In [ ]:
```
# This Python 3 environment comes with many helpful analytics libraries inst
# It is defined by the kaggle/python Docker image: https://github.com/kaggle
# 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)
import matplotlib.pyplot as plt

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

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
# You can also write temporary files to /kaggle/temp/, but they won't be sav
```

To start, let's assign the dataframe to df.

In [ ]:
```
df = pd.read_csv("RecCen_Fall2022.csv")
print(df.head(20))
```

```
         Date      Time  Visits     Day
0     9/18/22      Time       0  Sunday
1   9/18/2022   6:00 AM       0  Sunday
2   9/18/2022   7:00 AM       0  Sunday
3   9/18/2022   8:00 AM       0  Sunday
4   9/18/2022   9:00 AM       0  Sunday
5   9/18/2022  10:00 AM       0  Sunday
6   9/18/2022  11:00 AM       0  Sunday
7   9/18/2022  12:00 PM     379  Sunday
8   9/18/2022   1:00 PM     281  Sunday
9   9/18/2022   2:00 PM     214  Sunday
10  9/18/2022   3:00 PM     220  Sunday
11  9/18/2022   4:00 PM     205  Sunday
12  9/18/2022   5:00 PM     151  Sunday
13  9/18/2022   6:00 PM     146  Sunday
14  9/18/2022   7:00 PM      91  Sunday
15  9/18/2022   8:00 PM       9  Sunday
16  9/18/2022   9:00 PM       0  Sunday
17  9/19/2022  10:00 PM      98  Sunday
18  9/19/2022   6:00 AM     126  Monday
19  9/19/2022   7:00 AM     133  Monday
```

As we can see, the first row of the dataframe isn't necessary. Furthermore, the columns are not properly aligned. Let's fix this.

In [ ]:
```
df['Date'], df['Visits'] = df['Date'].shift(), df['Visits'].shift() #correct
df = df.drop(df.index[[0,1]]).reset_index(drop=True) #remove invalid rows
print(df.head(20))
```

	Date	Time	Visits	Day
0	9/18/2022	7:00 AM	0.0	Sunday
1	9/18/2022	8:00 AM	0.0	Sunday
2	9/18/2022	9:00 AM	0.0	Sunday
3	9/18/2022	10:00 AM	0.0	Sunday
4	9/18/2022	11:00 AM	0.0	Sunday
5	9/18/2022	12:00 PM	0.0	Sunday
6	9/18/2022	1:00 PM	379.0	Sunday
7	9/18/2022	2:00 PM	281.0	Sunday
8	9/18/2022	3:00 PM	214.0	Sunday
9	9/18/2022	4:00 PM	220.0	Sunday
10	9/18/2022	5:00 PM	205.0	Sunday
11	9/18/2022	6:00 PM	151.0	Sunday
12	9/18/2022	7:00 PM	146.0	Sunday
13	9/18/2022	8:00 PM	91.0	Sunday
14	9/18/2022	9:00 PM	9.0	Sunday
15	9/18/2022	10:00 PM	0.0	Sunday
16	9/19/2022	6:00 AM	98.0	Monday
17	9/19/2022	7:00 AM	126.0	Monday
18	9/19/2022	8:00 AM	133.0	Monday
19	9/19/2022	9:00 AM	192.0	Monday

In order to more efficiently process and visualize our data, let's convert the values in the 'Date' and 'Time' columns to datetime objects. Let's also convert the values in 'Time' into their corresponding values on the 24-hour clock, and create an additional column called 'Datetime' that combines the values in 'Date' with the newly revised values in 'Time.'

In [ ]:
```
df['Date'] = pd.to_datetime(df['Date'], format='%m/%d/%Y')
df['Time'] = pd.to_datetime(df['Time'], format='%I:%M %p').dt.time
df['Time'] = df['Time'].apply(lambda x: x.strftime('%H:%M'))
df['Datetime'] = pd.to_datetime(df['Date'].astype(str) + ' ' + df['Time'].as
print(df.head(20))
```

	Date	Time	Visits	Day	Datetime
0	2022-09-18	07:00	0.0	Sunday	2022-09-18 07:00:00
1	2022-09-18	08:00	0.0	Sunday	2022-09-18 08:00:00
2	2022-09-18	09:00	0.0	Sunday	2022-09-18 09:00:00
3	2022-09-18	10:00	0.0	Sunday	2022-09-18 10:00:00
4	2022-09-18	11:00	0.0	Sunday	2022-09-18 11:00:00
5	2022-09-18	12:00	0.0	Sunday	2022-09-18 12:00:00
6	2022-09-18	13:00	379.0	Sunday	2022-09-18 13:00:00
7	2022-09-18	14:00	281.0	Sunday	2022-09-18 14:00:00
8	2022-09-18	15:00	214.0	Sunday	2022-09-18 15:00:00
9	2022-09-18	16:00	220.0	Sunday	2022-09-18 16:00:00
10	2022-09-18	17:00	205.0	Sunday	2022-09-18 17:00:00
11	2022-09-18	18:00	151.0	Sunday	2022-09-18 18:00:00
12	2022-09-18	19:00	146.0	Sunday	2022-09-18 19:00:00
13	2022-09-18	20:00	91.0	Sunday	2022-09-18 20:00:00
14	2022-09-18	21:00	9.0	Sunday	2022-09-18 21:00:00
15	2022-09-18	22:00	0.0	Sunday	2022-09-18 22:00:00
16	2022-09-19	06:00	98.0	Monday	2022-09-19 06:00:00
17	2022-09-19	07:00	126.0	Monday	2022-09-19 07:00:00
18	2022-09-19	08:00	133.0	Monday	2022-09-19 08:00:00
19	2022-09-19	09:00	192.0	Monday	2022-09-19 09:00:00

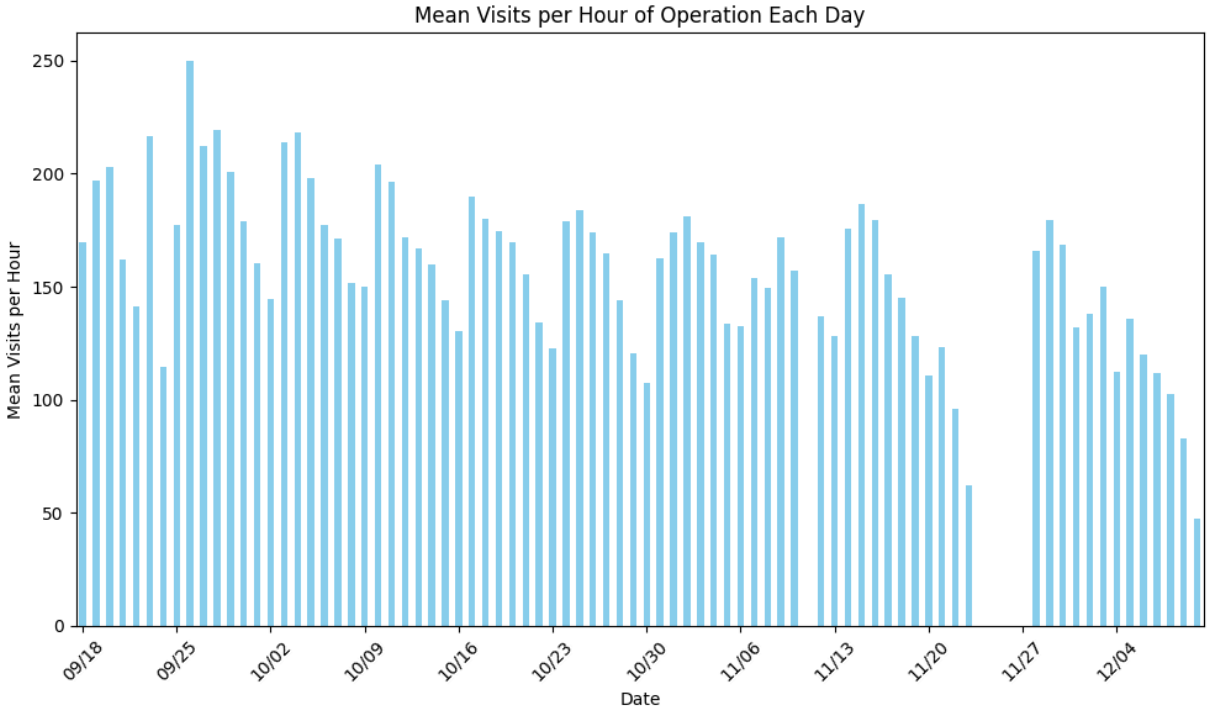
Now, let's remove any hours on days of operation during which the rec center was closed. This will help us better visualized how crowded it was each day depending not only on how many people swiped in but also how long the rec center was open.

In [ ]:
```
df = df[~(df['Datetime'] < '2022-09-18 12:00:00')]
df = df[~df['Datetime'].between('2022-11-23 15:00:00', '2022-11-23 22:00:00'
df = df[~(df['Datetime'] > '2022-12-10 19:00:00')]
df = df[~((df['Day'].isin(['Saturday', 'Sunday'])) & (df['Time'] < '09:00'))
df = df[~((df['Day'].isin(['Friday', 'Saturday'])) & (df['Time'] > '20:00'))
df = df[~((df['Day'] == 'Sunday') & (df['Time'] > '21:00'))]
print(df.head(20))
```

```
         Date   Time  Visits     Day            Datetime
5  2022-09-18  12:00     0.0  Sunday 2022-09-18 12:00:00
6  2022-09-18  13:00   379.0  Sunday 2022-09-18 13:00:00
7  2022-09-18  14:00   281.0  Sunday 2022-09-18 14:00:00
8  2022-09-18  15:00   214.0  Sunday 2022-09-18 15:00:00
9  2022-09-18  16:00   220.0  Sunday 2022-09-18 16:00:00
10 2022-09-18  17:00   205.0  Sunday 2022-09-18 17:00:00
11 2022-09-18  18:00   151.0  Sunday 2022-09-18 18:00:00
12 2022-09-18  19:00   146.0  Sunday 2022-09-18 19:00:00
13 2022-09-18  20:00    91.0  Sunday 2022-09-18 20:00:00
14 2022-09-18  21:00     9.0  Sunday 2022-09-18 21:00:00
16 2022-09-19  06:00    98.0  Monday 2022-09-19 06:00:00
17 2022-09-19  07:00   126.0  Monday 2022-09-19 07:00:00
18 2022-09-19  08:00   133.0  Monday 2022-09-19 08:00:00
19 2022-09-19  09:00   192.0  Monday 2022-09-19 09:00:00
20 2022-09-19  10:00   229.0  Monday 2022-09-19 10:00:00
21 2022-09-19  11:00   242.0  Monday 2022-09-19 11:00:00
22 2022-09-19  12:00   221.0  Monday 2022-09-19 12:00:00
23 2022-09-19  13:00   229.0  Monday 2022-09-19 13:00:00
24 2022-09-19  14:00   223.0  Monday 2022-09-19 14:00:00
25 2022-09-19  15:00   251.0  Monday 2022-09-19 15:00:00
```

It's time for our first visualization.

In [ ]:
```
# Plot the data with color-coded bars
visits_per_hour = df.groupby('Date')['Visits'].sum() / df.groupby('Date').si
plt.figure(figsize=(10, 6))
visits_per_hour.plot(kind='bar', color="skyblue")
plt.title('Mean Visits per Hour of Operation Each Day')
plt.xlabel('Date')
plt.ylabel('Mean Visits per Hour')
plt.xticks(range(0, len(visits_per_hour), 7), visits_per_hour.index[::7].str
plt.tight_layout()
plt.show()
```



A few key observations:

- Gym attendance started lower but peaked in week 1, then decreased each week until around the start of November, at which point it levelled out.
- Attendance is consistently higher at the start of the week (Monday, Tuesday, Wednesday) and lowest on weekends.
- Attendance was particularly low on days before breaks, which makes sense as some people head home before break is officially in session.

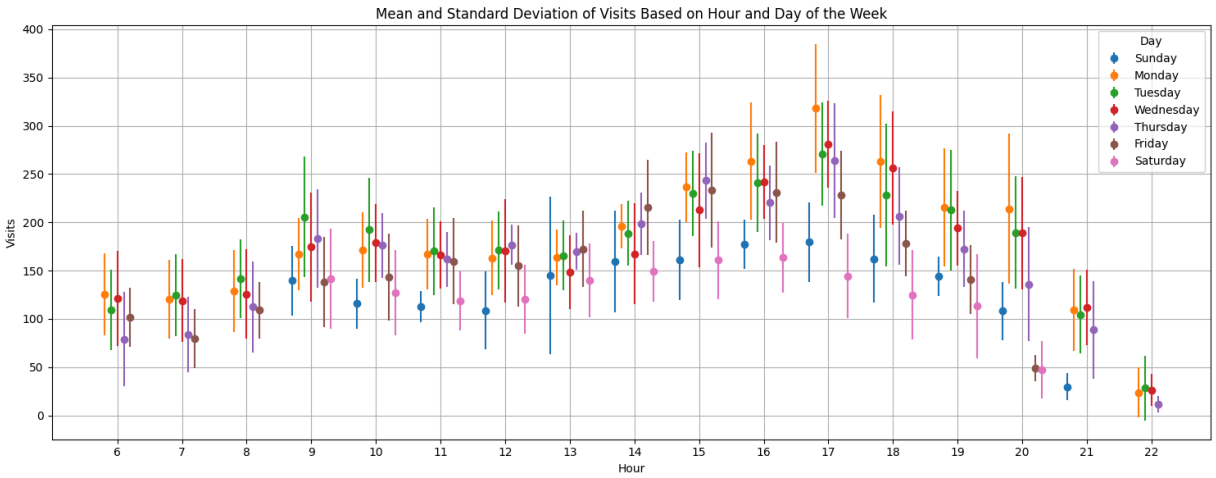
For the following visualizations, let's also remove days when the rec center was closed.

In [ ]:
```
df = df[~(df['Date'] == '2022-11-11')]
df = df[~df['Date'].between('2022-11-24', '2022-11-27')]
```

First, let's plot the mean and standard deviation of visits to the rec center across hours of the day and day of the week.

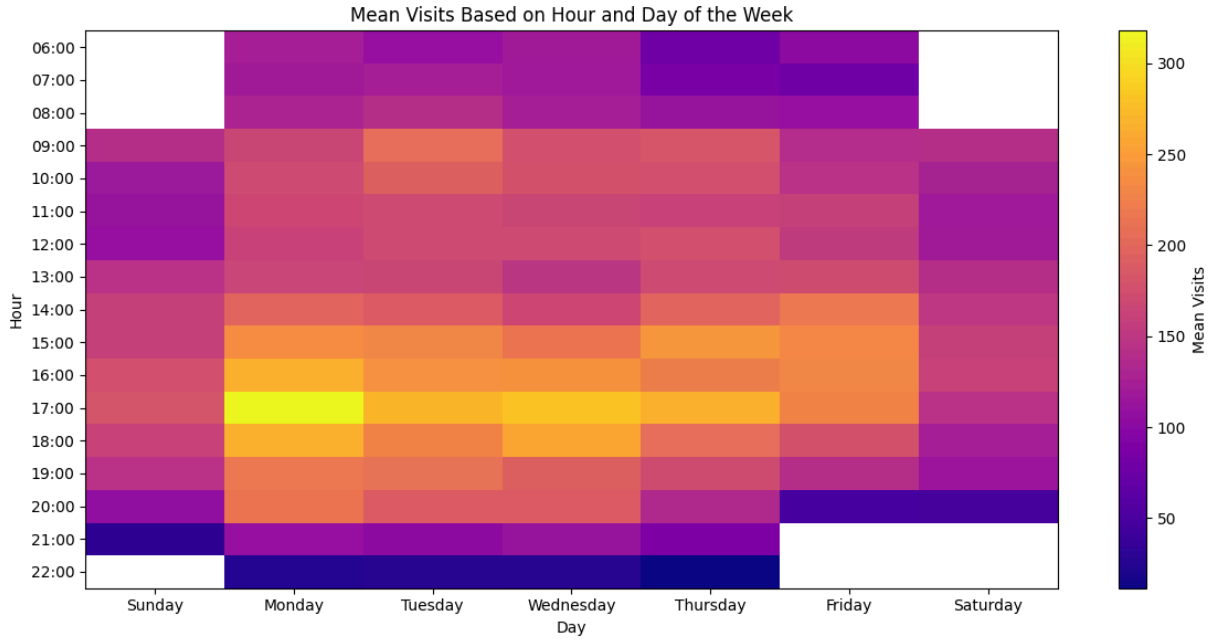
In [ ]:
```
mean_visits = df.groupby([df['Datetime'].dt.hour, 'Day'])['Visits'].mean().u
std_visits = df.groupby([df['Datetime'].dt.hour, 'Day'])['Visits'].std().uns
plt.figure(figsize=(15, 6))
days_of_week = ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Fri
for i, day in enumerate(days_of_week):
    x_values = mean_visits.index + i * 0.1 - 0.3 # Adjust the offset here
    plt.errorbar(x_values, mean_visits[day], yerr=std_visits[day], fmt='o',
plt.title('Mean and Standard Deviation of Visits Based on Hour and Day of th
plt.xlabel('Hour')
plt.ylabel('Visits')
plt.xticks(np.arange(6, 23))
plt.legend(title='Day')
plt.grid(True)
```

```
plt.tight_layout()
plt.show()
```



We can also create a heatmap to visualize the mean visits across hours and days of the week.

In [ ]:
```
df['Day'] = pd.Categorical(df['Day'], categories=days_of_week, ordered=True)
pivot_df = df.pivot_table(index='Time', columns='Day', values='Visits', aggf
plt.figure(figsize=(12, 6))
plt.imshow(pivot_df, cmap='plasma', aspect='auto', interpolation='nearest')
plt.colorbar(label='Mean Visits')
plt.xticks(range(len(pivot_df.columns)), pivot_df.columns)
plt.yticks(range(len(pivot_df.index)), pivot_df.index)
plt.xlabel('Day')
plt.ylabel('Hour')
plt.title('Mean Visits Based on Hour and Day of the Week')
plt.tight_layout()
plt.show()
```



A few key observations from the previous two graphs:

- The most visits to the rec center occur at Monday between 5:00-6:00pm.
- Late afternoon / early evening (roughly 3:00pm-6:00pm) is consistently the busiest time throughout the week.
- On each day, the fewest visits occur in the last hour of operation. Granted, this does not mean it is the least busiest hour, as people could have swiped in earlier and stayed in the rec center during this final hour.