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.

Rayce Loveland

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 [8]:
```
#Your code here
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 [10]:
```
#Your code here
businesses_df = pd.read_excel('co-new-businesses-1940s.xlsx', sheet_name='New CO Bu
cities_df = pd.read_excel('co-new-businesses-1940s.xlsx', sheet_name='Cities 1940')
```

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 [12]:
```
#Your code here
businesses_df.sample(10)
```

Out[12]:

	entityid	Business entity name	Address	city	state	zip_code	Country	date
629	19871113920	CAPRA, DE CANIO, GIOIA, TEZAK, POST NO. 6616 V...	4300 PECOS ST	DENVER	CO	80211.0	US	
12	19871117117	CACHE LA POUDRE GRANGE NUMBER 456	2929 N Co Rd 23	BELLVUE	CO	80512.0	US	
525	19871111011	DENVER MOTOR FINANCE CO., INC., Delinquent Dec...	1515 ARAPAHOE ST STE 1200	DENVER	CO	80202.0	US	
872	19871104982	Journey Church at Windsor	8075 County Road 72	Windsor	CO	80550.0	US	
169	19871036373	CALIFORNIA INSURANCE COMPANY	NaN	NaN	NaN	NaN	NaN	
338	19491117200	FOUR CORNERS URANIUM CORPORATION, Dissolved Ja...	NaN	NaN	NaN	NaN	NaN	
76	19871012906	Esurance Insurance Company of New Jersey	3100 Sanders Rd, Suite 201	Northbrook	IL	60062.0	US	
837	19871325937	RED ROCKS CONGREGATION OF JEHOVAH'S WITNESSES,...	4287 S Eldridge St	Morrison	CO	80465.0	US	
357	19871008920	AM INTERNATIONAL, INC., Colorado Authority Ter...	431 LAKEVIEW COURT	MOUNT PROSPECT	IL	60056.0	US	
128	19871007949	LINCOLN DIRECT LIFE INSURANCE CO., Delinquent ...	NaN	NaN	NaN	NaN	NaN	

In [13]:
```
cities_df.sample(10)
```

Out[13]:

	city	year	total population
121	las animas	1940	3232
88	grover	1940	137
25	carbondale	1940	437
156	orchard city	1940	865
22	buena vista	1940	779
1	alamosa	1940	5613
154	olney springs	1940	260
95	hillrose	1940	177
144	monument	1940	175
186	seibert	1940	249

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 [15]:
```
#Your code here
businesses_df.columns = businesses_df.columns.str.lower().str.replace(' ', '_')
cities_df.columns = cities_df.columns.str.lower().str.replace(' ', '_')
```

In [16]:
```
businesses_df.head(10)
```

Out[16]:

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

In [17]:
```
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

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

In [19]:

```
#Your code here
businesses_df['city'] = businesses_df['city'].str.strip().str.lower()
businesses_df.head(10)
```

Out[19]:

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

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

In [21]:
```
def categorize_city_size(population):
    if population < 1000:
        return 'Small Town'
    elif 1000 <= population < 5000:
        return 'Medium Town'
    elif 5000 <= population < 20000:
        return 'Large Town'
    else:
        return 'City'
```

Test Your Function

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

In [23]:
```
#Your code here
categorize_city_size(2000)
```

Out[23]: `'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 [25]:
```
#Your code here
cities_df['city_category'] = cities_df['total_population'].apply(categorize_city_si
cities_df.head(10)
```

Out[25]:

	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 [27]:
```
#Your code here
businesses_per_year = businesses_df['year_entity_formed'].value_counts().reset_inde
businesses_per_year.head(10)
```

Out[27]:

	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 [29]:
```
#Your code here
```

```
fig = px.bar(
    businesses_per_year,
    x='year_entity_formed',
    y='count',
    text='count',
    title='New Colorado Businesses by Year (1940s)',
    labels={
        'year_entity_formed': 'Year',
        'count': 'Number of New Businesses'
    }
)

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 [31]:
```
#Your code here
city_businesses = businesses_df['city'].value_counts().reset_index()
city_businesses.head(10)
```

Out[31]:

	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 [33]:
```
#Your code here
top_10_cities = city_businesses.head(10)
```

In [34]:
```
fig = px.bar(
    top_10_cities,
    x='city',
    y='count',
    text='count',
    title='Top 10 Colorado Cities by New Business Formation (1940s)',
    labels={
```

```
        'city': 'City',
        'count': 'Number of New 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 [36]:
```
#Your code here
merged_df = pd.merge(city_businesses, cities_df, on='city', how='inner')
merged_df.head(10)
```

Out[36]:

	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 [38]:
```
#Your code here
merged_df = merged_df[merged_df['total_population'].notna()]
merged_df.head(10)
```

Out[38]:

	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

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 [40]:

```
#Your code here
merged_df['biz_per_thousand'] = (merged_df['count'] / merged_df['total_population']
merged_df = merged_df.sort_values('biz_per_thousand', ascending=False)
merged_df.head(10)
```

Out[40]:

	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 [42]:
```
#Your code here
top_10_biz_rate = merged_df.head(10)
```

In [43]:
```
fig = px.bar(
    top_10_biz_rate,
    x='city',
    y='biz_per_thousand',
    color='city_category',
    text='biz_per_thousand',
    title='Top 10 Colorado Cities by Business Creation Rate (1940s)',
    labels={
        'city': 'City',
        'biz_per_thousand': 'New Businesses per 1,000 Residents',
        'city_category': 'City Category'
    }
)

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 [45]:
```
#Your code here
```

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 [47]:
```
#Your code here
```

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 [49]:
```
#Your code here
```

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