```
# LINK TO VIDEO SHOWING THE RUN OF THIS FILE, INCLUDING THE GRAPHS :
https://drive.google.com/file/d/1RCKuXKMNL9kEhOPESbGZmRW4wyl_rQY7/
view?usp=sharing

# imports here
import pandas as pd
import plotly.express as px
import numpy as np
import matplotlib.pyplot as plt
import plotly.graph_objects as go

# read data
data_path = 'weather.csv'
data = pd.read_csv(data_path)
print(data.head())

# Convert Kelvin to Fahrenheit, gonna need this
data['Ftemp'] = (data['Ktemp'] - 273.15) * 9/5 + 32

# gotta make sure the time is in datetime format
data['time'] = pd.to_datetime(data['time'])

# need year and month for analysis
data['year'] = data['time'].dt.year
data['month'] = data['time'].dt.month

# calculating the average monthly temp
monthly_avg = data.groupby(['year', 'month'])
['Ftemp'].mean().reset_index()
print(monthly_avg.head())

# plot the yearly curve, need matplotlib for this
def plot_curve(year):
    yearly_data = monthly_avg[monthly_avg['year'] == year]
    plt.figure(figsize=(10, 6))
    plt.plot(yearly_data['month'], yearly_data['Ftemp'], marker='o',
linestyle='-', color='blue')
    plt.title(f'Avg Monthly Temps in {year}')
    plt.xlabel('Month')
    plt.ylabel('Avg Temp (°F)')
    plt.xticks(range(1, 13), ['Jan', 'Feb', 'Mar', 'Apr', 'May',
'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec'])
    plt.grid(True)
    plt.show()

# show plot for a year
plot_curve(1960)

# now for the interactive plot with Plotly, really cool
def interactive_year_plot(df):
    fig = go.Figure()
    for year in df['year'].unique():
        yearly_df = df[df['year'] == year]
        fig.add_trace(go.Scatter(x=yearly_df['month'],
y=yearly_df['Ftemp'],
                                 mode='lines+markers',
                                 name=str(year),
                                 visible=False))  # hide initially

    # first year visible by default
    fig.data[0].visible = True

    # setting up the slider thing
    steps = []
    for i, year in enumerate(df['year'].unique()):
        step = dict(
            method="update",
            args=[{"visible": [False] * len(df['year'].unique())},
                  {"title": f'Avg Monthly Temps in {year}'}],
            label=str(year)
        )
        step["args"][0]["visible"][i] = True
        steps.append(step)

    sliders = [dict(
        active=0,
        currentvalue={"prefix": "Year: "},
        pad={"t": 50},
        steps=steps
    )]

    fig.update_layout(
        sliders=sliders,
        title="Avg Monthly Temps",
        xaxis_title="Month",
        yaxis_title="Avg Temp (°F)",
        xaxis=dict(tickmode='array',
                   tickvals=list(range(1, 13)),
                   ticktext=['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun',
                             'Jul', 'Aug', 'Sep', 'Oct', 'Nov',
'Dec'])
    )

    return fig

# showing the plot for the dataframe
interactive_year_plot(monthly_avg).show()

# Part B starts
print(monthly_avg)
```

```
# calculate yearly avg temp
yearly_avg_temp = monthly_avg[["year",
"Ftemp"]].groupby("year").mean()

# find first warm year
first_warm_year_index = np.where(yearly_avg_temp > 55.0)[0][0]
first_warm_year = yearly_avg_temp.index[first_warm_year_index]
print("First year with avg temp over 55 degrees:", first_warm_year)

# Part C, getting creative here
# seasons for months
def get_season(month):
    if month in [12, 1, 2]:
        return 'Winter'
    elif month in [3, 4, 5]:
        return 'Spring'
    elif month in [6, 7, 8]:
        return 'Summer'
    elif month in [9, 10, 11]:
        return 'Autumn'

# apply seasons
monthly_avg['season'] = monthly_avg['month'].apply(get_season)
# adjust year for December
monthly_avg['season_year'] = monthly_avg.apply(lambda row: row['year']
+ 1 if row['month'] == 12 else row['year'], axis=1)
# avg seasonal temp
seasonal_avg = monthly_avg.groupby(['season_year', 'season'])
['Ftemp'].mean().reset_index()
seasonal_avg_pivot = seasonal_avg.pivot(index='season_year',
columns='season', values='Ftemp')
print(seasonal_avg_pivot.head())

# plot the seasonal trends
def plot_trends(df):
    seasons = ['Winter', 'Spring', 'Summer', 'Autumn']
    fig = go.Figure()
    for season in seasons:
        fig.add_trace(go.Scatter(x=df.index, y=df[season],
mode='lines+markers', name=season))
    fig.update_layout(title='Seasonal Temp Trends Over Years',
                      xaxis_title='Year',
                      yaxis_title='Avg Temp (°F)',
                      legend_title='Season',
                      xaxis=dict(tickmode='linear'),
                      template='plotly_white')
    return fig

plot_trends(seasonal_avg_pivot).show()
```