# Homework 08

## ⚠ Before you start ⚠

*Duplicate this Jupyter Notebook in your `week-10` folder (right-click -> Duplicate) and then add your last name to the beginning of it (ie. `blevins-hw-08.ipynb` - otherwise you risk having all your work overwritten when you try to sync your GitHub repository with your instructor's repository.*

---

## Overview

In this assignment, you'll synthesize some of the Python skills you've learned over the past month or so, including Pandas and Plotly. You'll be analyzing the opening of new businesses in Colorado during the 1940s.

Draw on the following tutorials:

- 📚 Walsh, Pandas Basics Part 1
- 📚 Walsh, Pandas Basics Part 2
- 📚 Walsh, Pandas Basics Part 3
- 🐼 Pandas Concepts
- 📚 Introduction to Plotly
- 📚 Cleaning Excel Files

## The Data

First, get the necessary data files from our shared course repository:

- Open GitHub Desktop and select your course repository ( `lastname-sp25-data-materials` )
- Click `Fetch origin` to check for updates
- Go to `Branch` → `Merge into current branch` → select `upstream/main` -> `Merge`
- Click `Push origin` to sync everything up
- Launch Jupyter Lab and navigate to the `week-10` folder

You should see a single Excel file that you will be working with: `co-new-businesses-1940s.xlsx`. Inside that Excel file, there are two separate sheets: `New CO Businesses` and `Cities 1940`.

- `New CO Businesses`: This is a subset of new businesses that were established in Colorado during the 1940s - a subset of data drawn from this database.
- `Cities 1940`: this contains population statistics for Colorado cities in the 1940 Census.

## Import Libraries and Load Data

- Import the necessary libraries:
  - pandas (using the alias `pd`)
  - plotly.express (using the alias `px`)

In [7]:
```
import pandas as pd
import plotly.express as px
```

- Load both sheets from the Excel file:
  - Create a variable called `businesses_df` to store the "New CO Businesses" sheet in the Excel file
  - Create a variable called `cities_df` to store the "Cities 1940" sheet in the Excel file
  - Use `pd.read_excel()` with the appropriate parameters

In [9]:
```
businesses_df = pd.read_excel('co-new-businesses-1940s.xlsx', sheet_name='Ne
cities_df = pd.read_excel('co-new-businesses-1940s.xlsx', sheet_name='Cities
```

## Familiarize Yourself with the Data

Familiarize yourself with the data:

- Display a sample of 10 rows from each dataframe.
- Check the data types for the columns in each dataframe

In [11]:
```
print(businesses_df.sample(10))
print(cities_df.sample(10))

print(businesses_df.columns)
print(cities_df.columns)
```

```
        entityid                                Business entity name  \
97   19871050689          CHRISTIAN FIDELITY LIFE INSURANCE COMPANY
518  19871110768                        BONNIE BRAE BAPTIST CHURCH
344  19871001208                              ST. MARY CREDIT UNION
792  19871193754  NATIONAL ASSOCIATION OF INSURANCE WOMEN (INTER...
138  19871009642  THE MEDICAL PROTECTIVE COMPANY, Delinquent May...
862  19871277982                     GREAT WESTERN INORGANICS, INC.
133  19871008589             SOUTHERN STATES LIFE INSURANCE COMPANY
13   19871030120  HALLIBURTON ENERGY SERVICES, INC., Delinquent ...
317  19461110447     MEMORIAL HOSPITAL, Dissolved December 29, 1972
297  19871152159  THE UNITED PRESBYTERIAN CHURCH OF TRINIDAD, CO...

                      Address        city state  zip_code Country  \
97   2721 North Central Avenue     Phoenix    AZ       NaN      US
518       700 BONNIE BRAE BLVD      DENVER    CO   80209.0      US
344                116 E 6thSt  Walsenburg    CO   81089.0      US
792   1 Glenlake Pkwy Ste 1200     Atlanta    GA   30328.0      US
138                        NaN         NaN   NaN       NaN     NaN
862           17400 Highway 72      Arvada    CO   80007.0      US
133                        NaN         NaN   NaN       NaN     NaN
13   3000 N Sam Houston Pkwy E     Houston    TX   77032.0      US
317                        NaN         NaN   NaN       NaN     NaN
297           801 SAN PEDRO ST    TRINIDAD    CO   81082.0      US

    date_entity_formed  year_entity_formed
97          1945-09-07                1945
518         1946-05-16                1946
344         1943-01-09                1943
792         1942-01-30                1942
138         1947-07-02                1947
862         1946-11-13                1946
133         1946-08-15                1946
13          1945-04-28                1945
317         1946-03-20                1946
297         1946-04-05                1946
                   city  year  total population
208               walden  1940               668
6                  aspen  1940               777
100  hot sulphur springs  1940               235
22           buena vista  1940               779
129                lyons  1940               654
174            red cliff  1940               715
179           rocky ford  1940              3494
141          monte vista  1940              3208
33              collbran  1940               301
0                  akron  1940              1417
Index(['entityid', 'Business entity name', 'Address', 'city', 'state',
       'zip_code', 'Country', 'date_entity_formed', 'year_entity_formed'],
      dtype='object')
Index(['city', 'year', 'total population'], dtype='object')
```

# Data Cleaning and Preparation

## Cleaning column names

For both datasets, you want to clean and standardize the column names (headers):

- Change column names to all lowercase
- Replace any whitespace with an underscore ( `_` ) - ex. `some column` becomes `some_column`
- *Hint: Use* `str.lower()` *and* `str.replace()`
- Show the first 10 rows of your dataframe to make sure it worked

In [13]:
```
businesses_df.columns = [col.strip().lower().replace(' ', '_') for col in bu
cities_df.columns = [col.strip().lower().replace(' ', '_') for col in cities

print(businesses_df.head(10))
print(cities_df.head(10))
```

```
      entityid                                business_entity_name  \
0  19871004753                                ALAMOSA CREDIT UNION
1  19871241137   THE UNITED METHODIST CHURCH OF STEAMBOAT SPRINGS
2  19871275274              ALLIED JEWISH FEDERATION OF COLORADO
3  19871127721         Iglesia CRISTO REY + Christ the King, ELCA
4  19871117433     LYNCH-COTTEN POST NO. 190, THE AMERICAN LEGION
5  19871105155          THE BEAR RIVER VALLEY FARMERS COOPERATIVE
6  19871162072                               Belmar Baptist Church
7  19871110810        Bethel Lutheran Church of Windsor, Colorado
8  19871116977  BLACKINTON AND DECKER, INC., Delinquent Novemb...
9  19871113871                                BOW-MAR OWNERS, INC.

               address               city state  zip_code country  \
0         2437 MAIN ST            ALAMOSA    CO   81101.0      US
1           736 OAK ST  STEAMBOAT SPRINGS    CO   80487.0      US
2    300 S. Dahlia St.             DENVER    CO   80246.0      US
3     2300 S Patton Ct             Denver    CO   80219.0      US
4       425 Highway 92           Crawford    CO   81415.0      US
5  193 E Jefferson Ave             Hayden    CO   81639.0      US
6     460 S Kipling St           Lakewood    CO   80226.0      US
7        328 Walnut St            Windsor    CO   80550.0      US
8            424 LIPAN             DENVER    CO   80204.0      US
9    5380 Lakeshore Dr          Littleton    CO   80123.0      US

  date_entity_formed  year_entity_formed
0         1945-03-27                1945
1         1949-03-09                1949
2         1946-05-23                1946
3         1945-09-06                1945
4         1949-12-30                1949
5         1940-03-22                1940
6         1943-08-20                1943
7         1946-05-24                1946
8         1946-05-20                1946
9         1948-01-12                1948
       city  year  total_population
0     akron  1940              1417
1   alamosa  1940              5613
2      alma  1940               469
3  antonito  1940              1220
4    arriba  1940               286
5    arvada  1940              1482
6     aspen  1940               777
7    aurora  1940              3437
8    basalt  1940               212
9  bayfield  1940               372
```

## Standardize and clean data for cities

- Standardize city names in the business data so that it **removes any trailing or leading whitespace** and **changes the values to all lowercase** (hint: use `.str.strip()` and `.str.lower()`)
- Show the first 10 rows of your dataframe to make sure it worked

***I wanted it to be title case so I did that***

In [16]:
```
businesses_df['city'] = businesses_df['city'].str.strip().str.title()
businesses_df.head(10)
```

Out[16]:

| | entityid | business_entity_name | address | city | state | zip_code | countr |
|---|---|---|---|---|---|---|---|
| **0** | 19871004753 | ALAMOSA CREDIT UNION | 2437 MAIN ST | Alamosa | CO | 81101.0 | U |
| **1** | 19871241137 | THE UNITED METHODIST CHURCH OF STEAMBOAT SPRINGS | 736 OAK ST | Steamboat Springs | CO | 80487.0 | U |
| **2** | 19871275274 | ALLIED JEWISH FEDERATION OF COLORADO | 300 S. Dahlia St. | Denver | CO | 80246.0 | U |
| **3** | 19871127721 | Iglesia CRISTO REY + Christ the King, ELCA | 2300 S Patton Ct | Denver | CO | 80219.0 | U |
| **4** | 19871117433 | LYNCH-COTTEN POST NO. 190, THE AMERICAN LEGION | 425 Highway 92 | Crawford | CO | 81415.0 | U |
| **5** | 19871105155 | THE BEAR RIVER VALLEY FARMERS COOPERATIVE | 193 E Jefferson Ave | Hayden | CO | 81639.0 | U |
| **6** | 19871162072 | Belmar Baptist Church | 460 S Kipling St | Lakewood | CO | 80226.0 | U |
| **7** | 19871110810 | Bethel Lutheran Church of Windsor, Colorado | 328 Walnut St | Windsor | CO | 80550.0 | U |
| **8** | 19871116977 | BLACKINTON AND DECKER, INC., Delinquent Novemb... | 424 LIPAN | Denver | CO | 80204.0 | U |
| **9** | 19871113871 | BOW-MAR OWNERS, INC. | 5380 Lakeshore Dr | Littleton | CO | 80123.0 | U |

In [17]:
```
#realizing I need to make the other one title case for merging purposes late

cities_df['city'] = cities_df['city'].str.strip().str.title()
cities_df.head(10)
```

Out[17]:

| | city | year | total_population |
|---|---|---|---|
| 0 | Akron | 1940 | 1417 |
| 1 | Alamosa | 1940 | 5613 |
| 2 | Alma | 1940 | 469 |
| 3 | Antonito | 1940 | 1220 |
| 4 | Arriba | 1940 | 286 |
| 5 | Arvada | 1940 | 1482 |
| 6 | Aspen | 1940 | 777 |
| 7 | Aurora | 1940 | 3437 |
| 8 | Basalt | 1940 | 212 |
| 9 | Bayfield | 1940 | 372 |

# Categorize Cities

## Define your function

Create a function called `categorize_city_size` that does the following:

- Takes in a number that corresponds to the population for a city and returns the following based on the size of the city:
  - `Small Town` if population is less than 1,000
  - `Medium Town` if population is between 1,000 to 5,000
  - `Large Town` if population is between 5,000 to 20,000
  - `City` if population greater than or equal to 20,000

## Test Your Function

Test out the function on a single number ( `2,000` ) to make sure it returns `Medium Town`

*note below that I interpreted the question to be about inputting a city rather than population number, and when I realized the difference I figured it was good to figure out the inputting a city thing anyway*

In [21]:
```
cities_df.loc[cities_df['city'] == 'Akron', 'total_population'].values[0]
```

Out[21]: 1417

In [22]:
```
def categorize_city_size(city):
    citypop = cities_df.loc[cities_df['city'] == city, 'total_population'].v
    if citypop < 1000:
```

```
        return 'Small Town'
    elif citypop >= 1000 and citypop < 5000:
        return 'Medium Town'
    elif citypop >= 5000 and citypop < 20000:
        return 'Large Town'
    elif citypop >= 20000:
        return 'City'

categorize_city_size('Akron')
```

Out[22]: 'Medium Town'

### Apply the function

- Take your `cities_df` dataframe and add a new column called `city_category` that applies your function to the `total_population` column of the dataframe.
- *Hint: use* `apply()`
- Show the first 10 rows of your dataframe to make sure it worked

In [24]:
```
cities_df['city_category'] = cities_df['city'].apply(categorize_city_size)

cities_df.head(10)
```

Out[24]:

| | city | year | total_population | city_category |
|---|---|---|---|---|
| **0** | Akron | 1940 | 1417 | Medium Town |
| **1** | Alamosa | 1940 | 5613 | Large Town |
| **2** | Alma | 1940 | 469 | Small Town |
| **3** | Antonito | 1940 | 1220 | Medium Town |
| **4** | Arriba | 1940 | 286 | Small Town |
| **5** | Arvada | 1940 | 1482 | Medium Town |
| **6** | Aspen | 1940 | 777 | Small Town |
| **7** | Aurora | 1940 | 3437 | Medium Town |
| **8** | Basalt | 1940 | 212 | Small Town |
| **9** | Bayfield | 1940 | 372 | Small Town |

## Analyze Businesses by Year

Let's take a look at how many new businesses were formed in Colorado in each year during the 1940s:

### Calculate new businesses by year

Create a variable called `businesses_per_year` by:

- Counting the number of new businesses based on `year_entity_formed`
- *Hint: use* `value_counts()` *and* `reset_index()`
- Show the first 10 rows of your dataframe

In [26]:
```
businesses_per_year = businesses_df['year_entity_formed'].value_counts().res
businesses_per_year.head(10)
```

Out[26]:

| | year_entity_formed | count |
|---|---|---|
| **0** | 1947 | 161 |
| **1** | 1948 | 156 |
| **2** | 1946 | 153 |
| **3** | 1949 | 133 |
| **4** | 1945 | 87 |
| **5** | 1940 | 72 |
| **6** | 1941 | 69 |
| **7** | 1943 | 47 |
| **8** | 1944 | 43 |
| **9** | 1942 | 35 |

## Visualize new businesses by year

Create a bar chart using Plotly Express showing new businesses per year:

- Set x-axis to the year
- Set y-axis to the number of new businesses
- Add an appropriate title and labels
- Display text on each bar
- Hint: Use `px.bar()`

In [28]:
```
businesses_per_year_fig = px.bar(businesses_per_year, x='year_entity_formed'
                                 labels={'year_entity_formed': 'Year', 'count
                              )

businesses_per_year_fig.update_layout(
    title_text='New Businesses in Colorado in the 1940s', title_x=0.5,
    xaxis_title='Year Formed',
    yaxis_title='Number of New Businesses',
    xaxis_tickangle=-45,
    height=500,
    width=800,
    title_font=dict(size=22),
    plot_bgcolor='white',
    margin=dict(l=40, r=40, t=80, b=80),
```

```
    hoverlabel=dict(
        bgcolor="white",
        font_size=12,
        font_family="Arial")
)

businesses_per_year_fig.update_xaxes(type='category', categoryorder='categor

businesses_per_year_fig.show()
```

## Analyze Businesses by City

Let's take a look at how many new businesses were formed in each Colorado city during the 1940s:

## Calculate number of new businesses by city

Create a new variable called `city_businesses` that contains:

- A dataframe with counts of the number of new businesses in each city
- *Hint: Use `value_counts()` and `reset_index()`*
- Show the first 10 rows of your dataframe

In [30]:
```
city_businesses = businesses_df['city'].value_counts().reset_index()
city_businesses.head(10)
```

Out[30]:

| | city | count |
|---|---|---|
| 0 | Denver | 152 |
| 1 | Colorado Springs | 34 |
| 2 | Lakewood | 22 |
| 3 | Pueblo | 20 |
| 4 | Arvada | 14 |
| 5 | Grand Junction | 14 |
| 6 | Fort Collins | 13 |
| 7 | Greeley | 13 |
| 8 | Centennial | 12 |
| 9 | Englewood | 12 |

## Visualize new businesses by city

Create a bar chart with Plotly Express showing the top 10 cities with the most new businesses created during the 1940s:

- Filter to only show the top 10 cities (hint: use `.head()`)
- Set x-axis to `city`
- Set y-axis to `count`
- Add an appropriate title and labels

In [32]:
```
city_businesses_fig = px.bar(city_businesses.head(10), x='city', y='count',
                                labels={'city': 'City', 'count': 'Number of
                            )

city_businesses_fig.update_layout(
    title_text='New Businesses in Colorado in the 1940s by City', title_x=0.
    xaxis_title='City',
    yaxis_title='Number of New Businesses',
    xaxis_tickangle=-45,
    height=500,
    width=800,
    title_font=dict(size=22),
    plot_bgcolor='white',
    margin=dict(l=40, r=40, t=80, b=80),
    hoverlabel=dict(
        bgcolor="white",
        font_size=12,
        font_family="Arial")
```

```
)

city_businesses_fig.show()
```

# Combine Business and City Data

We have two datasets, both of which contain information about Colorado cities. Let's combine the two into a single dataframe that contains both information about new businesses and their population in the 1940 census.

## Merge dataframes

Merge the two dataframes together:

- Create a new variable called `merged_df`
- Use `pd.merge()` on the `city_businesses` and `cities_df` dataframes
- Figure out which column is shared between the two to use as your "key" to merge them
- ⚠ **Note: use the `how='inner'` parameter for your merge**
- Show the first 10 rows of your new dataframe

In [34]:
```
merged_df = pd.merge(
    city_businesses,
    cities_df,
    on='city',
    how='inner'
)

merged_df.head(10)
```

Out[34]:

| | city | count | year | total_population | city_category |
|---|---|---|---|---|---|
| **0** | Denver | 152 | 1940 | 322412 | City |
| **1** | Colorado Springs | 34 | 1940 | 36789 | City |
| **2** | Pueblo | 20 | 1940 | 52162 | City |
| **3** | Arvada | 14 | 1940 | 1482 | Medium Town |
| **4** | Grand Junction | 14 | 1940 | 12479 | Large Town |
| **5** | Fort Collins | 13 | 1940 | 12251 | Large Town |
| **6** | Greeley | 13 | 1940 | 15995 | Large Town |
| **7** | Englewood | 12 | 1940 | 9680 | Large Town |
| **8** | Littleton | 11 | 1940 | 2244 | Medium Town |
| **9** | Aurora | 10 | 1940 | 3437 | Medium Town |

## Filter out missing values

You'll note that several rows of data contain `NaN` or missing values - this means that there was a city listed in the businesses dataframe but it didn't have a corresponding match in the population dataframe. For now, remove these from the `merged_df` dataframe:

- Filter out rows where `total_population` is NaN
- *Hint: use a filter + `.notna()`*

In [36]:
```
merged_df['total_population'].notna().value_counts()
```

Out[36]:
```
total_population
True    113
Name: count, dtype: int64
```

In [37]:
```
city_businesses.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 243 entries, 0 to 242
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   city    243 non-null    object
 1   count   243 non-null    int64
dtypes: int64(1), object(1)
memory usage: 3.9+ KB
```

## Calculate new businesses on a per capita rate

To make it easier to compare larger cities with smaller cities, you're going to calculate a new column for each city: the number of new businesses per 1,000 residents.

- Add a new column to `merged_df` called `biz_per_thousand` that is filled with:
  - A calculation dividing the `count` column by the `total_population` column and multiplying by 1,000
- Sort the merged dataframe by `biz_per_thousand` in descending order
- Show the first 10 rows of the dataframe to check if it worked

In [39]:
```
merged_df['biz_per_thousand'] = (merged_df['count']/merged_df['total_populat
merged_df = merged_df.sort_values(by='biz_per_thousand', ascending=False)
merged_df.head(10)
```

Out[39]:

| | city | count | year | total_population | city_category | biz_per_thousand |
|---|---|---|---|---|---|---|
| **92** | Green Mountain Falls | 1 | 1940 | 87 | Small Town | 11.494253 |
| **36** | Keenesburg | 3 | 1940 | 284 | Small Town | 10.563380 |
| **59** | Bennett | 2 | 1940 | 199 | Small Town | 10.050251 |
| **3** | Arvada | 14 | 1940 | 1482 | Medium Town | 9.446694 |
| **52** | Morrison | 2 | 1940 | 216 | Small Town | 9.259259 |
| **20** | Castle Rock | 5 | 1940 | 580 | Small Town | 8.620690 |
| **33** | Woodland Park | 3 | 1940 | 372 | Small Town | 8.064516 |
| **54** | Granby | 2 | 1940 | 251 | Small Town | 7.968127 |
| **79** | Grover | 1 | 1940 | 137 | Small Town | 7.299270 |
| **96** | Timnath | 1 | 1940 | 147 | Small Town | 6.802721 |

# Visualize new business creation by city

Let's say we want to see the cities with the highest *rate* of business creation (ie. new businesses per thousand residents)

- Create a bar chart in Plotly of `merged_df`:
  - Filter to only show the top 10 cities (use `.head(10)`)
  - Set x-axis to `city`
  - Set y-axis to `biz_per_thousand`
  - Use `city_category` for color
  - Add an appropriate title and labels

In [41]:
```
biz_per_thousand_fig = px.bar(merged_df.head(10), x='city', y='biz_per_thous
                                labels={'city': 'City', 'biz_per_thousand':
                              )

biz_per_thousand_fig.update_layout(
    title_text='Colorado Cities With Highest Rate of Business Creation: 1940
    xaxis_title='City',
    yaxis_title='Number of New Businesses Per Thousand Residents',
    xaxis_tickangle=-45,
    height=500,
    width=800,
    showlegend=True,
    legend_title_text='City Category',
    title_font=dict(size=22),
    plot_bgcolor='white',
    margin=dict(l=40, r=40, t=80, b=80),
    hoverlabel=dict(
        bgcolor="white",
        font_size=12,
        font_family="Arial")
)

biz_per_thousand_fig.update_xaxes(categoryorder='total descending')

biz_per_thousand_fig.show()
```

# Bonus: New businesses by city category

Let's say we want to compare different size categories to see whether new businesses were cropping up in smaller places or bigger cities.

## Create a new dataframe

First, you'll need to create a new dataframe that consists of four rows, with each row a different category of city containing the total number of businesses created within that category of city.

- Create a new dataframe called `city_category_totals`
- Start with `merged_df`
- Group by `city_category`
- Add up ( `sum()` ) the `count` column
- Use `.reset_index()`

In [43]:
```
city_category_totals = merged_df.groupby('city_category').sum('count').reset
city_category_totals.head()
```

Out[43]:

| | city_category | count | year | total_population | biz_per_thousand |
|---|---|---|---|---|---|
| **0** | City | 206 | 5820 | 411363 | 1.779057 |
| **1** | Large Town | 97 | 25220 | 121943 | 10.205952 |
| **2** | Medium Town | 156 | 83420 | 99517 | 77.352727 |
| **3** | Small Town | 82 | 104760 | 27963 | 205.226406 |

## Visualize businesses by city category

- Create a pie chart in Plotly:
  - Use `px.pie()` with appropriate parameters
  - Use `city_category_totals` as your dataframe
  - Use `count` for your values
  - Use `city_category` for your names
  - Add an appropriate title and labels

In [45]:

```
category_fig = px.pie(city_category_totals, values='count', names='city_cate
                      labels={'city_category': 'City Category', 'count': 'Nu
                            )

category_fig.update_layout(
    title_text='Business Growth By City Size in Colorado: 1940s', title_x=0.
    showlegend=True,
    legend_title_text='City Category',
    width=800,
    height=600
)

category_fig.show()
```

## Bonus Challenge: Create a Scatterplot

Create a scatter plot in Plotly showing:

- The relationship between city population (x-axis) and new businesses (y-axis)
- Only data for towns with a population of 2,000 or more people.
- Dots sized according to the number of new businesses in that city
- Dots colored according to their size category

In [47]:
```
bigger_pop_filter = merged_df['total_population'] >= 2000
scatter_df = merged_df[bigger_pop_filter]

scatter_fig = px.scatter(scatter_df, x='total_population', y='count', title=
                         color='city_category',
                         size='count',
                         hover_data=['city'],
                         labels={'city': 'City', 'total_population': 'Popula
```

```
scatter_fig.update_layout(
    title_text='New Businesses in Colorado: 1940s', title_x=0.5,
    showlegend=True,
    legend_title_text='City Category',
    width=1300,
    height=700
)

scatter_fig.show()
```

# Submission Guidelines

- Run all code cells and make sure it is outputting without errors
- Submit both the notebook file (.ipynb) and a PDF export of your notebook on Canvas
- Note: the PDF probably won't display the Plotly figures - that's okay