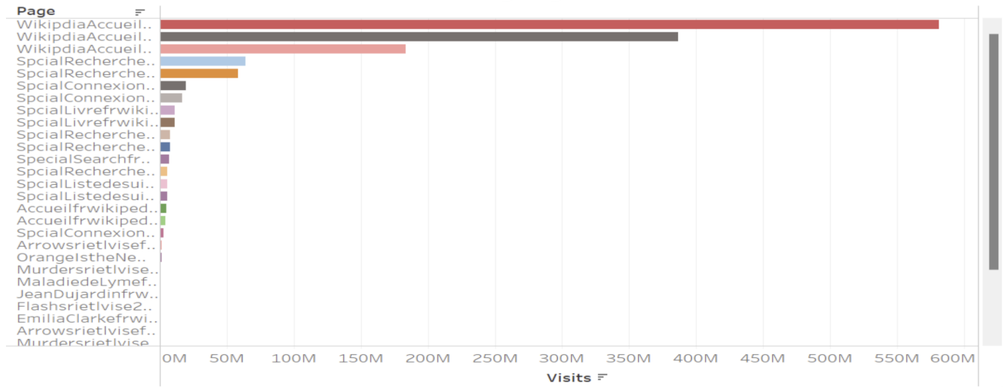

Page Languages
Page La..
English
German
Russian
Spanish
French
Japanese
ChinSpanishe
NA
1B 2B 3B 4B 5B 6B 7B 8B 9B 10B 11B 12B 13B 14B 15B 16B 17B 18B 19B 20B 21B 22B 23B 24B
Visits
Page Language
ChinSpanishe
English
French
German
Japanese
NA
Russian
Spanish

Languages

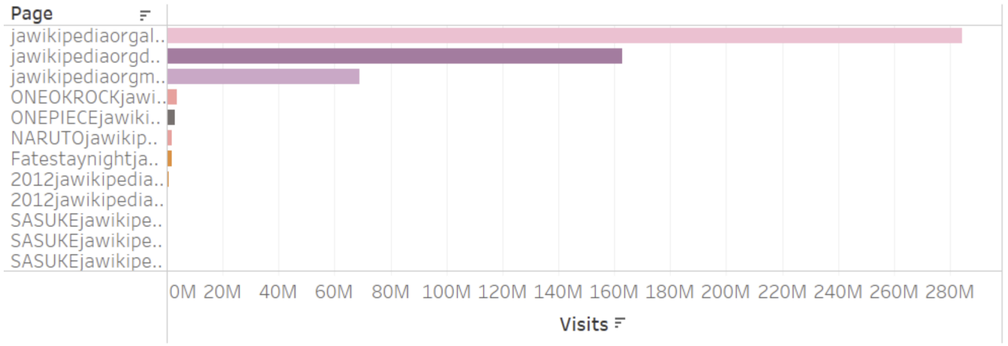
English Language



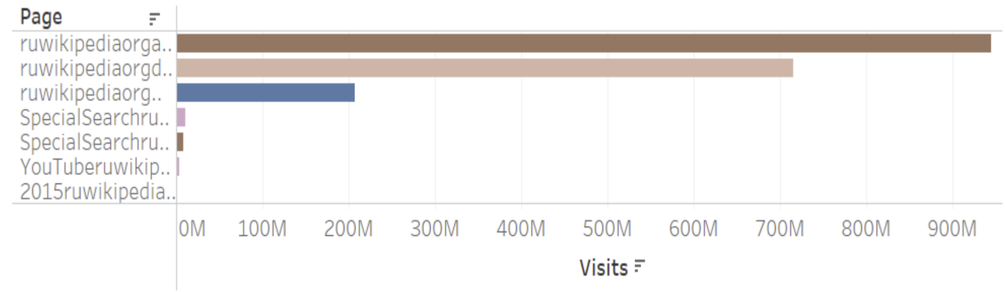
French Language



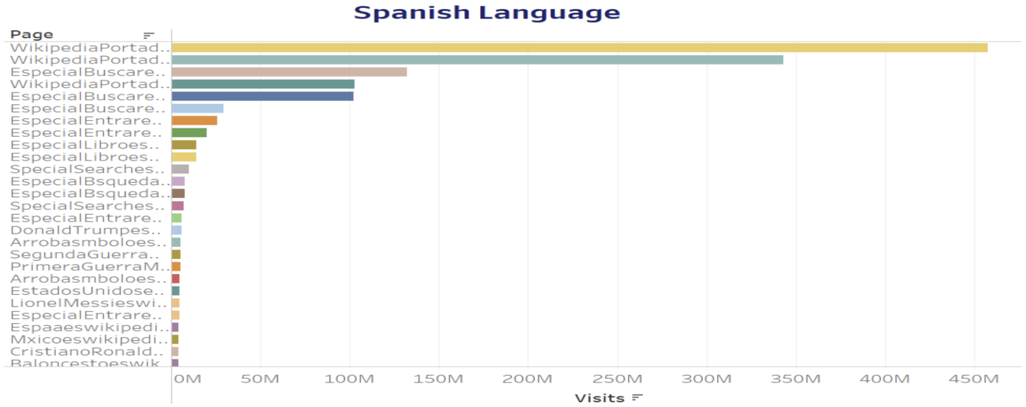
Japanese Language



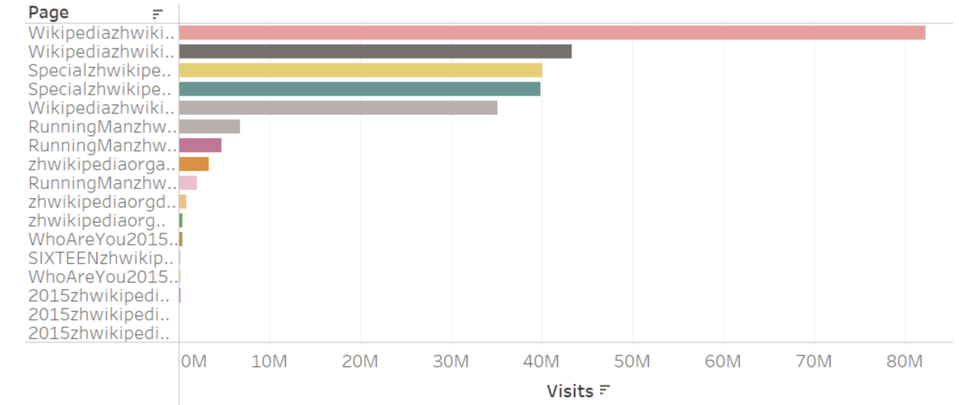
Russian Language



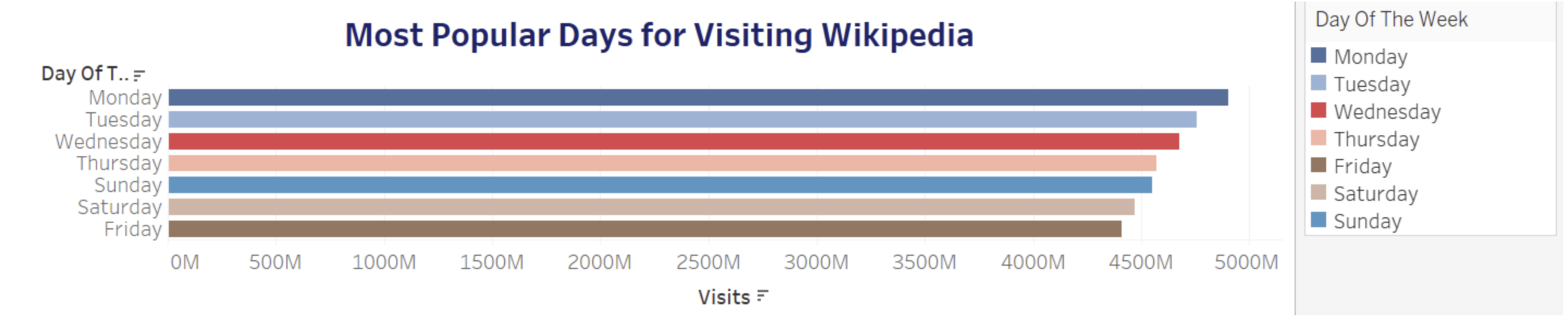
Spanish Language

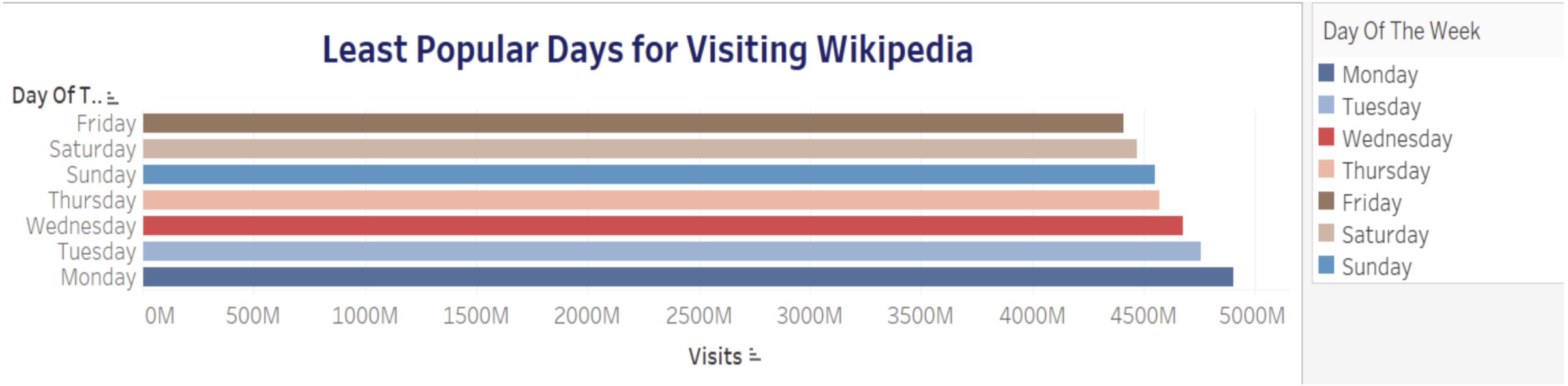


Chinese Language

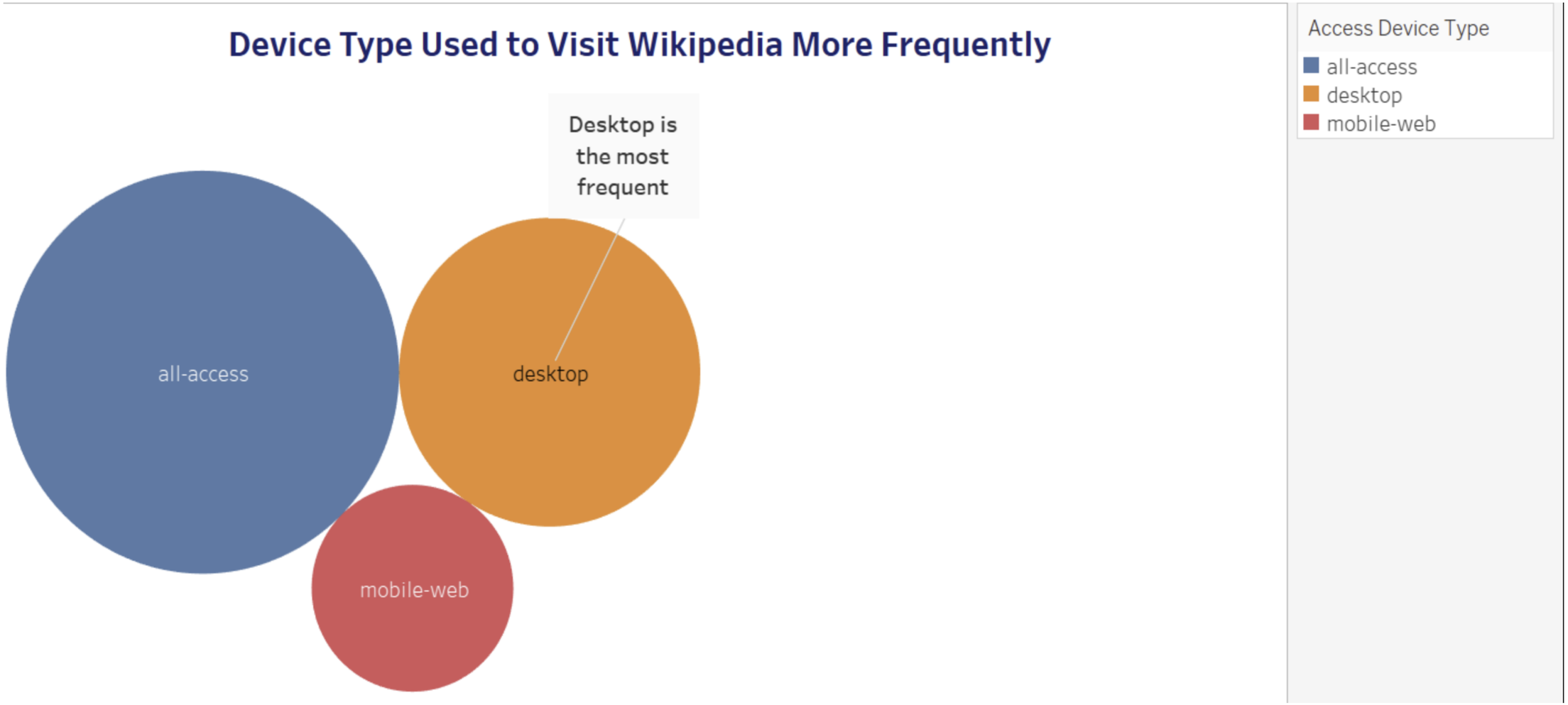


Summary of Most/Least popular for visiting wilkipedia



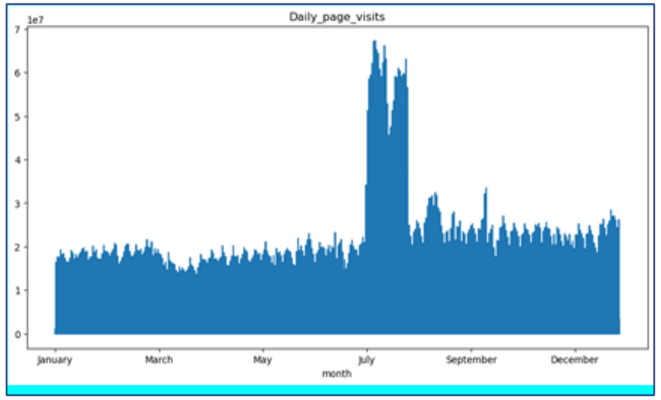


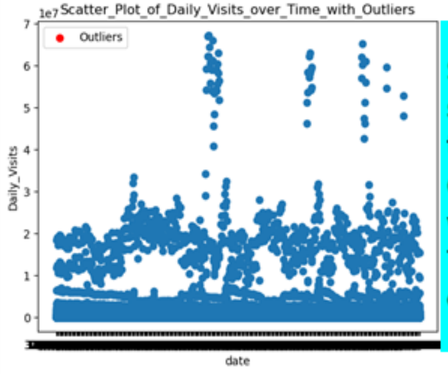
Device Type


Device Type Used to Visit Wikipedia More Frequently
Desktop is the most frequent
all-access
desktop
mobile-web
Access Device Type
all-access
desktop
mobile-web

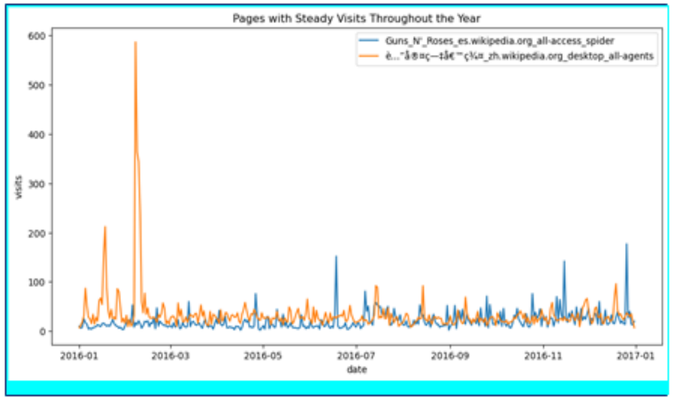
Dashboard

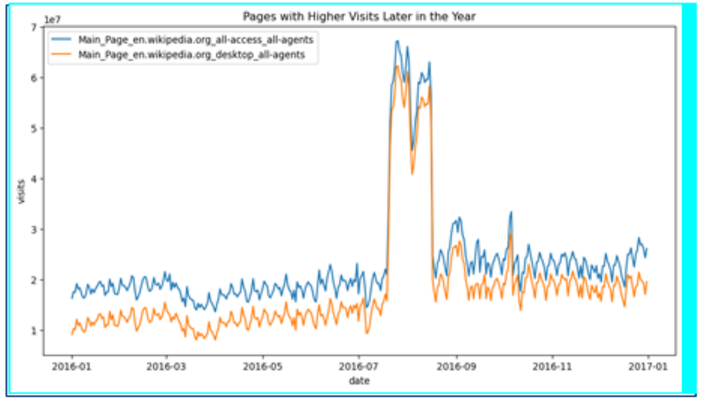
Time Series Plots

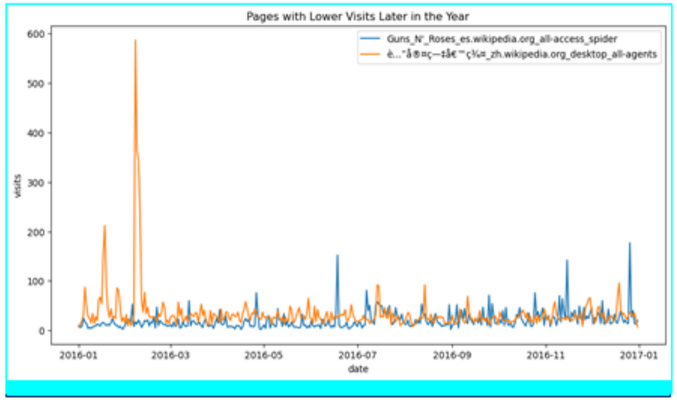




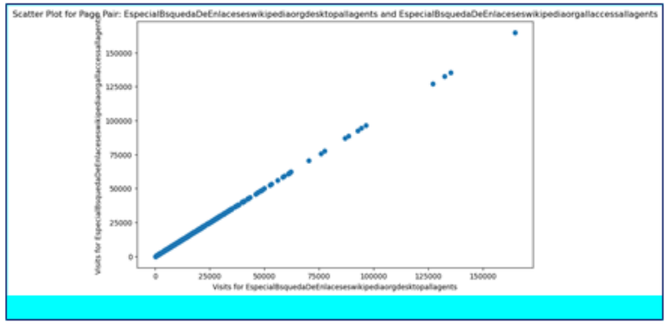
Scatter Plot showing distribution of daily page visits. There are outliers







Time Series Plot of the pages visits showing:
1. steady number of visits
2. Higher number of visits later in the year
3. Lower number of visits later in the year



A Time Series Plot showing positive correlation between the pages: EspecialBsquedaDeEnlaceseswikipediaorgdesktopallagents and EspecialBsquedaDeEnlaceseswikipediaorgallaccessallagents

Codes

```
import pandas as pd
from datetime import datetime
import json
import re


def flatten(df, col_dim, row_dim, value_dim, page_dim, dev_dim, day_dim, mon_dim):
    entries = []
    for col in df.columns:
        print(f'parsing column {col} ...')
        for row in df.index:
            page_match = re.search(r'_([^_]+)\.wikipedia\.org', row)
            dev_match = re.search(r'org\_(.*?)_',row)



            entry = {
                col_dim: col,
                row_dim: row,
                value_dim: df[col].loc[row],
                page_dim: re.search(r'_([^_]+)\.wikipedia\.org', row).group(1) if page_match else 'NA',
                dev_dim: re.search(r'org\_(.*?)_',row).group(1) if dev_match else None,
                day_dim: pd.to_datetime(col, format='%Y/%m/%d').day_name(),
               mon_dim: pd.to_datetime(col, format='%Y/%m/%d').strftime('%m') #pd.to_datetime(col, format='%Y/%m/%d').month
```

code

```
                }
                entries.append(entry)
    return pd.DataFrame(entries)

if __name__ == '__main__':
    config = json.load(open('flatten_config.json', 'r'))
    df = pd.read_excel(config['file_name'], index_col=0)

    df_flat = flatten(df=df,
                      col_dim=config['column_dimension'],
                      row_dim=config['row_dimension'],
                      value_dim=config['value_dimension'],
                      page_dim=config['page_lang_dim'],
                      dev_dim=config['dev_type_dim'],
                      day_dim=config['week_day_dim'],
                      mon_dim=config['month_dim']
                  )
# Define a custom cleaning function
def clean_text(page):
    cleaned_text = re.sub('[^A-Za-z0-9 ]+', '', page)
    return cleaned_text

# apply the cleaning function to the 'page' and create a new cleaned column 'page'
df_flat['page'] = df_flat['page'].apply(clean_text)

output_name = config["file_name"].split('.')[0] + '_flat.xlsx'
df_flat.to_excel(output_name)
```

code

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load your dataset
df = pd.read_excel('C:\\Users\\davel\\OneDrive\\Week\\wikipedia_dataset_flat2.xlsx')

# Display basic information about the dataset
print(df.info())

# Display summary statistics of numeric columns
print(df.describe())

# Distribution of visits
plt.figure(figsize=(10, 6))
sns.histplot(df['visits'], bins=30, kde=True)
plt.title('Distribution of Page Visits')
plt.xlabel('Visits')
plt.ylabel('Frequency')
plt.show()


from datetime import datetime
import statsmodels.api as sm
from statsmodels.tsa.seasonal import seasonal_decompose

# Convert the 'date' column to datetime format
df['date'] = pd.to_datetime(df['date'])

# Set 'date' as the index
df.set_index('date', inplace=True)
```

code

```
# Resample data to daily frequency if not already
# df = df.resample('D').sum()  # Uncomment if needed

# Time series decomposition (trend, seasonal, residual)
decomposition = seasonal_decompose(df['visits'], model='additive')
trend = decomposition.trend
seasonal = decomposition.seasonal
residual = decomposition.resid

# Plot the decomposition
plt.figure(figsize=(12, 8))
plt.subplot(411)
plt.plot(df['visits'], label='Original')
plt.legend(loc='upper left')
plt.subplot(412)

# Criterion 1: Relatively steady number of visits throughout the year
# Identify two pages that meet the criterion
steady_pages = df.groupby('page')['visits'].std().sort_values().index[:2]
for page in steady_pages:
    page_data = df[df['page'] == page]
    plt.figure(figsize=(12, 6))
    plt.plot(page_data['date'], page_data['visits'], label=page)
    plt.title(f'Time Series Plot for Page: {page}')
    plt.xlabel('Date')
    plt.ylabel('Visits')
    plt.legend()
    plt.show()
# Criterion 2: Significantly higher number of visits later in the year compared to
earlier in the year
# Identify two pages that meet the criterion
high_visit_pages = df.groupby('page')['visits'].sum().sort_values().index[-2:]
for page in high_visit_pages:
    page_data = df[df['page'] == page]
```

code

```
    plt.figure(figsize=(12, 6))
    plt.plot(page_data['date'], page_data['visits'], label=page)
    plt.title(f'Time Series Plot for Page: {page}')
    plt.xlabel('Date')
    plt.ylabel('Visits')
    plt.legend()
    plt.show()

# For Criterion 2
high_visits_pages = df.groupby('page')['visits'].mean().sort_values(ascending=False).index[:2]

# For Criterion 3
low_visits_pages = df.groupby('page')['visits'].mean().sort_values().index[:2]

# Plot pages with higher visits later in the year
plt.figure(figsize=(12, 6))
for page in high_visits_pages:
    page_data = df[df['page'] == page]
    sns.lineplot(x='date', y='visits', data=page_data, label=page)
plt.title('Pages with Higher Visits Later in the Year')
plt.xlabel('date')
plt.ylabel('visits')
plt.legend()
plt.show()

# Plot pages with lower visits later in the year
plt.figure(figsize=(12, 6))
for page in low_visits_pages:
    page_data = df[df['page'] == page]
    sns.lineplot(x='date', y='visits', data=page_data, label=page)

plt.title('Pages with Lower Visits Later in the Year')
plt.xlabel('date')
plt.ylabel('visits')
plt.legend()
```

code

```
plt.show()
corr_matrix = df.pivot_table(index='date', columns='page', values='visits',
aggfunc='sum').corr()

# Find page pairs with high correlation
high_corr_pairs = []
for page1 in corr_matrix.columns:
    for page2 in corr_matrix.columns:
        if page1 != page2 and abs(corr_matrix.loc[page1, page2]) > 0.8:
            high_corr_pairs.append((page1, page2))

# Plot scatter plots for high correlation page pairs
for pair in high_corr_pairs:
    page1, page2 = pair
    plt.figure(figsize=(8, 4))
    sns.scatterplot(x=df[df['page'] == page1]['visits'], y=df[df['page'] ==
page2]['visits'])
    plt.title(f'Scatter Plot for {page1} and {page2}')
    plt.xlabel(f'visits - {page1}')
    plt.ylabel(f'visits - {page2}')
    plt.show()
```

```
Select *
from [dbo].[Copy of wikipedia_dataset_flat2csv];
```

```
SELECT page, visits
FROM [dbo].[Copy of wikipedia_dataset_flat2csv]
```

code

```
WHERE date = '2016-11-08'
ORDER BY visits DESC

#Page with the biggest decline in page visits during 2016:

SELECT page, MIN(visits) as min_visits
FROM [dbo].[Copy of wikipedia_dataset_flat2csv]
WHERE YEAR(date) = 2016
GROUP BY page
ORDER BY min_visits;



#Page with the biggest increase in page visits during 2016:

SELECT page, MAX(visits) as max_visits
FROM [dbo].[Copy of wikipedia_dataset_flat2csv]
WHERE YEAR(date) = 2016
```

code

```
GROUP BY page
ORDER BY max_visits DESC;


# Number of languages represented in the dataset and proportion of pages for each language:
SELECT
    page_language,
    COUNT(DISTINCT page) as num_pages,
    CAST(COUNT(DISTINCT page) AS bigint) / COUNT(DISTINCT page_language) as proportion
FROM
    [dbo].[Copy of wikipedia_dataset_flat2csv]
GROUP BY
    page_language;
```

code

```
SELECT
    day_of_the_week,
    SUM(CAST(visits AS bigint)) as total_visits
FROM
  [dbo].[Copy of wikipedia_dataset_flat2csv]
GROUP BY
    day_of_the_week
ORDER BY
    total_visits ASC;

SELECT
    access_device_type,
    SUM(CAST(visits AS bigint)) as total_visits
FROM
```

code

```
 [dbo].[Copy of wikipedia_dataset_flat2csv]
GROUP BY
    access_device_type
ORDER BY
    total_visits DESC;

SELECT page, COALESCE(MIN(visits), 0) as min_visits
FROM [dbo].[Copy of wikipedia_dataset_flat2csv]
WHERE YEAR(date) = 2016
GROUP BY page
ORDER BY min_visits;
```

Thank you!