```
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/web-server-access-logs/access.log
/kaggle/input/web-server-access-logs/client_hostname.csv
```

Data pre-processing

The log file comprises 3.3GB of web server logs extracted from zanbil.ir, an Iranian ecommerce platform, offering a comprehensive view of user interactions, crawler activities, and business trends. This log file, compiled by Zaker and Farzin in 2019, is available via Harvard Dataverse for research and analytical purposes.

Loading the log file into a dataframe

I extracted relevant information such as client IP, user ID, timestamp, HTTP method, request, status code, size, referer, and user agent from each log line.

```
import pandas as pd
import re

# Define the log file path
log_file_path = '/kaggle/input/web-server-access-logs/access.log'

# Define the regex pattern to extract information from log lines
regex_pattern = r'^(?P<client>\S+) \S+ (?P<userid>\S+) \[(?
P<datetime>[\w:/]+\s[+\-]\d{4})\] "(?P<method>[A-Z]+) (?P<request>[^
"]+)? HTTP/[0-9.]+" (?P<status>[0-9]{3}) (?P<size>[0-9]+|-) "(?
P<referer>[^"]*)" "(?P<user_agent>[^"]*)"'

# Define the column names
columns = ['client', 'userid', 'datetime', 'method', 'request',
'status', 'size', 'referer', 'user_agent']

# Read the first 10000 rows of the log file into a list of
dictionaries using regex pattern matching
log_data = []
with open(log_file_path, 'r') as file:
    for i, line in enumerate(file):
        if i >= 10000:
            break
        match = re.match(regex_pattern, line)
        if match:
            log_data.append({
                'client': match.group('client'),
                'userid': match.group('userid'),
                'datetime': match.group('datetime'),
                'method': match.group('method'),
                'request': match.group('request'),
                'status': match.group('status'),
                'size': match.group('size'),
                'referer': match.group('referer'),
                'user_agent': match.group('user_agent')
            })
        else:
            print("Error: Line does not match regex pattern:", line)

# Create DataFrame from the list of dictionaries
logs_df = pd.DataFrame(log_data, columns=columns)

# Diplaying the first 5 rows of the dataframe
logs_df.head()
```

```
          client userid                    datetime method  \
0   54.36.149.41      -  22/Jan/2019:03:56:14 +0330    GET   
1    31.56.96.51      -  22/Jan/2019:03:56:16 +0330    GET   
2    31.56.96.51      -  22/Jan/2019:03:56:16 +0330    GET   
3  40.77.167.129      -  22/Jan/2019:03:56:17 +0330    GET   
4    91.99.72.15      -  22/Jan/2019:03:56:17 +0330    GET   

                                             request status   size  \
0  /filter/27|13%20%D9%85%DA%AF%D8%A7%D9%BE%DB%8C...    200  30577   
1                /image/60844/productModel/200x200    200   5667   
2                /image/61474/productModel/200x200    200   5379   
3                /image/14925/productModel/100x100    200   1696   
4  /product/31893/62100/%D8%B3%D8%B4%D9%88%D8%A7%...    200  41483   
```

```
                               referer  \
0                                    -
1  https://www.zanbil.ir/m/filter/b113
2  https://www.zanbil.ir/m/filter/b113
3                                    -
4                                    -

                                          user_agent
0  Mozilla/5.0 (compatible; AhrefsBot/6.1; +http:...
1  Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build...
2  Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build...
3  Mozilla/5.0 (compatible; bingbot/2.0; +http://...
4  Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:16...
```

Understanding and processing the dataset

```
# Checking the overview of the dataframe
logs_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype 
---  ------      --------------  ----- 
 0   client      10000 non-null  object
 1   userid      10000 non-null  object
 2   datetime    10000 non-null  object
 3   method      10000 non-null  object
 4   request     10000 non-null  object
 5   status      10000 non-null  object
 6   size        10000 non-null  object
 7   referer     10000 non-null  object
 8   user_agent  10000 non-null  object
dtypes: object(9)
memory usage: 703.2+ KB

from datetime import datetime
import pytz

# Function to parse the datetime (from the class session practice
exercise)
def parse_datetime(x):
    '''
    Parses datetime with timezone formatted as:
        `[day/month/year:hour:minute:second zone]`

    Example:
        `>>> parse_datetime('13/Nov/2015:11:45:42 +0000')`
        `datetime.datetime(2015, 11, 3, 11, 45, 4, tzinfo=<UTC>)`
    Due to problems parsing the timezone (`%z`) with
`datetime.strptime`, the
    timezone will be obtained using the `pytz` library.
    '''
    try:
        dt = datetime.strptime(x[1:-7], '%d/%b/%Y:%H:%M:%S')
        dt_tz = int(x[-6:-3])*60+int(x[-3:-1])
        return dt.replace(tzinfo=pytz.FixedOffset(dt_tz))
    except ValueError:
        return '-'

logs_df['status'] = logs_df['status'].astype(int)
logs_df['size'] = logs_df['size'].astype(int)
logs_df['datetime'] = logs_df['datetime'].apply(parse_datetime)

logs_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 10000 entries, 0 to 9999
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   client      10000 non-null  object
 1   userid      10000 non-null  object
 2   datetime    10000 non-null  datetime64[ns, pytz.FixedOffset(33)]
 3   method      10000 non-null  object
 4   request     10000 non-null  object
 5   status      10000 non-null  int64
 6   size        10000 non-null  int64
 7   referer     10000 non-null  object
 8   user_agent  10000 non-null  object
dtypes: datetime64[ns, pytz.FixedOffset(33)](1), int64(2), object(6)
memory usage: 703.2+ KB
```

Dropping the userid column

This because it has one unique value which is just a hyphen

```
users = logs_df['userid'].unique()
print(users)

['-']

logs_df.drop(columns=['userid'], inplace=True)
```

Dropping duplicates

There were duplicates which are not adding value to the analysis.

```
# Count duplicates in the dataframe
duplicate_count = logs_df.duplicated().sum()

# Display the count of duplicates
print("Number of duplicates:", duplicate_count)

Number of duplicates: 49

# Drop the duplicates
logs_df = logs_df.drop_duplicates()
```

The sample of processed data

```
# Diplaying the first 5 rows of the dataframe
logs_df.head()

          client                  datetime method  \
0   54.36.149.41 2019-01-02 03:56:01+00:33    GET
1    31.56.96.51 2019-01-02 03:56:01+00:33    GET
2    31.56.96.51 2019-01-02 03:56:01+00:33    GET
3  40.77.167.129 2019-01-02 03:56:01+00:33    GET
4    91.99.72.15 2019-01-02 03:56:01+00:33    GET

                                             request  status   size  \
0  /filter/27|13%20%D9%85%DA%AF%D8%A7%D9%BE%DB%8C...     200  30577
1                /image/60844/productModel/200x200     200   5667
2                /image/61474/productModel/200x200     200   5379
3                /image/14925/productModel/100x100     200   1696
4  /product/31893/62100/%D8%B3%D8%B4%D9%88%D8%A7%...     200  41483

                                referer  \
0                                     -
1  https://www.zanbil.ir/m/filter/b113
2  https://www.zanbil.ir/m/filter/b113
3                                     -
4                                     -

                                          user_agent
0  Mozilla/5.0 (compatible; AhrefsBot/6.1; +http:...
1  Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build...
2  Mozilla/5.0 (Linux; Android 6.0; ALE-L21 Build...
3  Mozilla/5.0 (compatible; bingbot/2.0; +http://...
4  Mozilla/5.0 (Windows NT 6.2; Win64; x64; rv:16...
```

Answering the prompts

Q1. 10 people who visited the site frequently

To get the most frequent users, we had to do the user identication by grouping according the client ip address (client) and the user agent (user_agent). Then sort by the count.

```
# Group by client and user_agent, count occurrences, and sort in
descending order
frequent_visitors = logs_df.groupby(['client',
'user_agent']).size().reset_index(name='count').sort_values(by='count'
, ascending=False)

# Select the top 10 frequent visitors
top_10 = frequent_visitors.head(10)

index = 0
# Display the top 10 frequent visitors
for i, row in top_10.iterrows():
    print(f"{index + 1}. Client: {row['client']}, User Agent:
{row['user_agent']}, Count: {row['count']}\n")
    index += 1
```

```
1. Client: 66.249.66.194, User Agent: Mozilla/5.0 (Linux; Android
6.0.1; Nexus 5X Build/MMB29P) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/41.0.2272.96 Mobile Safari/537.36 (compatible; Googlebot/2.1;
+http://www.google.com/bot.html), Count: 778

2. Client: 66.249.66.91, User Agent: Mozilla/5.0 (compatible;
Googlebot/2.1; +http://www.google.com/bot.html), Count: 739

3. Client: 130.185.74.243, User Agent: Mozilla/5.0 (Windows NT 6.1;
rv:42.0) Gecko/20100101 Firefox/42.0, Count: 660

4. Client: 66.249.66.194, User Agent: Mozilla/5.0 (compatible;
Googlebot/2.1; +http://www.google.com/bot.html), Count: 558

5. Client: 5.211.97.39, User Agent: Mozilla/5.0 (iPhone; CPU iPhone OS
10_3_2 like Mac OS X) AppleWebKit/603.2.4 (KHTML, like Gecko)
Version/10.0 Mobile/14F89 Safari/602.1, Count: 474

6. Client: 207.46.13.136, User Agent: Mozilla/5.0 (compatible;
bingbot/2.0; +http://www.bing.com/bingbot.htm), Count: 416

7. Client: 194.94.127.7, User Agent: Mozilla/5.0 (Windows NT 6.1;
Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/65.0.3325.181 Safari/537.36\x09Chrome 65.0, Count: 225

8. Client: 23.101.169.3, User Agent: Mozilla/5.0 (compatible; MSIE
9.0; Windows NT 6.1; Trident/5.0;  Trident/5.0), Count: 204
```

```
9. Client: 5.121.43.23, User Agent: Mozilla/5.0 (Linux; Android 7.0;
FRD-L09) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/70.0.3538.80
Mobile Safari/537.36, Count: 165

10. Client: 40.77.167.170, User Agent: Mozilla/5.0 (compatible;
bingbot/2.0; +http://www.bing.com/bingbot.htm), Count: 164
```

Q2. Sessions and the page views per each session

Basic sessionization (skip)

```
# Group by client and user_agent to identify sessions and count page
views per session
sessions = logs_df.groupby(['client', 'user_agent'])

# Initialize empty lists to store session information
session_info = []

# Iterate over each session
for (client, user_agent), session_data in sessions:
    # Extract timestamps and page views for the session
    timestamps = session_data['datetime'].tolist()
    pages = session_data['request'].tolist()

    # Store session information in a tuple
    session_info.append((client, user_agent, timestamps, pages))

session_info[1:4]

[('104.194.24.33',
  'Mozilla/5.0 (Linux; Android 8.0.0; SM-G955F) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.99 Mobile Safari/537.36',
  [Timestamp('2019-01-02 03:57:00+0033', tz='pytz.FixedOffset(33)')],
  ['/amp-helper-frame.html?appId=a624a1c1-0c93-466a-a546-
e146710f97e6&parentOrigin=https://www-zanbil-ir.cdn.ampproject.org']),
 ('104.194.24.54',
  'Dalvik/2.1.0 (Linux; U; Android 6.0.1; SM-G900H Build/MMB29K)',
  [Timestamp('2019-01-02 04:24:00+0033', tz='pytz.FixedOffset(33)'),
   Timestamp('2019-01-02 04:26:04+0033', tz='pytz.FixedOffset(33)')],
  ['/image/33888?name=model-b2048u-1-.jpg&wh=200x200',
   '/image/11947?name=11947-1-fw.jpg&wh=200x200']),
 ('104.194.25.207',
  'Dalvik/2.1.0 (Linux; U; Android 5.0.2; P01V Build/LRX22G)',
  [Timestamp('2019-01-02 04:06:04+0033', tz='pytz.FixedOffset(33)'),
   Timestamp('2019-01-02 04:06:05+0033', tz='pytz.FixedOffset(33)'),
   Timestamp('2019-01-02 04:06:05+0033', tz='pytz.FixedOffset(33)')],
  ['/image/33888?name=model-b2048u-1-.jpg&wh=200x200',
   '/image/11947?name=11947-1-fw.jpg&wh=200x200',
   '/image/11926?name=sm812aaa.jpg&wh=200x200'])]
```

```python
# Display at least five sessions and their page views per session
for i, (client, user_agent, timestamps, pages) in
enumerate(session_info[:5], start=1):
    print(f"Session {i} - Client: {client}, User Agent: {user_agent}")
    for timestamp, page in zip(timestamps, pages):
        print(f"    Timestamp: {timestamp}, Page: {page}")
    print()
```

```
Session 1 - Client: 104.156.210.196, User Agent: Dalvik/2.1.0 (Linux;
U; Android 8.0.0; SM-A720F Build/R16NW)
    Timestamp: 2019-01-02 04:20:00+00:33, Page: /image/32768?
name=24xs450-33.jpg&wh=200x200

Session 2 - Client: 104.194.24.33, User Agent: Mozilla/5.0 (Linux;
Android 8.0.0; SM-G955F) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/71.0.3578.99 Mobile Safari/537.36
    Timestamp: 2019-01-02 03:57:00+00:33, Page: /amp-helper-
frame.html?appId=a624a1c1-0c93-466a-a546-
e146710f97e6&parentOrigin=https://www-zanbil-ir.cdn.ampproject.org

Session 3 - Client: 104.194.24.54, User Agent: Dalvik/2.1.0 (Linux; U;
Android 6.0.1; SM-G900H Build/MMB29K)
    Timestamp: 2019-01-02 04:24:00+00:33, Page: /image/33888?
name=model-b2048u-1-.jpg&wh=200x200
    Timestamp: 2019-01-02 04:26:04+00:33, Page: /image/11947?
name=11947-1-fw.jpg&wh=200x200

Session 4 - Client: 104.194.25.207, User Agent: Dalvik/2.1.0 (Linux;
U; Android 5.0.2; P01V Build/LRX22G)
    Timestamp: 2019-01-02 04:06:04+00:33, Page: /image/33888?
name=model-b2048u-1-.jpg&wh=200x200
    Timestamp: 2019-01-02 04:06:05+00:33, Page: /image/11947?
name=11947-1-fw.jpg&wh=200x200
    Timestamp: 2019-01-02 04:06:05+00:33, Page: /image/11926?
name=sm812aaa.jpg&wh=200x200

Session 5 - Client: 104.248.138.218, User Agent: Mozilla/5.0 (iPhone;
CPU iPhone OS 12_1_2 like Mac OS X) AppleWebKit/605.1.15 (KHTML, like
Gecko) Version/12.0 Mobile/15E148 Safari/604.1
    Timestamp: 2019-01-02 04:35:01+00:33, Page: /m/browse/sewing-
machine/%DA%86%D8%B1%D8%AE-%D8%AE%DB%8C%D8%A7%D8%B7%DB%8C
    Timestamp: 2019-01-02 04:35:01+00:33, Page: /favicon.ico
    Timestamp: 2019-01-02 04:35:01+00:33, Page:
/static/images/guarantees/goodShopping.png
    Timestamp: 2019-01-02 04:35:02+00:33, Page:
/static/css/font/wyekan/font.woff
    Timestamp: 2019-01-02 04:35:02+00:33, Page:
```

```
/static/images/guarantees/bestPrice.png
    Timestamp: 2019-01-02 04:35:02+00:33, Page:
/static/images/guarantees/warranty.png
    Timestamp: 2019-01-02 04:35:02+00:33, Page:
/static/images/guarantees/support.png
    Timestamp: 2019-01-02 04:35:02+00:33, Page:
/static/images/guarantees/fastDelivery.png
    Timestamp: 2019-01-02 04:35:03+00:33, Page:
/m/browse/dishwasher/%D9%85%D8%A7%D8%B4%DB%8C%D9%86-
%D8%B8%D8%B1%D9%81%D8%B4%D9%88%DB%8C%DB%8C
    Timestamp: 2019-01-02 04:36:00+00:33, Page: /m/browse/sewing-
machine/%DA%86%D8%B1%D8%AE-%D8%AE%DB%8C%D8%A7%D8%B7%DB%8C
    Timestamp: 2019-01-02 04:36:02+00:33, Page: /m/browse/sewing-
machine/%DA%86%D8%B1%D8%AE-%D8%AE%DB%8C%D8%A7%D8%B7%DB%8C
```

Using time heuristics

Went over each row in the sorted DataFrame, tracking changes in client, user agent, or time gap exceeding the threshold to identify sessions and store session information.

```
# Define the session threshold time in seconds (10 minutes)
SESSION_THRESHOLD_SECONDS = 10 * 60

# Sort the logs_df by client, user_agent, and datetime
logs_df_sorted = logs_df.sort_values(by=['client', 'user_agent',
'datetime'])

# Initialize empty lists to store session information
session_info = []

# Initialize variables for tracking sessions
current_client = None
current_user_agent = None
current_session_start = None
current_session_end = None
current_session_pages = []

# Iterate over each row in the sorted DataFrame
for index, row in logs_df_sorted.iterrows():
    # Check if the client or user_agent has changed, or if the time
gap exceeds the threshold
    if (row['client'] != current_client or row['user_agent'] !=
current_user_agent or
            (current_session_start and (row['datetime'] -
current_session_end).seconds > SESSION_THRESHOLD_SECONDS)):
        # If so, store the current session information
        if current_session_start:
            session_info.append((current_client, current_user_agent,
current_session_start, current_session_end, current_session_pages))
```

```
            # Check if we have at least five sessions, if so, break
the loop
            if len(session_info) >= 5:
                break

        # Start a new session
        current_client = row['client']
        current_user_agent = row['user_agent']
        current_session_start = row['datetime']
        current_session_end = row['datetime']
        current_session_pages = [(row['datetime'], row['request'])]
    else:
        # Otherwise, add the page to the current session
        current_session_pages.append((row['datetime'],
row['request']))
        # Update session end time
        current_session_end = row['datetime']

# Display session information for at least five sessions
index = 0
for session in session_info:
    print(f"Session {index+1}")
    print("Client:", session[0])
    print("User Agent:", session[1])
    print("Session Start Time:", session[2])
    print("Session End Time:", session[3])
    print("Pages Visited:")
    for timestamp, page in session[4]:
        print(f"    {timestamp}: {page}")
    index +=1
    print("\n\n")
```

```
Session 1
Client: 104.156.210.196
User Agent: Dalvik/2.1.0 (Linux; U; Android 8.0.0; SM-A720F
Build/R16NW)
Session Start Time: 2019-01-02 04:20:00+00:33
Session End Time: 2019-01-02 04:20:00+00:33
Pages Visited:
    2019-01-02 04:20:00+00:33: /image/32768?name=24xs450-
33.jpg&wh=200x200



Session 2
Client: 104.194.24.33
User Agent: Mozilla/5.0 (Linux; Android 8.0.0; SM-G955F)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.99 Mobile
Safari/537.36
Session Start Time: 2019-01-02 03:57:00+00:33
Session End Time: 2019-01-02 03:57:00+00:33
Pages Visited:
    2019-01-02 03:57:00+00:33: /amp-helper-frame.html?appId=a624a1c1-
0c93-466a-a546-e146710f97e6&parentOrigin=https://www-zanbil-
ir.cdn.ampproject.org



Session 3
Client: 104.194.24.54
User Agent: Dalvik/2.1.0 (Linux; U; Android 6.0.1; SM-G900H
Build/MMB29K)
Session Start Time: 2019-01-02 04:24:00+00:33
Session End Time: 2019-01-02 04:26:04+00:33
Pages Visited:
    2019-01-02 04:24:00+00:33: /image/33888?name=model-b2048u-
1-.jpg&wh=200x200
    2019-01-02 04:26:04+00:33: /image/11947?name=11947-1-
fw.jpg&wh=200x200



Session 4
Client: 104.194.25.207
User Agent: Dalvik/2.1.0 (Linux; U; Android 5.0.2; P01V Build/LRX22G)
Session Start Time: 2019-01-02 04:06:04+00:33
Session End Time: 2019-01-02 04:06:05+00:33
Pages Visited:
    2019-01-02 04:06:04+00:33: /image/33888?name=model-b2048u-
1-.jpg&wh=200x200
    2019-01-02 04:06:05+00:33: /image/11947?name=11947-1-
fw.jpg&wh=200x200
    2019-01-02 04:06:05+00:33: /image/11926?
name=sm812aaa.jpg&wh=200x200



Session 5
Client: 104.248.138.218
User Agent: Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like Mac OS X)
AppleWebKit/605.1.15 (KHTML, like Gecko) Version/12.0 Mobile/15E148
Safari/604.1
Session Start Time: 2019-01-02 04:35:01+00:33
Session End Time: 2019-01-02 04:36:02+00:33
Pages Visited:
    2019-01-02 04:35:01+00:33: /m/browse/sewing-machine/%DA
%86%D8%B1%D8%AE-%D8%AE%DB%8C%D8%A7%D8%B7%DB%8C
    2019-01-02 04:35:01+00:33: /favicon.ico
    2019-01-02 04:35:01+00:33:
/static/images/guarantees/goodShopping.png
```

```
    2019-01-02 04:35:02+00:33: /static/css/font/wyekan/font.woff
    2019-01-02 04:35:02+00:33: /static/images/guarantees/bestPrice.png
    2019-01-02 04:35:02+00:33: /static/images/guarantees/warranty.png
    2019-01-02 04:35:02+00:33: /static/images/guarantees/support.png
    2019-01-02 04:35:02+00:33: 
/static/images/guarantees/fastDelivery.png
    2019-01-02 04:35:03+00:33: 
/m/browse/dishwasher/%D9%85%D8%A7%D8%B4%DB%8C%D9%86-
%D8%B8%D8%B1%D9%81%D8%B4%D9%88%DB%8C%DB%8C
    2019-01-02 04:36:00+00:33: /m/browse/sewing-machine/%DA
%86%D8%B1%D8%AE-%D8%AE%DB%8C%D8%A7%D8%B7%DB%8C
    2019-01-02 04:36:02+00:33: /m/browse/sewing-machine/%DA
%86%D8%B1%D8%AE-%D8%AE%DB%8C%D8%A7%D8%B7%DB%8C
```

Create a dataframe of session the the session time interval of 10 minutes

```
# Create a session dataframe from the session info array
session_df = pd.DataFrame(session_info)

# Set the columns
session_df.columns = ['client', 'user_agent', 'start_time', 
'end_time',  'pages']

# Extract only the pages from the tuple
session_df['pages'] = session_df['pages'].apply(lambda x: [page[1] for 
page in x])

# Diplaying the first 2 rows of the dataframe
session_df.head(2)

            client                                        user_agent  
\
0  104.156.210.196  Dalvik/2.1.0 (Linux; U; Android 8.0.0; SM-A720...

1    104.194.24.33  Mozilla/5.0 (Linux; Android 8.0.0; SM-G955F) A...


                 start_time                  end_time  \
0 2019-01-02 04:20:00+00:33 2019-01-02 04:20:00+00:33
1 2019-01-02 03:57:00+00:33 2019-01-02 03:57:00+00:33

                                               pages
0      [/image/32768?name=24xs450-33.jpg&wh=200x200]
1  [/amp-helper-frame.html?appId=a624a1c1-0c93-46...

session_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   client      5 non-null      object
 1   user_agent  5 non-null      object
 2   start_time  5 non-null      datetime64[ns, pytz.FixedOffset(33)]
 3   end_time    5 non-null      datetime64[ns, pytz.FixedOffset(33)]
 4   pages       5 non-null      object
dtypes: datetime64[ns, pytz.FixedOffset(33)](2), object(3)
memory usage: 328.0+ bytes
```

Q3. Five frequent referrer website

First cleaned the referer column by droping the rows with referer missing then filtering the malformed URLs, and finally extracting the base URL.

```
from urllib.parse import urlparse

# Drop rows where 'referer' is NaN
referer_df = logs_df.dropna(subset=['referer'])

# Filter out URLs that are not well-formed
referer_df = referer_df[referer_df['referer'].apply(lambda x:
re.match(r'^https?://', x) is not None)]

# Parse the referer URLs to extract the base URL
referer_df['base_url'] = referer_df['referer'].apply(lambda x:
urlparse(x.replace('"', '')).scheme + "://" + urlparse(x.replace('"',
'')).netloc)

# Find the top 5 frequent referrer websites
counting_referer_occurence =
referer_df['base_url'].value_counts().sort_values(ascending=False)

# Get top 5
top_5_referers = counting_referer_occurence.head(5)

# Display the top 5 frequent referrer websites
for i, (referer, count) in enumerate(top_5_referers.items(), start=1):
    print(f"{i}. {referer} - Count: {count}")
```

```
1. https://www.zanbil.ir - Count: 3886
2. https://znbl.ir - Count: 141
3. https://torob.com - Count: 91
4. https://www-zanbil-ir.cdn.ampproject.org - Count: 72
5. http://www.zanbil.ir - Count: 50
```

```
# Setting the maximum column width to None allows for displaying the
full content of each column without truncation.
pd.set_option('display.max_colwidth', None)
```

Q4. **Pages that are frequently visited together with a support ratio not less than 25%.**

Used the dataframe with the sessions grouped because for the pages to be said they are frequently visited together, they must be in the same session.

```
import pandas as pd
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

# Convert pages visited per session into a list of lists format
pages_accessed = session_df['pages'].tolist()

# Initialize TransactionEncoder
te = TransactionEncoder()

# Fit and transform the data into a one-hot encoded DataFrame
onehot = te.fit_transform(pages_accessed)

# Convert the one-hot encoded DataFrame into a DataFrame
df = pd.DataFrame(onehot, columns=te.columns_)

# Find frequent itemsets using Apriori with minimum support of 0.25
frequent_itemsets = apriori(df, min_support=0.25, use_colnames=True)
```

Filtered itemsets with more than one items because we are looking for pages that are visted together which means the itemset should have 2 or more pages

```
# Filter frequent itemsets to include only those with more than 1
items
filter_frequent_itemsets =
frequent_itemsets[frequent_itemsets['itemsets'].apply(lambda x:
len(x)) > 1]

# Display frequent itemsets
print("Frequent Itemsets:")
filter_frequent_itemsets

Frequent Itemsets:

   support  \
2      0.4


itemsets
```

```
2  (/image/33888?name=model-b2048u-1-.jpg&wh=200x200, /image/11947?
name=11947-1-fw.jpg&wh=200x200)
```

Insight: /image/11947?name=11947-1-fw.jpg&wh=200x200 and /image/33888?name=model-b2048u-1-.jpg&wh=200x200 are frequently visited together

Q5. **Association rules with lift values not less than 2.05**

```
# Getting the rules with the lift above 2.05
rules = association_rules(frequent_itemsets, metric='lift',
min_threshold=2.05)

# Displaying rules in a formatted manner
print("Association Rules with Lift > 2.05:\n")
for index, rule in rules.iterrows():
    antecedents = ', '.join(list(rule['antecedents']))
    consequents = ', '.join(list(rule['consequents']))
    support = rule['support']
    confidence = rule['confidence']
    lift = rule['lift']
    print(f"Rule {index+1}: {antecedents} -> {consequents}")
    print(f"Support: {support:.4f}, Confidence: {confidence:.4f},
Lift: {lift:.4f}\n")

Association Rules with Lift > 2.05:

Rule 1: /image/33888?name=model-b2048u-1-.jpg&wh=200x200 ->
/image/11947?name=11947-1-fw.jpg&wh=200x200
Support: 0.4000, Confidence: 1.0000, Lift: 2.5000

Rule 2: /image/11947?name=11947-1-fw.jpg&wh=200x200 -> /image/33888?
name=model-b2048u-1-.jpg&wh=200x200
Support: 0.4000, Confidence: 1.0000, Lift: 2.5000
```

Q6. Ten frequent sequential patterns

The GSP algorithm identifies patterns that occur frequently among user navigation sequences, which helps to understand common browsing behaviors.

```
!pip install gsppy

Collecting gsppy
  Downloading gsppy-1.1-py3-none-any.whl.metadata (3.1 kB)
Downloading gsppy-1.1-py3-none-any.whl (5.7 kB)
Installing collected packages: gsppy
Successfully installed gsppy-1.1

import argparse
import logging
```

```python
import random
from gsppy.gsp import GSP
logging.basicConfig(level=logging.DEBUG)

result = GSP(pages_accessed).search(0.25)

result
```

```
[{('/image/33888?name=model-b2048u-1-.jpg&wh=200x200',): 2,
  ('/image/11947?name=11947-1-fw.jpg&wh=200x200',): 2},
 {('/image/33888?name=model-b2048u-1-.jpg&wh=200x200',
   '/image/11947?name=11947-1-fw.jpg&wh=200x200'): 2}]
```

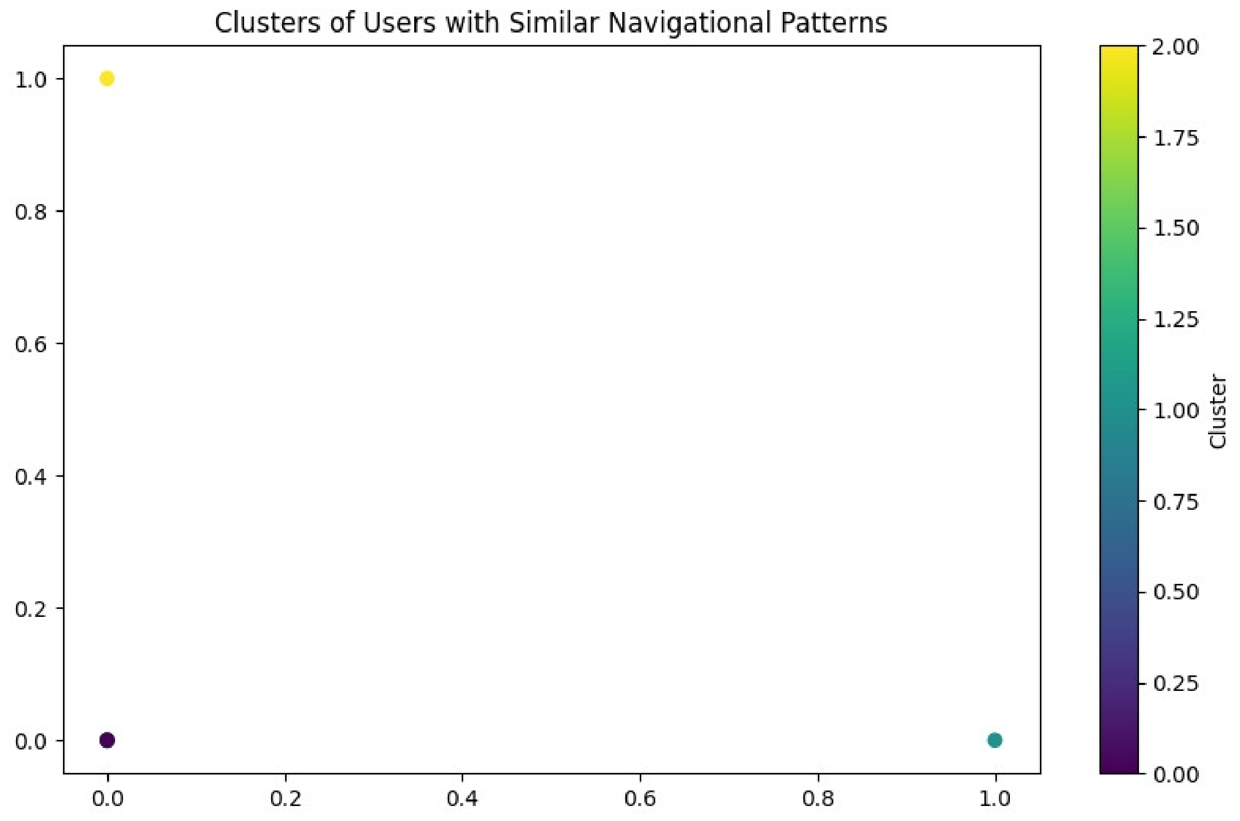
```python
max_length = 3
# Filter out frequent sequential patterns based on maximum length
filtered_patterns = [pattern for pattern in result if len(pattern) <=
max_length]

# Display the frequent sequential patterns
for pattern in filtered_patterns:
    print(pattern)
```

```
{('/image/33888?name=model-b2048u-1-.jpg&wh=200x200',): 2,
('/image/11947?name=11947-1-fw.jpg&wh=200x200',): 2}
{('/image/33888?name=model-b2048u-1-.jpg&wh=200x200', '/image/11947?
name=11947-1-fw.jpg&wh=200x200'): 2}
```

When I went lower than 0.25, the GSP was taking hours to run and the RAM ran out

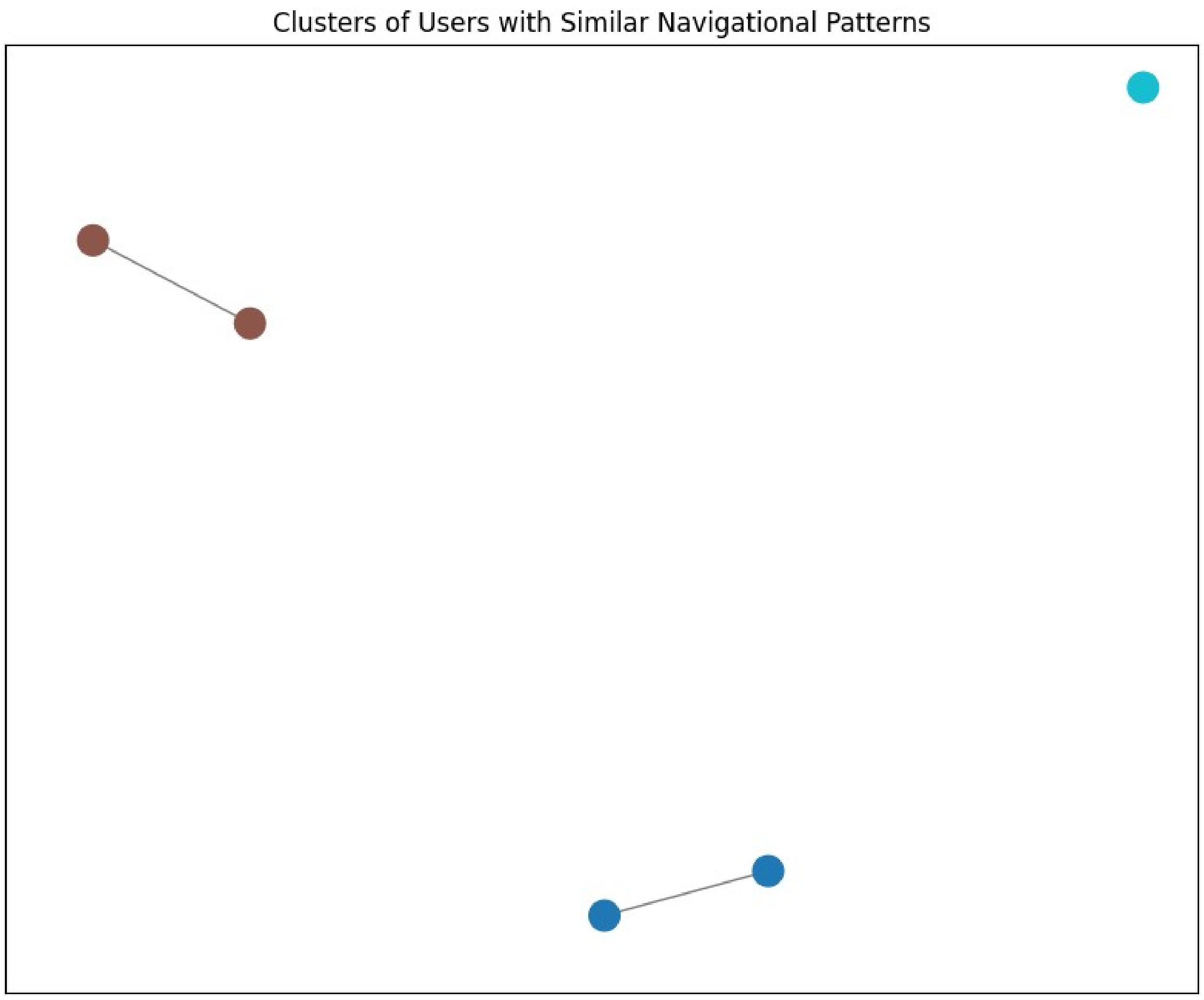
Q7. Graph that shows clusters of users with similar navigational patterns

```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Setting patterns into a suitable format for analysis
patterns = onehot

# Cluster users using K-means
kmeans = KMeans(n_clusters=3,  n_init=10)
clusters = kmeans.fit_predict(patterns)

# Visualize clusters
plt.figure(figsize=(10, 6))
plt.scatter(patterns[:, 0], patterns[:, 1], c=clusters,
cmap='viridis')
plt.title('Clusters of Users with Similar Navigational Patterns')
plt.colorbar(label='Cluster')
plt.show()
```



```
import networkx as nx
import matplotlib.pyplot as plt

# Create a graph
G = nx.Graph()

# Add nodes (users) to the graph
for i in range(len(patterns)):
    # Assign cluster as a node attribute
    G.add_node(i, label=f"User {i}", cluster=clusters[i])

# Add edges between similar users (in the same cluster)
for i in range(len(patterns)):
    for j in range(i + 1, len(patterns)):
        if clusters[i] == clusters[j]:
            G.add_edge(i, j)

# Visualize the graph
plt.figure(figsize=(10, 8))
# Layout for the graph
pos = nx.spring_layout(G)
```

```
# Draw nodes colored by cluster
node_color = [clusters[node] for node in G.nodes()]
nx.draw_networkx_nodes(G, pos, node_color=node_color,
cmap=plt.cm.tab10, node_size=200)

# Draw edges
nx.draw_networkx_edges(G, pos, alpha=0.5)

plt.title('Clusters of Users with Similar Navigational Patterns')
# plt.axis('off')
plt.show()
```



```python
# This Python 3 environment comes with many helpful analytics
libraries installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/"
directory
# For example, running this (by clicking run or pressing Shift+Enter)
will list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/)
that gets preserved as output when you create a version using "Save &
Run All"
# You can also write temporary files to /kaggle/temp/, but they won't
be saved outside of the current session
```

```
/kaggle/input/web-server-access-logs/access.log
/kaggle/input/web-server-access-logs/client_hostname.csv
```

```python
import re
import os
import time
from tqdm import tqdm

common_regex = '^(?P<client>\S+) \S+ (?P<userid>\S+) \[(?
P<datetime>[^\]]+)\] "(?P<method>[A-Z]+) (?P<request>[^ "]+)? HTTP/[0-
9.]+" (?P<status>[0-9]{3}) (?P<size>[0-9]+|-)'
combined_regex = '^(?P<client>\S+) \S+ (?P<userid>\S+) \[(?
P<datetime>[^\]]+)\] "(?P<method>[A-Z]+) (?P<request>[^ "]+)? HTTP/[0-
9.]+" (?P<status>[0-9]{3}) (?P<size>[0-9]+|-) "(?P<referrer>[^"]*)"
"(?P<useragent>[^"]*)'
columns = ['client', 'userid', 'datetime', 'method', 'request',
'status', 'size', 'referer', 'user_agent']

def logs_to_df(logfile, output_dir, errors_file):
    with open(logfile) as source_file:
        linenumber = 0
        parsed_lines = []
        for line in tqdm(source_file):
            try:
                log_line = re.findall(combined_regex, line)[0]
                parsed_lines.append(log_line)
            except Exception as e:
                with open(errors_file, 'at') as errfile:
                    print((line, str(e)), file=errfile)
                continue
            linenumber += 1
            if linenumber % 250_000 == 0:
                df = pd.DataFrame(parsed_lines, columns=columns)

df.to_parquet(f'{output_dir}/file_{linenumber}.parquet')
                parsed_lines.clear()
        else:
            df = pd.DataFrame(parsed_lines, columns=columns)
            df.to_parquet(f'{output_dir}/file_{linenumber}.parquet')
            parsed_lines.clear()
```

```
mkdir parquet_dir
```

```
logs_to_df(logfile='/kaggle/input/web-server-access-logs/access.log',
output_dir='parquet_dir/', errors_file='errors.txt')
```

```
10365152it [02:26, 70811.95it/s]
```

```
logs_df = pd.read_parquet('parquet_dir/')
```

```
logs_df
```

```
                  client userid                   datetime method  \
0          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
1          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
2            85.9.73.119      -  22/Jan/2019:12:38:27 +0330    GET
3          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
4            85.9.73.119      -  22/Jan/2019:12:38:27 +0330    GET
...                  ...    ...                        ...    ...
10364860  86.104.110.254      -  26/Jan/2019:16:01:31 +0330    GET
10364861   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364862    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET
10364863   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364864    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET

                                                   request status
size  \
0         /image/29314?name=%D8%AF%DB%8C%D8%A8%D8%A7-7.j...    200
1105
1                            /static/images/zanbil-kharid.png    200
358
2                                     /static/images/next.png    200
3045
3         /image/29314?name=%D8%AF%DB%8C%D8%A8%D8%A7-4.j...    200
1457
4                                  /static/images/checked.png    200
1083
```

```
...                                                     ...    ...   .
..
10364860                                   /settings/logo    200
4120
10364861                                   /image/5/brand    200
2171
10364862                /image/64646/productModel/150x150    200
5318
10364863                                   /image/1/brand    200
3924
10364864                /image/56698/productModel/150x150    200
3570

                                                    referer  \
0         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
1         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
2         https://znbl.ir/static/bundle-bundle_site_head...
3         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
4         https://znbl.ir/static/bundle-bundle_site_head...
...                                                     ...
10364860  https://www.zanbil.ir/m/browse/tv/%D8%AA%D9%84...
10364861      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364862  https://www.zanbil.ir/browse/audio-and-video-e...
10364863      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364864  https://www.zanbil.ir/browse/audio-and-video-e...

                                                 user_agent
0         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
1         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
2         Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...
3         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
4         Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...
...                                                     ...
10364860  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like M...
10364861  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364862  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...
10364863  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364864  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...

[10364865 rows x 9 columns]
```

```
df = logs_df.query("request.str.contains('.css') == False and
request.str.contains('.png') == False and request.str.contains('.jpg')
== False and request.str.contains('.jpeg') == False and
request.str.contains('.mp3') == False and request.str.contains('.js')
== False")

df
```

```
                   client userid                     datetime method  \
5         37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
6         77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
7         37.27.128.139      -  22/Jan/2019:12:38:27 +0330    GET
8         77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
9         77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
...                 ...    ...                         ...    ...
10364860  86.104.110.254      -  26/Jan/2019:16:01:31 +0330    GET
10364861   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364862    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET
10364863   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364864    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET

                                                   request status
size  \
5                               /static/images/loading.gif    200
7370
6                         /image/11082/productType/240x180    200
12458
7         /browse/Tablet-Arm-Chair/%D8%B5%D9%86%D8%AF%D9...    200
30604
8                                      /image/851/mainSlide    200
89859
9                                      /image/848/mainSlide    200
93168
...                                                     ...    ...
...
10364860                                     /settings/logo    200
4120
10364861                                     /image/5/brand    200
2171
10364862                 /image/64646/productModel/150x150    200
5318
10364863                                     /image/1/brand    200
3924
10364864                 /image/56698/productModel/150x150    200
3570

                                                   referer  \
5         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
6         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
7         https://www.zanbil.ir/browse/Classroom-Furnitu...
8         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
9         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
...                                                     ...
10364860  https://www.zanbil.ir/m/browse/tv/%D8%AA%D9%84...
10364861      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364862  https://www.zanbil.ir/browse/audio-and-video-e...
10364863      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364864  https://www.zanbil.ir/browse/audio-and-video-e...
```

```
                                                       user_agent
5         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
6         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
7         Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.3...
8         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
9         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
...                                                     ...
10364860  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like M...
10364861  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364862  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...
10364863  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364864  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...

[7305489 rows x 9 columns]
```

```
df1 = df.query("method.str.contains('POST') == False")

df1
```

```
                  client userid                     datetime method  \
5          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
6          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
7          37.27.128.139      -  22/Jan/2019:12:38:27 +0330    GET
8          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
9          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
...                  ...    ...                        ...    ...
10364860  86.104.110.254      -  26/Jan/2019:16:01:31 +0330    GET
10364861   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364862    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET
10364863   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364864    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET

                                                    request status
size  \
5                                  /static/images/loading.gif    200
7370
6                             /image/11082/productType/240x180    200
12458
7         /browse/Tablet-Arm-Chair/%D8%B5%D9%86%D8%AF%D9...    200
30604
8                                       /image/851/mainSlide    200
89859
9                                       /image/848/mainSlide    200
93168
...                                                     ...    ...
...
10364860                                      /settings/logo    200
4120
10364861                                      /image/5/brand    200
```

```
2171
10364862                  /image/64646/productModel/150x150    200
5318
10364863                                   /image/1/brand    200
3924
10364864                  /image/56698/productModel/150x150    200
3570

                                                    referer  \
5         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
6         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
7         https://www.zanbil.ir/browse/Classroom-Furnitu...
8         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
9         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
...                                                     ...
10364860  https://www.zanbil.ir/m/browse/tv/%D8%AA%D9%84...
10364861      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364862  https://www.zanbil.ir/browse/audio-and-video-e...
10364863      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364864  https://www.zanbil.ir/browse/audio-and-video-e...

                                                 user_agent
5         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
6         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
7         Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.3...
8         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
9         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
...                                                     ...
10364860  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like M...
10364861  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364862  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...
10364863  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364864  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...

[7166334 rows x 9 columns]
```

```
df2 = df[df['method'].str.contains("POST") == False]
df2
```

```
                  client userid                     datetime method  \
5          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
6          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
7          37.27.128.139      -  22/Jan/2019:12:38:27 +0330    GET
8          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
9          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
...                  ...    ...                         ...    ...
10364860  86.104.110.254      -  26/Jan/2019:16:01:31 +0330    GET
10364861   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364862    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET
10364863   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
```

```
10364864    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET

                                                        request status
size  \
5                                  /static/images/loading.gif    200
7370
6                             /image/11082/productType/240x180    200
12458
7         /browse/Tablet-Arm-Chair/%D8%B5%D9%86%D8%AF%D9...    200
30604
8                                          /image/851/mainSlide    200
89859
9                                          /image/848/mainSlide    200
93168
...                                                         ...    ...
...
10364860                                       /settings/logo    200
4120
10364861                                       /image/5/brand    200
2171
10364862                     /image/64646/productModel/150x150    200
5318
10364863                                       /image/1/brand    200
3924
10364864                     /image/56698/productModel/150x150    200
3570

                                                        referer  \
5         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
6         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
7         https://www.zanbil.ir/browse/Classroom-Furnitu...
8         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
9         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
...                                                         ...
10364860  https://www.zanbil.ir/m/browse/tv/%D8%AA%D9%84...
10364861      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364862  https://www.zanbil.ir/browse/audio-and-video-e...
10364863      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364864  https://www.zanbil.ir/browse/audio-and-video-e...

                                                     user_agent
5         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
6         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
7         Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.3...
8         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
9         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
...                                                         ...
10364860  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like M...
10364861  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364862  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...
10364863  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364864  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...

[7166334 rows x 9 columns]
```

```
df3 = df2.query("status.str.contains('200') == True")
```

```
df3
```

```
                  client userid                    datetime method  \
5          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
6          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
7          37.27.128.139      -  22/Jan/2019:12:38:27 +0330    GET
8          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
9          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
...                  ...    ...                         ...    ...
10364860  86.104.110.254      -  26/Jan/2019:16:01:31 +0330    GET
10364861   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364862    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET
10364863   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364864    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET

                                                    request status
size  \
5                                 /static/images/loading.gif    200
7370
6                           /image/11082/productType/240x180    200
12458
7         /browse/Tablet-Arm-Chair/%D8%B5%D9%86%D8%AF%D9...    200
30604
8                                      /image/851/mainSlide    200
89859
9                                      /image/848/mainSlide    200
93168
...                                                     ...    ...
...
10364860                                     /settings/logo    200
4120
10364861                                     /image/5/brand    200
2171
10364862                     /image/64646/productModel/150x150    200
5318
10364863                                     /image/1/brand    200
3924
10364864                     /image/56698/productModel/150x150    200
3570

                                                    referer  \
5         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
6         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
```

```
7          https://www.zanbil.ir/browse/Classroom-Furnitu...
8          https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
9          https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
...                                                      ...
10364860  https://www.zanbil.ir/m/browse/tv/%D8%AA%D9%84...
10364861      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364862  https://www.zanbil.ir/browse/audio-and-video-e...
10364863      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364864  https://www.zanbil.ir/browse/audio-and-video-e...

                                                user_agent
5          Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
6          Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
7          Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.3...
8          Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
9          Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
...                                                      ...
10364860  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like M...
10364861  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364862  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...
10364863  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364864  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...

[6871942 rows x 9 columns]
```

```
df3.client
```

```
5             37.152.163.59
6             77.245.233.52
7             37.27.128.139
8             77.245.233.52
9             77.245.233.52
                  ...
10364860    86.104.110.254
10364861     5.125.254.169
10364862      65.49.68.192
10364863     5.125.254.169
10364864      65.49.68.192
Name: client, Length: 6871942, dtype: object
```

```
df3.client.nunique()
```

```
147899
```

```
print(df3.groupby('client').get_group('37.152.163.59'))
```

```
                client userid                     datetime method  \
5        37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
56       37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
186      37.152.163.59      -  22/Jan/2019:12:38:31 +0330    GET
7863700  37.152.163.59      -  22/Jan/2019:12:38:01 +0330    GET
```

```
7863704  37.152.163.59      -  22/Jan/2019:12:38:01 +0330    GET
...                ...    ...                        ...    ...
9828922  37.152.163.59      -  26/Jan/2019:12:57:13 +0330    GET
9828925  37.152.163.59      -  26/Jan/2019:12:57:14 +0330    GET
9828928  37.152.163.59      -  26/Jan/2019:12:57:14 +0330    GET
9828991  37.152.163.59      -  26/Jan/2019:12:57:15 +0330    GET
9828993  37.152.163.59      -  26/Jan/2019:12:57:15 +0330    GET

                                                       request status
size  \
5                                 /static/images/loading.gif    200
7370
56                                  /site/alexaGooleAnalitic    200
323
186                               /static/images/favicon.ico    200
152
7863700  /product/29314/%DA%A9%D8%A7%D9%84%D8%B3%DA%A9%...    200
41580
7863704  /image/%7B%7BbasketItem.id%7D%7D?type=productM...    200
5
...                                                    ...    ...     .
..
9828922                                   /filter/p62,b5    200
34238
9828925                                   /settings/logo    200
4120
9828928               /image/62191/productModel/150x150    200
5862
9828991                                     /site/enamad    200
278
9828993                         /site/alexaGooleAnalitic    200
323

                                                       referer  \
5        https://www.zanbil.ir/product/29314/%DA%A9%D8%...
56       https://www.zanbil.ir/product/29314/%DA%A9%D8%...
186                                                      -
7863700  https://www.google.com/url?sa=t&rct=j&q=&esrc=...
7863704  https://www.zanbil.ir/product/29314/%DA%A9%D8%...
...                                                    ...
9828922              https://www.zanbil.ir/filter/b5,p62
9828925              https://www.zanbil.ir/filter/p62,b5
9828928              https://www.zanbil.ir/filter/p62,b5
9828991              https://www.zanbil.ir/filter/p62,b5
9828993              https://www.zanbil.ir/filter/p62,b5

                                                    user_agent
5        Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
56       Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
186      Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
7863700  Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
7863704  Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
...                                                   ...
9828922  Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...
9828925  Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...
9828928  Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...
9828991  Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...
9828993  Mozilla/5.0 (Windows NT 6.1; Win64; x64) Apple...

[136 rows x 9 columns]

df3

                  client userid                    datetime method  \
5          37.152.163.59      -  22/Jan/2019:12:38:27 +0330    GET
6          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
7          37.27.128.139      -  22/Jan/2019:12:38:27 +0330    GET
8          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
9          77.245.233.52      -  22/Jan/2019:12:38:27 +0330    GET
...                  ...    ...                         ...    ...
10364860  86.104.110.254      -  26/Jan/2019:16:01:31 +0330    GET
10364861   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364862    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET
10364863   5.125.254.169      -  26/Jan/2019:16:01:31 +0330    GET
10364864    65.49.68.192      -  26/Jan/2019:16:01:31 +0330    GET

                                                    request status
size  \
5                                /static/images/loading.gif    200
7370
6                          /image/11082/productType/240x180    200
12458
7         /browse/Tablet-Arm-Chair/%D8%B5%D9%86%D8%AF%D9...    200
30604
8                                     /image/851/mainSlide    200
89859
9                                     /image/848/mainSlide    200
93168
...                                                     ...    ...
...
10364860                                     /settings/logo    200
4120
10364861                                     /image/5/brand    200
2171
10364862                      /image/64646/productModel/150x150    200
5318
10364863                                     /image/1/brand    200
3924
10364864                      /image/56698/productModel/150x150    200
3570
```

```
                                                    referer  \
5         https://www.zanbil.ir/product/29314/%DA%A9%D8%...
6         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
7         https://www.zanbil.ir/browse/Classroom-Furnitu...
8         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
9         https://www.zanbil.ir/browse/sports/%D8%AA%D8%...
...                                                     ...
10364860  https://www.zanbil.ir/m/browse/tv/%D8%AA%D9%84...
10364861      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364862  https://www.zanbil.ir/browse/audio-and-video-e...
10364863      https://www.zanbil.ir/m/filter/p62%2Cstexists
10364864  https://www.zanbil.ir/browse/audio-and-video-e...

                                                 user_agent
5         Mozilla/5.0 (Windows NT 6.1; WOW64; Trident/7....
6         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
7         Mozilla/5.0 (Windows NT 5.1) AppleWebKit/537.3...
8         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
9         Mozilla/5.0 (Windows NT 6.1; rv:64.0) Gecko/20...
...                                                     ...
10364860  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1 like M...
10364861  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364862  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...
10364863  Mozilla/5.0 (iPhone; CPU iPhone OS 12_1_2 like...
10364864  Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:6...

[6871942 rows x 9 columns]
```

Clickstream Sales Analysis

Focusing on Sales by Month, Clothing Type, and Display Page

By Eric Wilson

Image via Google

Importing Libraries and Data

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

csdata = pd.read_csv('../input/clickstream-data-for-online-shopping/e-shop clothing 2008.csv',
                     delimiter = ';')
print(csdata.shape)
csdata.head(3)
```

```
(165474, 14)

   year  month  day  order  country  session ID  page 1 (main category)  \
0  2008      4    1      1       29           1                       1
1  2008      4    1      2       29           1                       1
2  2008      4    1      3       29           1                       2

  page 2 (clothing model)  colour  location  model photography  price  \
0                     A13       1         5                  1     28
1                     A16       1         6                  1     33
2                      B4      10         2                  1     52
```

```
   price 2  page
0        2     1
1        2     1
2        1     1
```

Work Citation as per Request on Kaggle

Å)apczyÅ„ski M., BiaÅ‚owÄ…s S. (2013) Discovering Patterns of Users' Behaviour in an E-shop - Comparison of Consumer Buying Behaviours in Poland and Other European Countries, â€œStudia Ekonomiczneâ€, nr 151, â€œLa sociÃ©tÃ© de l'information : perspective europÃ©enne et globale : les usages et les risques d'Internet pour les citoyens et les consommateursâ€, p. 144-153

Defining Goals and Variables

Questions / Goals

1. When do sales peak?
2. What type of clothing sells most? What type of clothing sells most per month?
3. Does a correlation exist between price and page, and, if so, how strongly are price and product placement related?

Predictions

1. I expect sales to peak in June, as buyers purchase clothing for vacation months / outdoor months.
2. No strong feelings / expectations of what to find
3. I believe higher priced items will be located towards the front page, in order to maximize profits.

Chosen Variables

The columns which will be relevant for this analysis are (as defined in the uploaded data):

- MONTH -> from April (4) to August (8)
- DAY -> day number of the month
- PAGE 1 (MAIN CATEGORY) -> concerns the main product category:

```
– 1-trousers
– 2-skirts
– 3-blouses
– 4-sale
```

- PRICE -> price in US dollars

- PAGE -> page number within the e-store website (from 1 to 5)

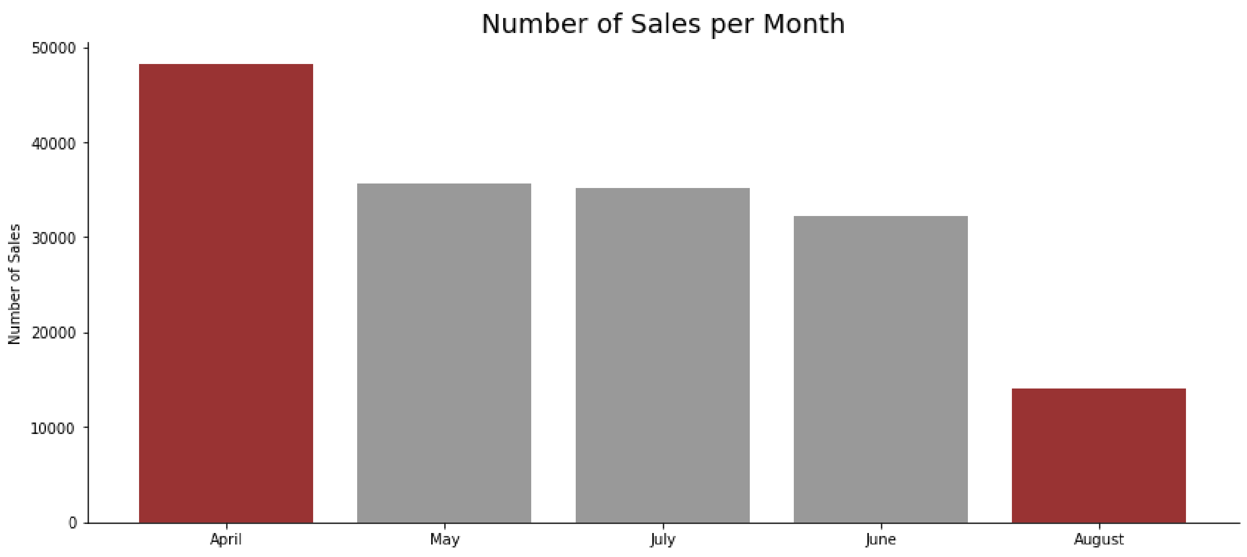
```
### New dataframe relevant columns
csdf = csdata[['month', 'day', 'page 1 (main category)', 'price',
'page']]
csdf = csdf.rename(columns={'month':'Month', 'day':'Day', 'page 1
(main category)':'Type',
                     'price':'Price', 'page':'Page'})
csdf.Type = csdf.Type.replace({1: 'Trousers', 2: 'Skirts', 3:
'Blouses', 4: 'Sale'})
csdf.Month = csdf.Month.replace({4: 'April', 5: 'May', 6: 'June',
7:'July', 8: 'August'})
csdf.head()
```

```
   Month  Day      Type  Price  Page
0  April    1  Trousers     28     1
1  April    1  Trousers     33     1
2  April    1    Skirts     52     1
3  April    1    Skirts     38     1
4  April    1    Skirts     52     1
```

Data Exploration

Sales by Month

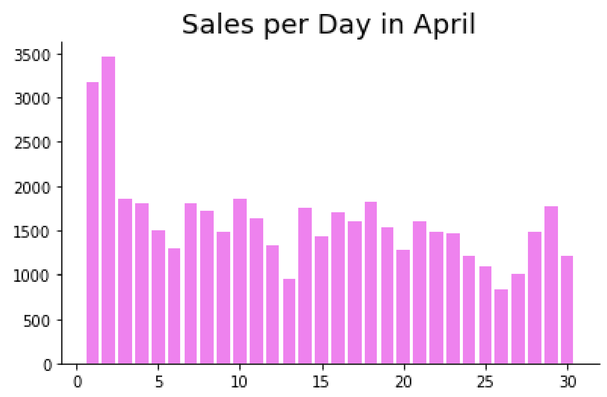
```
### Number of goods sold each month
csmsm = csdf.Month.value_counts()

fig , ax = plt.subplots(figsize = [14,6])

ax.bar(csmsm.keys(), csmsm.values,
color=['maroon','gray','gray','gray','maroon'], alpha=.8)

ax.set_title('Number of Sales per Month', fontsize = 18)
ax.set_ylabel('Number of Sales')
ax.spines[['right', 'top']].set_visible(False)

plt.show()
```

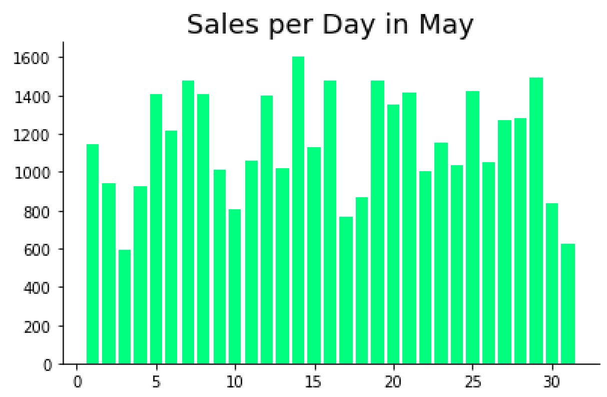


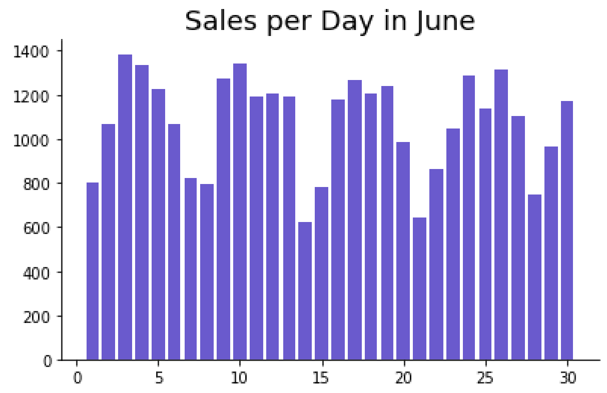
April has the highest number of sales, and August has the fewest; August, however, seems surprisingly low - could we have incomplete data for that month? Ultimately, it is probably worth checking each month, just to ensure a full range of dates, and to see the sales trends as they unfold throughout the month.

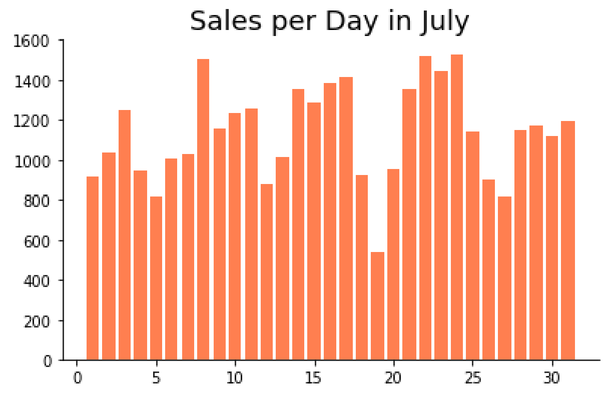
```
### Dataframes for each month
csau = csdf.loc[csdf['Month'] == 'August']
csjn =csdf.loc[csdf['Month'] == 'June']
csjl = csdf.loc[csdf['Month'] == 'July']
csmy = csdf.loc[csdf['Month'] == 'May']
csap = csdf.loc[csdf['Month'] == 'April']

fig, axs = plt.subplots(nrows=4, ncols = 2, figsize=[14,18])

axs[0,0].bar(csap.Day.value_counts().keys(),
csap.Day.value_counts().values, color='violet')
axs[0,0].set_title('Sales per Day in April', fontsize=18)
axs[0,0].spines[['right', 'top']].set_visible(False)

axs[0,1].bar(csmy.Day.value_counts().keys(),
csmy.Day.value_counts().values, color='springgreen')
axs[0,1].set_title('Sales per Day in May', fontsize=18)
axs[0,1].spines[['right', 'top']].set_visible(False)

axs[1,0].bar(csjn.Day.value_counts().keys(),
csjn.Day.value_counts().values, color='slateblue')
axs[1,0].set_title('Sales per Day in June', fontsize=18)
axs[1,0].spines[['right', 'top']].set_visible(False)


axs[1,1].bar(csjl.Day.value_counts().keys(),
csjl.Day.value_counts().values, color='coral')
axs[1,1].set_title('Sales per Day in July', fontsize=18)
```

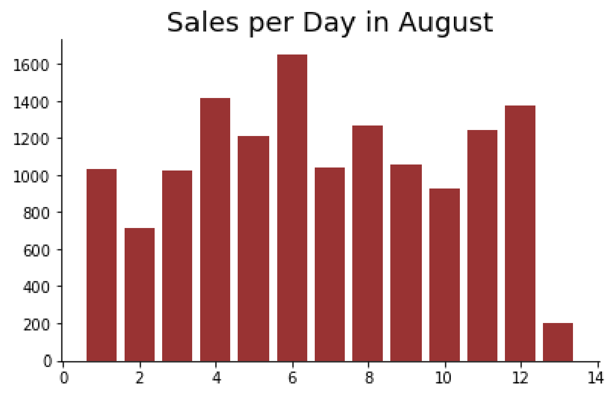
```
axs[1,1].spines[['right', 'top']].set_visible(False)


axs[2,0].bar(csau.Day.value_counts().keys(),
csau.Day.value_counts().values, color='maroon', alpha=.8)
axs[2,0].set_title('Sales per Day in August', fontsize=18)
axs[2,0].spines[['right', 'top']].set_visible(False)


axs[2,1].scatter(csdf.Day.value_counts().keys(),
csdf.Day.value_counts().values, color='dimgray')
axs[2,1].set_title('Sales per Day Each Month', fontsize = 18)
axs[2,1].spines[['right', 'top']].set_visible(False)

csna = csdf.loc[csdf['Month'] != 'August']
csnac = csna.Month.value_counts()

axs[3,0].scatter(csna.Day.value_counts().keys(),
csna.Day.value_counts().values, color='maroon')
axs[3,0].set_title('Sales per Day, April - July', fontsize = 18)
axs[3,0].spines[['right', 'top']].set_visible(False)

axs[3,1].bar(csnac.keys(), csnac.values,
color=['maroon','gray','gray','gray'], alpha=.8)
axs[3,1].set_title('Number of Sales, April - July', fontsize = 18)
axs[3,1].spines[['right', 'top']].set_visible(False)

plt.show()
```

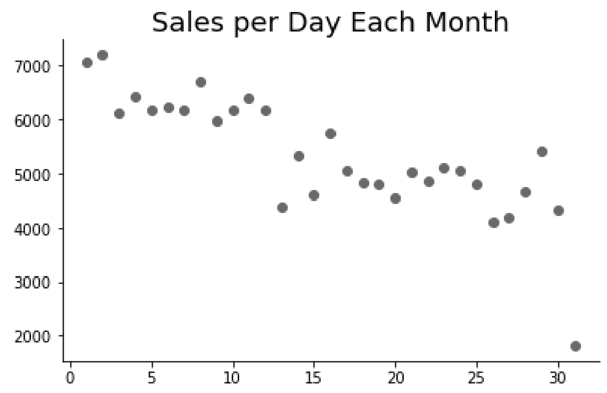


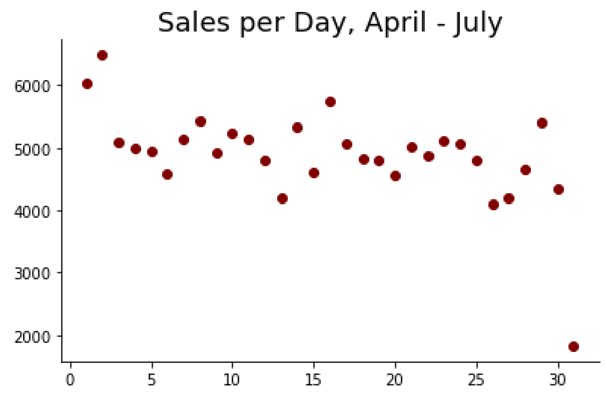


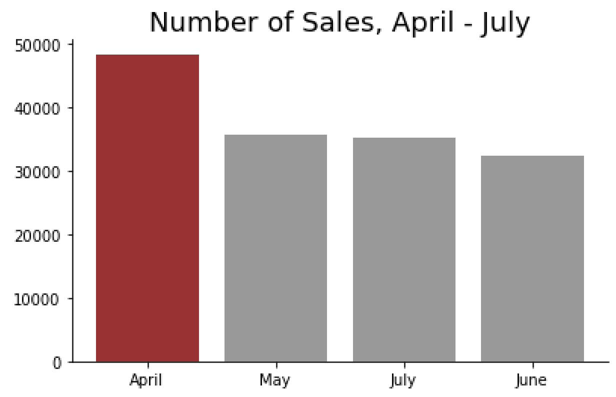








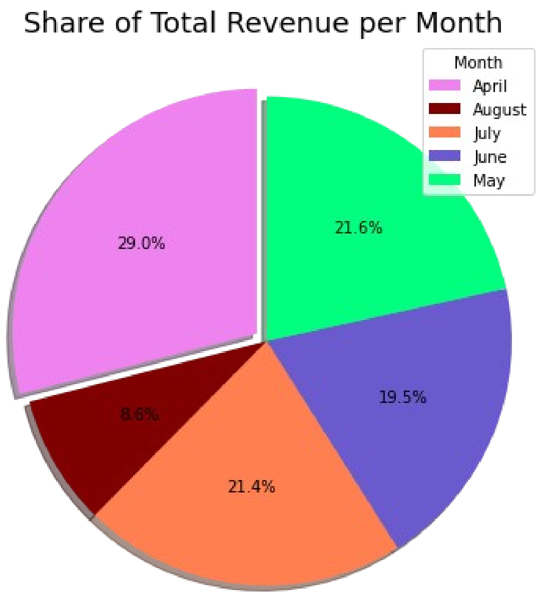


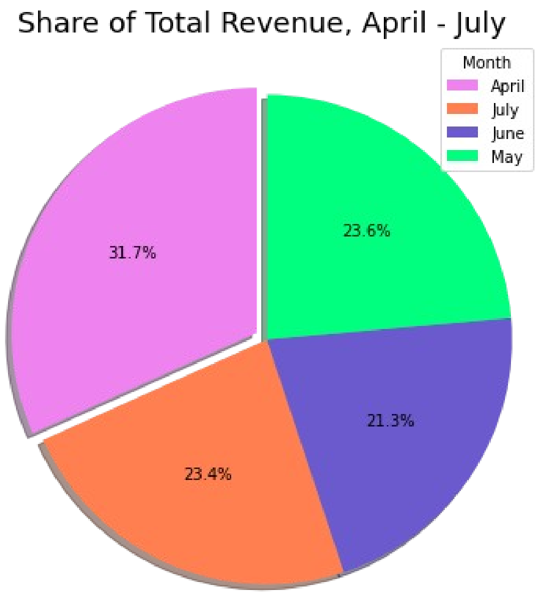


So, it looks like August is incomplete - only about half of the month has data accounted for. April may have either confounding data or incomplete data, as the first two days have so many more sales than any of the other days - maybe cumulative data from March leaked into April?

Interestingly, as for the overall data, it seems that sales peak towards the start of the month, and dwindle as the month goes on - however, the only month that follows that trend exactly is April, whereas the rest of the months seem to be relatively steady throughout.

```
csmp = csdf[['Month', 'Price']]
csmpna = csna[['Month', 'Price']]
csmp2 = csmp.groupby('Month').sum()
csmpna2 = csmpna.groupby('Month').sum()

l1 = csmp2.index
l2 = csmpna2.index

fig, ax = plt.subplots(nrows=1, ncols=2, figsize=[14,7])
ax[0].pie(csmp2.Price, explode=(0.05, 0, 0, 0, 0), autopct='%1.1f%%',
        shadow=True, startangle=90, colors=['violet', 'maroon',
'coral', 'slateblue', 'springgreen'])
ax[0].axis('equal')
ax[0].set_title("Share of Total Revenue per Month", fontsize=18)
ax[0].legend(l1, title="Month", loc="upper right")

ax[1].pie(csmpna2.Price, explode=(0.05, 0, 0, 0), autopct='%1.1f%%',
        shadow=True, startangle=90, colors=['violet', 'coral',
'slateblue', 'springgreen'])
ax[1].axis('equal')
ax[1].set_title("Share of Total Revenue, April - July", fontsize=18)
ax[1].legend(l2, title="Month", loc="upper right")

plt.show()
```



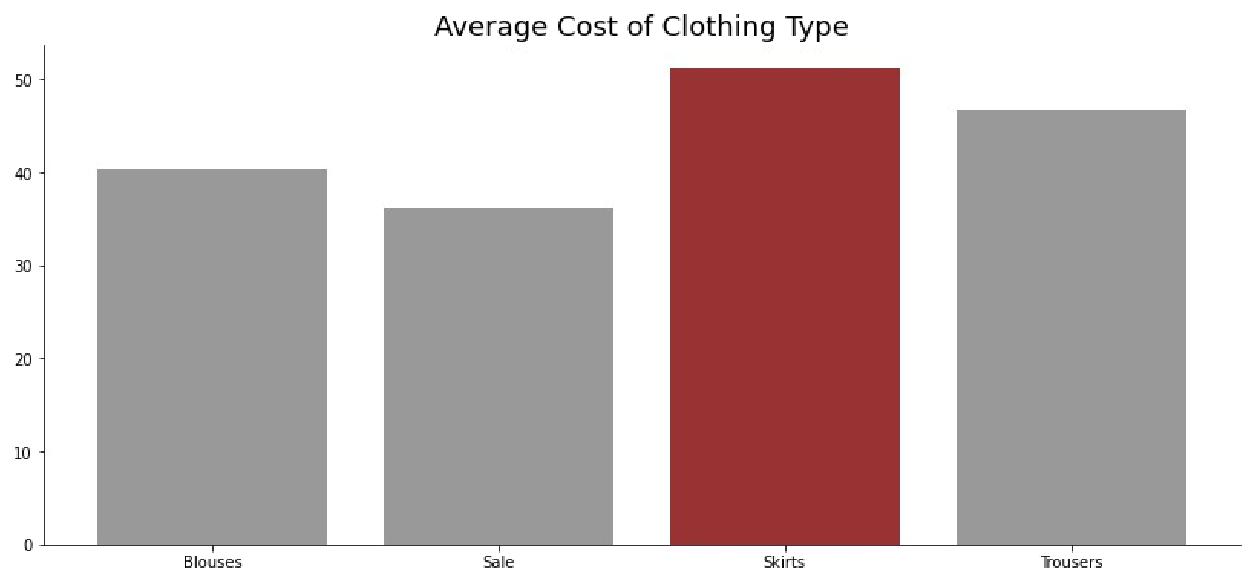


Of all months in the data set, April has the highest revenue, and August has the lowest. If August is removed, in order to control for an incomplete month of data, June becomes the lowest selling month - the exact opposite of what I had predicted / expected to find in the data.

Why might April be the highest sales month? Could it be tax returns (in the US)? Prepping for Spring Break, or Summer Break (maybe the vacation preparation starts earlier for most people than procrastinators, like myself)? Could it be replacing wardrobes after a long winter and spring cleaning?

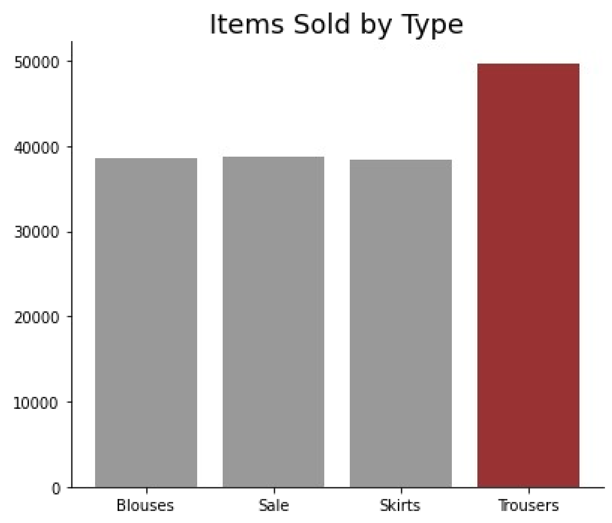
Sales by Clothing Type

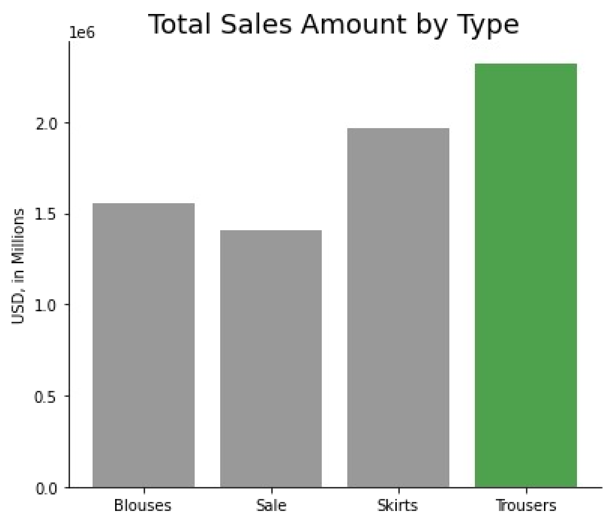
```
### Number of types of clothing sold
csctc = csdf.Type.value_counts()
### Monetary amount sold per type of clothing
csctr = csdf[['Type', 'Price']]
cscts = csctr.groupby('Type').sum()
cscta = csctr.groupby('Type').mean()

csct = cscts
csct['Total'] = csctc
csct['Average'] = cscta['Price']
csct = csct.rename(columns={'Price' : 'Value'})

fig, ax = plt.subplots(figsize = [14,6])

ax.bar(csct.index, csct.Average, color = ['gray', 'gray', 'maroon',
'gray'], alpha = .8)
ax.set_title('Average Cost of Clothing Type', fontsize=18)
ax.spines[['right', 'top']].set_visible(False)

plt.show()
```

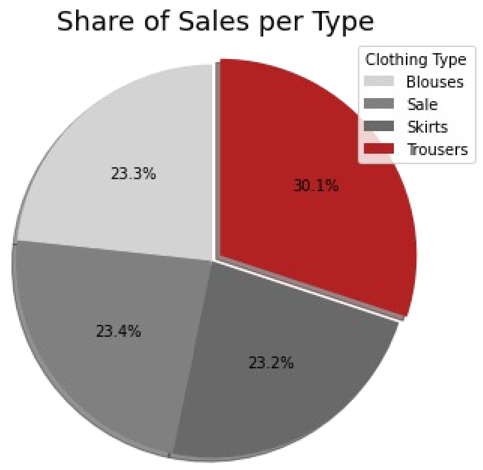


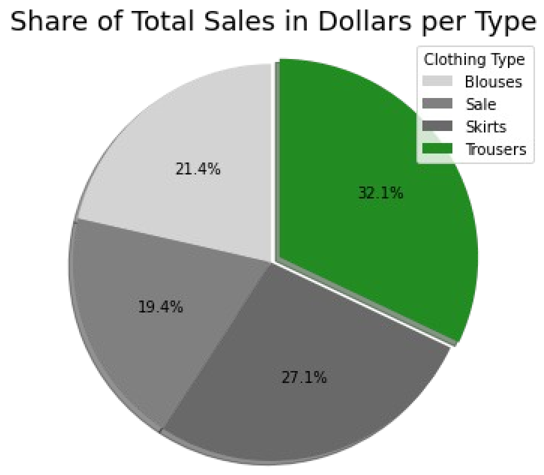
Skirts, on average, have the highest cost, whereas items on sale, intuitively, have the lowest price. How many of each item sell, and what type of monetary value do they generate?

```python
fig, axs = plt.subplots(nrows=2, ncols = 2, figsize=[14, 12])

axs[0,0].bar(csct.index, csct.Total,
           color=['gray', 'gray', 'gray', 'maroon'], alpha=.8)
axs[0,0].set_title('Items Sold by Type', fontsize=18)
axs[0,0].spines[['right', 'top']].set_visible(False)

axs[0,1].bar(csct.index, csct.Value,
           color=['gray', 'gray', 'gray', 'forestgreen'], alpha=.8)
axs[0,1].set_title('Total Sales Amount by Type', fontsize=18)
axs[0,1].set_ylabel('USD, in Millions')
axs[0,1].spines[['right', 'top']].set_visible(False)

axs[1,0].pie(csct.Total, explode=(0, 0, 0, 0.05), autopct='%1.1f%%',
        shadow=True, startangle=90,
        colors=['lightgray', 'gray', 'dimgray', 'firebrick'])
axs[1,0].axis('equal')
axs[1,0].set_title("Share of Sales per Type", fontsize=18)
axs[1,0].legend(csct.index, title="Clothing Type", loc="upper right")

axs[1,1].pie(csct.Value, explode=(0, 0, 0, 0.05), autopct='%1.1f%%',
        shadow=True, startangle=90,
        colors=['lightgray', 'gray', 'dimgray', 'forestgreen'])
axs[1,1].axis('equal')
axs[1,1].set_title("Share of Total Sales in Dollars per Type",
fontsize=18)
axs[1,1].legend(csct.index, title="Clothing Type", loc="upper right")

plt.show()
```









Although skirts have the highest price, they are the least sold item, although not by a large margin. In terms of the amount of dollars generated by sales, skirts bring in the second highest numbers.

Trousers both sell the highest number of pieces and hold the highest position in terms of dollars generated, with a significantly greater number of sales and dollars seperating trousers from the next closest items.

Items on sale sell slightly more than blouses or skirts, but, due to the lower average price, generate the fewest dollars in sales.

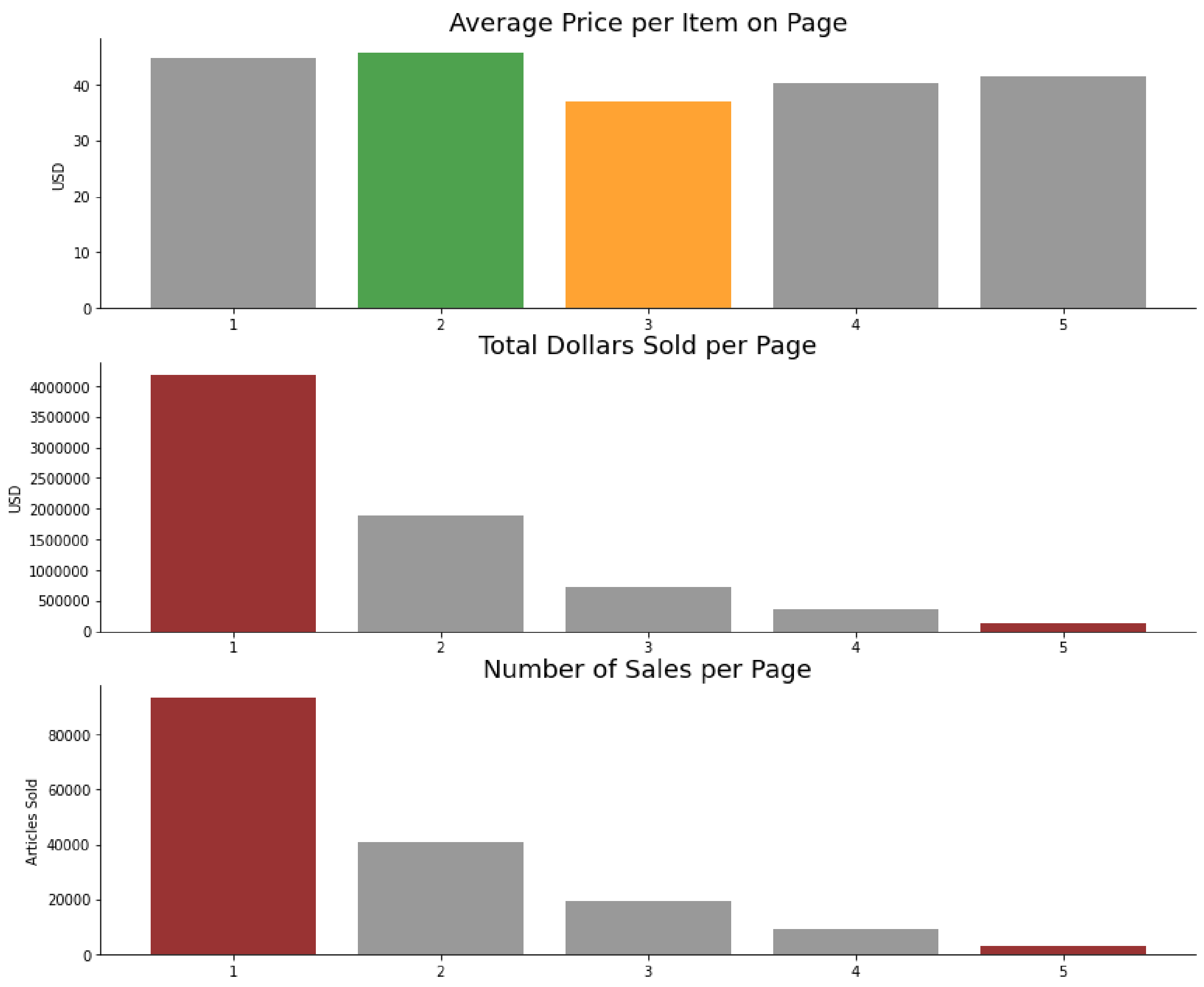
Price vs. Page

```
csdf.corr()
```

```
            Day     Price      Page
Day    1.000000 -0.002818  0.011125
Price -0.002818  1.000000 -0.150455
Page   0.011125 -0.150455  1.000000
```

The basic correlation does show a slightly negative relation between price and page - that is, the further back an item goes, the lower its price tends to be. Page 5 may not have as many items as pages 1 - 4, but that shouldn't have any real effect on price (it may have an effect on overall sales, though).

```
cspp = csdf[['Price', 'Page']]

ppavg = cspp.groupby('Page').mean()
pptot = cspp.groupby('Page').sum()
ppcnt = cspp.Page.value_counts()

ppdf = ppavg
ppdf['Total'] = pptot.Price
ppdf['Count'] = ppcnt
ppdf = ppdf.rename(columns={'Price':'Average'})

fig, axs = plt.subplots(nrows=3, ncols = 1, figsize=[14, 12])

axs[0].bar(ppdf.index, ppdf.Average,
           color=['gray', 'forestgreen', 'darkorange', 'gray',
'gray'], alpha=.8)
axs[0].set_title('Average Price per Item on Page', fontsize=18)
axs[0].set_ylabel('USD')

axs[1].bar(ppdf.index, ppdf.Total,
           color=['maroon', 'gray', 'gray', 'gray', 'maroon'],
alpha=.8)
axs[1].set_title('Total Dollars Sold per Page', fontsize=18)
axs[1].set_ylabel('USD')
axs[1].ticklabel_format(useOffset=False, style='plain')

axs[2].bar(ppdf.index, ppdf.Count,
           color=['maroon', 'gray', 'gray', 'gray', 'maroon'],
alpha=.8)
axs[2].set_title('Number of Sales per Page', fontsize=18)
axs[2].set_ylabel('Articles Sold')

for ax in axs:
    ax.yaxis.grid(False)
    ax.spines[['right', 'top']].set_visible(False)

plt.show()
```



Interestingly, pages 5 is the median page in terms of the average price of an item, with pages 1 and 2 having a higher average price per item, and pages 3 and 4 having a lower average price per item. Oddly, page 2, rather than page 1, has the highest average price per item, whereas page 3 has the lowest average price per item. Was this intentional?

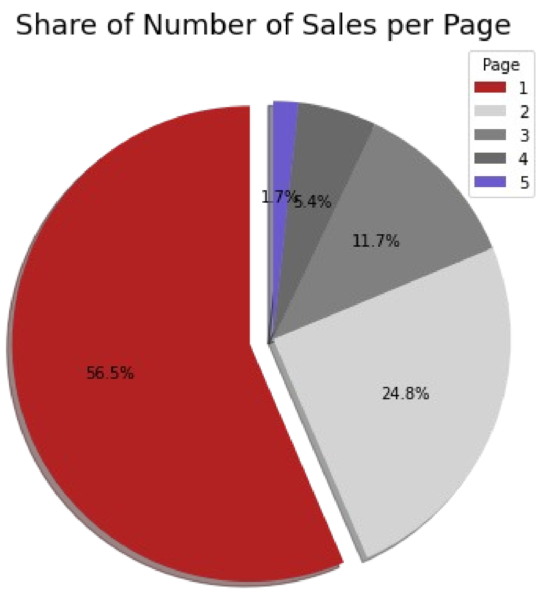
- `Is this meant to incentivize shoppers to buy on page 1, because page 2 is a little more expensive?`

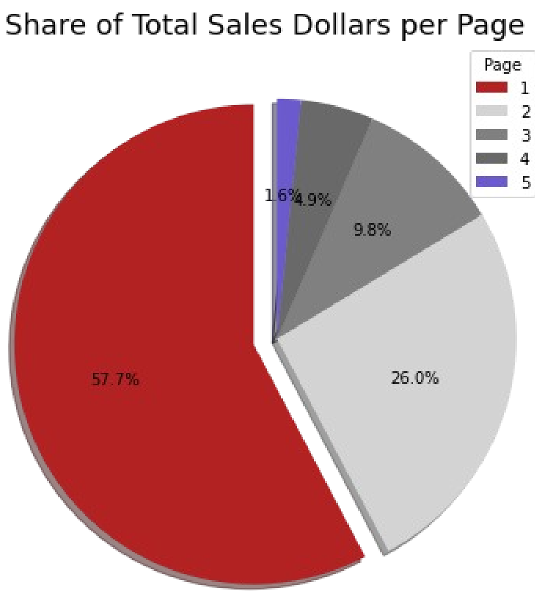
- `Should they jump on the bargains of page 3, before the prices go back up or they lose interest?`

Or, perhaps product placement is a little more arbitrary.

The revenue generated by page and the total number of sales by page seem to have a similar distribution - let's look a little closer...

```
fig, ax = plt.subplots(nrows=1, ncols=2, figsize=[14,7])
ax[0].pie(ppdf.Count, explode=(0.1, 0, 0, 0, 0), autopct='%1.1f%%',
        shadow=True, startangle=90,
        colors=['firebrick', 'lightgray', 'gray', 'dimgray',
'slateblue'])
ax[0].axis('equal')
ax[0].set_title("Share of Number of Sales per Page", fontsize=18)
ax[0].legend(ppdf.index, title="Page", loc="upper right")

ax[1].pie(ppdf.Total, explode=(0.1, 0, 0, 0, 0), autopct='%1.1f%%',
        shadow=True, startangle=90,
        colors=['firebrick', 'lightgray', 'gray', 'dimgray',
'slateblue'])
ax[1].axis('equal')
ax[1].set_title("Share of Total Sales Dollars per Page", fontsize=18)
ax[1].legend(ppdf.index, title="Page", loc="upper right")

plt.show()
```





Both the revenue and sales generated per page skew heavily towards page 1, with well over half of all sales coming from the first page, and over 80% of each coming from pages 1 and 2. Two factors that may contribute to this are:

- Customers see an item they like, and decide to purchase it rather than continue looking.
- The more popular an item is, the closer it moves to page 1, in order to maximize sales.

Conclusions

Data Findings

The data has several important findings:

- Sales peak in April, and decline slightly in following months; the data for August is incomplete, but the decline is present with and without the August data included.
- Overall, sales peak at the beginning of the month, and decline slightly over the course of the month. Once incomplete data and possible confounders are adjusted for, the sales trend seems to be more stable over the course of a month.
- Skirts have the highest average price of all goods, but trousers sell the highest number of units and generate the most dollars in sales by a significant margin. Sales items bring in the least money, but generate more sales than either blouses or skirts.
- Pages one and two have the highest priced items, and account for over 80% of all sales and dollars. Page 5 accounts for the fewest sales, but it is unknown if it has as many items shown as pages 1-4. Page 3 has the least expensive items of all pages.

Thank you!

Thank you for taking the time to read this Data Analysis.

I appreciate feedback - there's always room to improve, and a kind word goes a long way.