User Rating Calculation

Using the User Application dataset, we will answer the following questions:

1) **Clean the 'applications.csv' data set using Pandas DataFrame:**
   - Remove duplicates based on 'applicant_id'.
   - Fill in missing values in the 'External Rating' field with zeros.
   - Fill in missing values in the 'Education level' field with the text "Середня".
   - Add industry ratings data from the 'industries.csv' file to this DataFrame.
2) **Calculate the application rating based on the following conditions:**
   - The rating is the sum of scores for the application across 6 criteria and must be a number from 0 to 100
   - The rating is zero if the 'Amount' value is missing or 'External Rating' equals zero.

   The rating is composed of the following components:
   - 20 points are added to the rating if the applicant's age is between 35 and 55
   - 20 points are added to the rating if the application was submitted on a weekday
   - 20 points are added to the rating if the applicant is married
   - 10 points are added to the rating if the applicant is located in Kyiv or the Kyiv region
   - 'Score' value from the 'industries.csv' table is also added to the application (ranging from 0 to 20 points).
   - 20 points are added to the rating if 'External Rating' is greater than or equal to 7
   - 20 points are subtracted from the rating if 'External Rating' is less than or equal to 2
3) **Applications are considered accepted if the rating is greater than zero**
4) **Group the data from the resulting table by the submission week, and plot the average rating of accepted applications for each week on a graph**

Python

```python
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import matplotlib.dates as mdates
```

```python
# load data
applications = pd.read_csv('C:/Users/plish/Desktop/Python_HW/User Rating
Calculation/applications.csv')
industries = pd.read_csv('C:/Users/plish/Desktop/Python_HW/User Rating
Calculation/industries.csv')
```

```python
#industries.info()
applications.info() # check data structure
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 13315 entries, 0 to 13314
Data columns (total 10 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   Applied at       13315 non-null  object
 1   Amount           13272 non-null  float64
 2   Age              13315 non-null  int64
 3   Gender           13315 non-null  object
 4   Industry         13315 non-null  object
 5   Marital status   13315 non-null  object
 6   External Rating  13243 non-null  float64
 7   Education level  13282 non-null  object
 8   Location         11540 non-null  object
 9   applicant_id     13315 non-null  object
dtypes: float64(2), int64(1), object(7)
memory usage: 1.0+ MB
```

```
# duplicates remove
applications.drop_duplicates('applicant_id', inplace = True)

# Filling Nan values
applications['External Rating'].fillna(0, inplace = True)
applications['Education level'].fillna('Середня', inplace = True)

# joining two data sets
full_df = pd.merge(applications, industries, on = 'Industry', how = 'left')
```

```
# calculate the application rating
full_df['Applied at'] = pd.to_datetime(full_df['Applied at'], format = 'mixed') # change
column format from string to date
```

```
# age score column add
full_df['Age Score'] = full_df['Age'].apply(lambda x : 20 if x >= 35 and x <=55 else 0)
#full_df['Age Score'] = full_df['Age'].between(35, 55).astype(int)*20 # second way to find
age score
```

```
# create weekdays list
weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```

```
# get day names
full_df['Applied Day'] = full_df['Applied at'].apply(lambda x : x.day_name())
```

```
# score for weekdays application
full_df['Apply Score'] = (full_df['Applied Day'].isin(weekdays))*20
# second way to find the score
#full_df['Apply Score']= full_df['Applied at'].apply(lambda x: 20 if x in weekdays else 0)
```

```
# marital score
full_df['Marital Score'] = full_df['Marital status'].apply(lambda x : 20 if x == 'Married'
else 0)
```

```
# location score
full_df['Location Score'] = full_df['Location'].apply(lambda x : 10 if x == 'Київ чи
область' else 0)
```

```
# external score
full_df['External Score'] = full_df['External Rating'].apply(lambda x : 20 if x >= 7 else
(-20 if x <= 2 else 0)).astype(int)
```

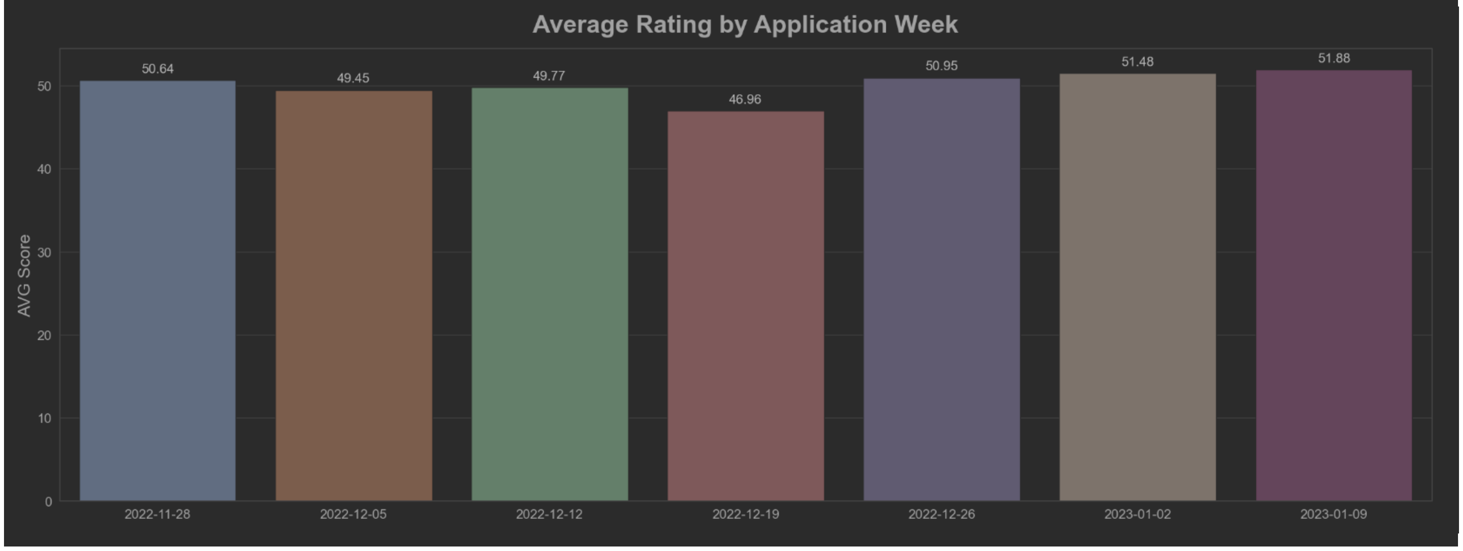
```
full_df['Total Score'] = full_df[['Score', 'Apply Score', 'Age Score', 'Marital Score',
'Location Score', 'External Score']].sum(axis = 1)
full_df['Total Score'] = full_df['Total Score'].apply(lambda x : 0 if x <0 else (100 if x
> 100 else x)) # adjust rating to be on scale from 0 to 100
full_df.loc[full_df['External Rating'] == 0, ['Total Score']] = 0 # adjust rating Total
Score = 0 when External Rating = 0
full_df.loc[full_df['Amount'].isnull(), ['Total Score']] = 0 # adjust Total Score = 0 when
Amount is Nun
```

```
# resulting table with the accepted applications (rating > 0)
full_df.drop(columns = ['Score', 'Age Score', 'Apply Score', 'Marital Score', 'Location
Score', 'External Score', 'Applied Day'], inplace = True)

accepted = full_df.loc[full_df['Total Score'] > 0, :]
```

```
accepted['Application Week'] = accepted['Applied at'].apply(lambda x : x.week) #
application week number
accepted['Week Start'] = accepted['Applied at'].dt.to_period('W').apply(lambda y:
y.start_time) # date value of the week start
```

```
# avg rating grouped by submission week
data = round(accepted.groupby('Week Start')['Total Score'].mean().reset_index(), 2)
```

```
# bar chart
data['Week Start'] = data['Week Start'].astype(str)

plt.rcParams['figure.figsize'] = [30, 10]
sns.set(style = "whitegrid")

ax = sns.barplot(x = 'Week Start', y = 'Total Score', data = data)

plt.title('Average Rating by Application Week', fontdict={'fontsize': 30, 'fontweight':
'bold'}, pad = 20)
plt.xlabel(None, fontsize= 'xx-large')
plt.ylabel('AVG Score', fontsize='xx-large')

# Display values on each bar
for p in ax.patches:
   ax.annotate(f'{(p.get_height()):.2f}', (p.get_x() + p.get_width() / 2.,
p.get_height()),
               ha='center', va='center', fontsize=16, color='black', xytext=(0, 14),
               textcoords='offset points')
ax.tick_params(axis= 'both', which='major', labelsize=16)

plt.show()
```


Average Rating by Application Week
AVG Score