Extraction, Transformation, and Load Technical Report

<Point of Living>

Rachael Munyua, Lingzi Xiaoli, Susan Pan

1. INTRODUCTION

1.1 Background

Point of Living is a globally recognized magazine based in the United States. The mission of Point of Living (POL) is to inspire readers in making a positive impact in their lives. One of the main events for the magazine is to publish the city rankings in terms of economic health and living conditions.
The objective of this project is to identify the best cities to live in in the United States.

1.2. Client Request

POL is interested in compiling a database of United States cities, which will include most of the indicators of economic status and living conditions. By querying through the database, POL will be able to calculate the city scores in terms of “best cities to raise a family”, “ best cities to start a career”, “best cities for retirement” to provide valuable information to different customers (e.g. young professionals, real estate investors) who are willing to explore different cities.
Parameters for this demo database includes: Employment rate | Income | Rental rates | Weather

- ❖ Sources:
- U.S. Bureau of Labor Statistics
    - www.bls.gov/web/metro/ssamatab1.txt
- Bureau of Economic Analysis
    - https://www.bea.gov/data/income-saving/personal-income-county-metro-and-other-areas
- Apartment List
    - www.apartmentlist.com/rentonomics/rental-price-data/
- NOAA (National Weather For Environmental Information)
    - https://www.ncdc.noaa.gov/cdo-web/webservices/v2

1.3 Technologies and resource contributions

In efforts to obtain the necessary data, the team utilized API & web scraping methodologies and Python, Pandas, Excel, ERD, SQLAlchemy technologies.

- Lingzi Xiaoli: Data source research (five non-economic factors)| Data retrieving and clean up (API weather) |Data import and manipulation example| Final report write-up (2.1, 2.2, 2.3)
- Rachael Munyua: Database creation | Repository creation and maintenance | Each Section Discussion/Brainstorming | Proofreading final report | Final report write-up (1.1, 1.2, 1.3, 1.4)
- Susan Pan: Data source research (five economic factors) | Data clean-up (economic tables) | ERD Graph | Data import and manipulation example | Final report write-up (2.4, 2.5, 2.6)

1.4 Definitions, Acronyms and Abbreviations

APL - Application Programming Interface
CSV - Comma-Separated Values
ETL - Extract, Transform and Load
ERD - Entity-relationship diagram
MSA - Metropolitan Statistical Area
NOAA - National Oceanic and Atmospheric Administration
TMAX - Maximum
TMIN - Minimum
TAVG - Average
SQL - Structured Query Language

2. ETL DETAILS

2.1 Data Import/Extract Sources and Method

The original **unemployment dataset** covered the unemployment rates and ranks for 389 cities in metropolitan statistical cities in the United States. A csv table was retrieved from the U.S. Bureau of Labor Statistics through **web scraping** (https://www.bls.gov/web/metro/laummtrk.htm#laummtrk.f.p).

The original **weather dataset** covered the minimum temperatures (TMIN), maximum temperatures (TMAX), and average temperatures (TAVG) for all weather stations in every metropolitan city shared with the unemployment dataset. We used the following parameters: datatypeid (TMAX, TMIN, TAVG), startdate (2019-01-01), enddate (2019-12-31), units (standard), limit (1000), offset (for loop to exhaust). Json files were retrieved from the National Oceanic and Atmospheric Administration (NOAA) using **API** with email request for token as the permission. The basic URL is https://www.ncdc.noaa.gov/cdo-web/api/v2/data.

The original **apartment rent dataset** covered the rental prices for four different bedroom types (studio, 1 bedroom (br), 2 br, 3br, 4br) for every month of year 2014 to 2020 April in 660 U.S. cities. A csv table was downloaded directly from the Apartment list (https://www.apartmentlist.com/rentonomics/rental-price-data/).

The original **income** dataset covered per capita personal income in two consecutive years (2018, 2019) for 384 U.S. cities in metropolitan statistical areas. A csv table was downloaded directly from the Bureau of Economic Analysis (BETA), U.S. Department of Commerce.
(https://www.bea.gov/news/archive?field_related_product_target_id=All&created_1=All&title=)

2.2 Data Acquisition

The **unemployment dataset** used in building our database covered the unemployment rates in March of 2020 for 389 cities in the metropolitan Area in the United States. The **income dataset** covered the Per capita personal income in 383 U.S. metropolitan cities in 2018. The **apartment rent dataset** covered rental prices for four different bedroom types (studio, 1 bedroom (br), 2 br, 3br, 4br) for every month of year 2019 in 660 U.S. cities. The **weather dataset** covered average temperatures of 348 cities shared with our Unemployment dataset for each month in year 2019.

The **unemployment and apartment rent** datasets are going to be dynamic based on the global economic and political environments, new industry shifts and trends as well as each city's long-term plan, therefore the frequency of updating them is aimed to be **monthly.** The **personal income** is also going to be relatively static, so the updating frequency will be **annually**. The weather dataset is going to update **monthly**. Therefore, our dataset will be able to provide the monthly update per request, and the overall comprehensive update will be annually.

The data acquisition resources and manipulation tools (including API and web scraping) will be provided. The unemployment/income/apartment rent data can be approached through periodically checking the authoritative websites, while weather data can be accessed through specifying the new start date and end date. For data manipulation/updating, the client needs to have python and excel as the prerequisites if they would like to curate the database by themselves, or we can provide services to update the database per their detailed requirements.

2.3 Data Transform

For the unemployment dataset, we dropped N/As and separated the city/area into two columns for city and state. At the same time, we generated a city list named as City_MSA as a linkage table and the city reference for weather searches.

For the apartment rent dataset, we only kept the most recent info of the year 2019 and 2020 for our demo database, and dropped data from 2014-2018.

For the income dataset, we imported personal income information for year 2018 and 2019 and dropped historical data before 2018.

For the weather data, we first found the shared cities between the City_MSA and city list from NOAA, extracted TAVG for all weather stations in a specific city, then calculated the mean temperature for that month. Further examination revealed four cities (Hilton Head Island, SC; Hammond, LA; Owensboro, KY; Elizabethtown, KY) did not have any temperature record for the year 2019, and six cities (Gadsden, AL; Chico, CA; Ocala, FL; Bloomsburg, PA; Sumter, SC; Lynchburg, VA) had missed a temperature value for one or two months, so we filled in N/A for them. Where uploading all the csv datasets, we changed the N/A as NaN for SQL to recognize.

Example of data clean-up (split “Metropolitan area” into “City/Area” and “State”)

```
new = df2["Metropolitan area"].str.split(", ", n = 1, expand = True)
df2["City/Area"] = new[0]
new2 = new[1].str.split(" ", n = 1, expand = True)
df2["State"] = new2[0]
df2
```

	Metropolitan area	Unemployment Rate	Rank	Year	Month	City/Area	State
2	Kahului-Wailuku-Lahaina, HI Metropolitan Stati...	2.1	1	2020	3	Kahului-Wailuku-Lahaina	HI
3	Urban Honolulu, HI Metropolitan Statistical Area	2.1	1	2020	3	Urban Honolulu	HI
4	Ames, IA Metropolitan Statistical Area	2.2	3	2020	3	Ames	IA
5	Ann Arbor, MI Metropolitan Statistical Area	2.4	4	2020	3	Ann Arbor	MI
6	Idaho Falls, ID Metropolitan Statistical Area	2.4	4	2020	3	Idaho Falls	ID
...	...	...	...	...	...	...	...
386	Hanford-Corcoran, CA Metropolitan Statistical ...	12.3	385	2020	3	Hanford-Corcoran	CA
387	Merced, CA Metropolitan Statistical Area	12.9	386	2020	3	Merced	CA
388	Visalia-Porterville, CA Metropolitan Statistic...	14.5	387	2020	3	Visalia-Porterville	CA
389	Yuma, AZ Metropolitan Statistical Area	14.8	388	2020	3	Yuma	AZ
390	El Centro, CA Metropolitan Statistical Area	20.5	389	2020	3	El Centro	CA

Example of data clean-up (find the shared cities in City_MSA and NOAA city list) :

```
# all US cities from noaa
df = pd.DataFrame.from_dict(city_list)
df.info()
df.to_csv("all_cities.csv")
# shared cities from both Employment and Temperature datasets
city_list_msa = pd.read_csv("City_MSA_list.csv")[["Unified State", "Unified City"]]
shared_cities=[]
diff_cities=[]
for s1, c1 in city_list_msa.values.tolist():
    found = 0
    for id, cname in city_list:
        c2 = cname.split(',')[0]
        s2 = cname.split(' ')[-2]
        if s1 == s2 and c1 == c2:
            shared_cities.append([id, c2, s2])
            found = 1
            break
    if found != 1:
        diff_cities.append([s1, c1])

df = pd.DataFrame.from_dict(shared_cities)
df.info()
df.to_csv("shared_cities.csv")
df = pd.DataFrame.from_dict(diff_cities)
df.info()
df.to_csv("diff_cities.csv")
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 845 entries, 0 to 844
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       845 non-null    object
 1   1       845 non-null    object
dtypes: object(2)
memory usage: 13.3+ KB
Salt Lake City
Salt Lake City
UT
UT
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 348 entries, 0 to 347
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       348 non-null    object
 1   1       348 non-null    object
 2   2       348 non-null    object
dtypes: object(3)
memory usage: 8.3+ KB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41 entries, 0 to 40
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       41 non-null     object
 1   1       41 non-null     object
dtypes: object(2)
memory usage: 784.0+ bytes
```

2.4 Data Integrity & Update Frequency

Please refer to Image 2.4.1 for the database's ERD.

The unemployment_rate data is from the U.S. Bureau of Labor Statistics. This data is updated by month, and current month data will be released by the end of next month, the data table can be updated accordingly.

The personal_income data comes from the U.S. Bureau of Economic Analysis, which is released annually, the data table can be updated accordingly.

The rent_rate table should be updated monthly from the source of www.apartmentlist.com.

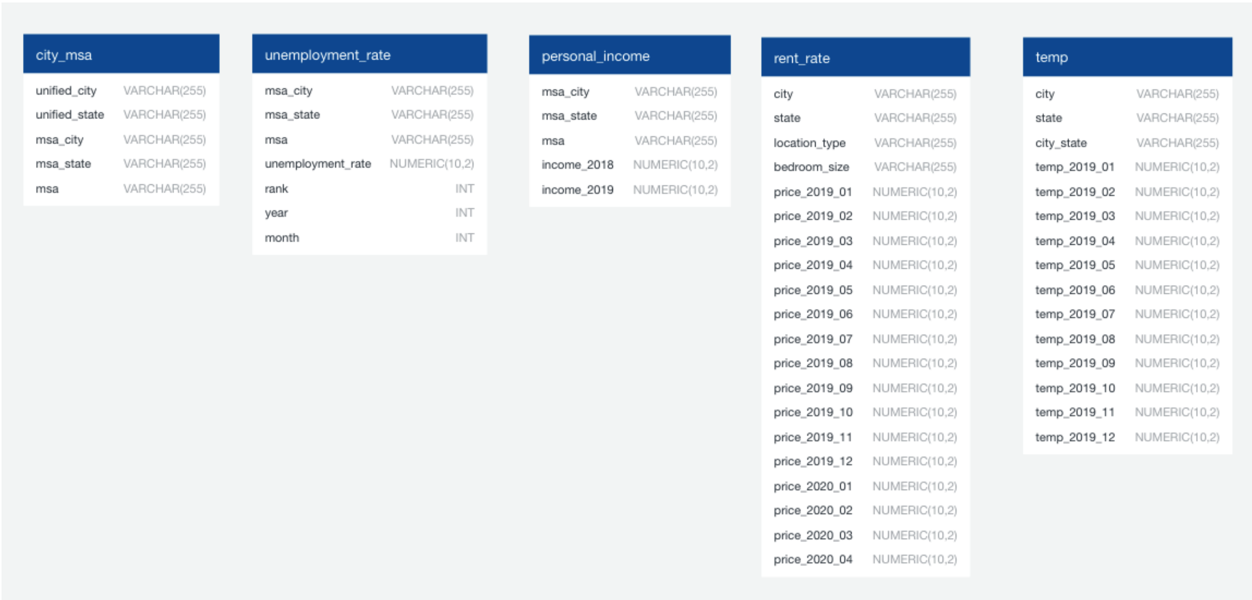
The temp date should be updated by month through API query to NOAA.

Both unemployment_rate data and personal_income data are surveyed at MSA* level, so we created the city_msa table to make the linkage between two geographic systems.

All data tables can be queried/joined by "city" and "state". But each data source has a slight difference in the location pool. So we didn't set up the primary key and forgegin key system. But this won't impact the demo database's usage.

**MSA: (metropolitan statistical area) a geographical region with a relatively high population density at its core and close economic ties throughout the area.*

Image 2.4.1



2.5 Data Loading and Availability

This database is built on Amazon Web Service, the client can access it by logging in their account and querying the data through the platform SQL tools.

The client can export the data as a CSV file for data manipulation, or they can import the data directly to data manipulation tools like Python for further exploration.

For future use, this database should continue importing datasets like university list by city, crime data by city and so on to enrich the content and serve its purpose. The client can eventually query the data and use their self-developed formula to calculate scores for each city in terms of living environment, economic potential.

Example of data loading into Python:

```
employment_data = pd.read_sql("SELECT c.unified_city, c.unified_state, u.unemployment_rate \
                              FROM city_msa AS C, unemployment_rate AS u WHERE c.msa_city = u.msa_city \
                              AND C.msa_state = u.msa_state;", conn)
employment_data
```

	unified_city	unified_state	unemployment_rate
0	Fairbanks	AK	5.0
1	Anchorage	AK	5.2
2	Anniston	AL	3.9
3	Auburn	AL	3.0
4	Birmingham	AL	3.0
...	...	...	...
384	Morgantown	WV	4.3
385	Charleston	WV	5.9
386	Beckley	WV	6.7
387	Cheyenne	WY	3.8
388	Casper	WY	4.5

Example of data calculation:

```
rent_data = pd.read_sql("SELECT city, state, bedroom_size, price_2019_01, price_2019_02, price_2019_03, \
                         price_2019_04, price_2019_05, price_2019_06, price_2019_07, \
                         price_2019_08, price_2019_09, price_2019_10, price_2019_11, price_2019_12 FROM rent_rate", conn)
```

```
rent_data["2019_avg_rent"] = rent_data.mean(axis=1)
grouped_rent_data = rent_data.groupby(["city","state"]).sum()
grouped_rent_data["2019_rent_per_room"] = grouped_rent_data["2019_avg_rent"]/11
rent_per_room = grouped_rent_data["2019_rent_per_room"].reset_index()
rent_per_room
```

	city	state	2019_rent_per_room
0	Abilene	TX	423.325758
1	Acworth	GA	542.628788
2	Adelanto	CA	654.333333
3	Aiken	SC	440.287879
4	Albany	NY	511.090909
...	...	...	...
654	Yakima	WA	447.659091
655	Youngstown	OH	347.659091
656	Ypsilanti	MI	538.560606
657	Yucca Valley	CA	436.272727
658	Zion	IL	513.871212

2.6 Data Application Demo

For this section, we will use the data currently available in the demo database to show a simple calculation of “best cities in the US to make money”.

Calculate the employment score:

```
employment_data = pd.read_sql("SELECT c.unified_city AS city, c.unified_state AS state, u.unemployment_rate \
                              FROM city_msa AS C, unemployment_rate AS u WHERE c.msa_city = u.msa_city \
                              AND C.msa_state = u.msa_state;", conn)
```

```
employment_data["employment score"] = employment_data["unemployment_rate"].max() - employment_data["unemployment_rate"]
employment_data
```

	city	state	unemployment_rate	employment score
0	Fairbanks	AK	5.0	15.5
1	Anchorage	AK	5.2	15.3
2	Anniston	AL	3.9	16.6
3	Auburn	AL	3.0	17.5
4	Birmingham	AL	3.0	17.5
...	...	...	...	...
384	Morgantown	WV	4.3	16.2
385	Charleston	WV	5.9	14.6
386	Beckley	WV	6.7	13.8
387	Cheyenne	WY	3.8	16.7
388	Casper	WY	4.5	16.0

389 rows × 4 columns

Calculate the income score:

```
rent_data = pd.read_sql("SELECT city, state, bedroom_size, price_2019_01, price_2019_02, price_2019_03, \
                        price_2019_04, price_2019_05, price_2019_06, price_2019_07, \
                        price_2019_08, price_2019_09, price_2019_10, price_2019_11, price_2019_12 FROM rent_rate", conn)
```

```
rent_data["2019_avg_rent"] = rent_data.mean(axis=1)
grouped_rent_data = rent_data.groupby(["city","state"]).sum()
grouped_rent_data["2019_rent_per_room"] = grouped_rent_data["2019_avg_rent"]/11
rent_per_room = grouped_rent_data["2019_rent_per_room"].reset_index()
rent_per_room
```

	city	state	2019_rent_per_room
0	Abilene	TX	423.325758
1	Acworth	GA	542.628788
2	Adelanto	CA	654.333333
3	Aiken	SC	440.287879
4	Albany	NY	511.090909
...	...	...	...
654	Yakima	WA	447.659091
655	Youngstown	OH	347.659091
656	Ypsilanti	MI	538.560606
657	Yucca Valley	CA	436.272727
658	Zion	IL	513.871212

```python
pi_data = pd.read_sql("SELECT c.unified_city AS city, c.unified_state AS state, p.income_2019 \
                      FROM city_msa AS c, personal_income AS p \
                      WHERE c.msa_city = p.msa_city \
                      AND c.msa_state = p.msa_state;", conn)
pi_data["income_2019_monthly"] = pi_data["income_2019"]/12
```

```python
merge_pi_rent = pd.merge(rent_per_room, pi_data, on=["city","state"])
merge_pi_rent["monthly_flexible_income"] = merge_pi_rent["income_2019_monthly"] - merge_pi_rent["2019_rent_per_room"]
```

```python
merge_pi_rent["income score"] = merge_pi_rent["monthly_flexible_income"]/100
merge_pi_rent
```

	city	state	2019_rent_per_room	income_2019	income_2019_monthly	monthly_flexible_income	income score
0	Abilene	TX	423.325758	44730.0	3727.500000	3304.174242	33.041742
1	Albany	NY	511.090909	60557.0	5046.416667	4535.325758	45.353258
2	Albuquerque	NM	454.537879	43770.0	3647.500000	3192.962121	31.929621
3	Amarillo	TX	404.265152	47769.0	3980.750000	3576.484848	35.764848
4	Ames	IA	498.515152	45150.0	3762.500000	3263.984848	32.639848
...	...	...	...	...	...	...	...
171	Wichita	KS	361.848485	54666.0	4555.500000	4193.651515	41.936515
172	Williamsport	PA	350.674242	45876.0	3823.000000	3472.325758	34.723258
173	Worcester	MA	589.606061	57939.0	4828.250000	4238.643939	42.386439
174	Yakima	WA	447.659091	45014.0	3751.166667	3303.507576	33.035076
175	Youngstown	OH	347.659091	44032.0	3669.333333	3321.674242	33.216742

Merge employment score with income score and calculate the total score

```python
total_score = merge_pi_rent_unemployment[["city", "state", "income score", "employment score"]]
total_score["total_score"] = total_score["income score"]+total_score["employment score"]
total_score
```

	city	state	income score	employment score	total_score
0	Abilene	TX	33.041742	16.5	49.541742
1	Albany	NY	45.353258	16.4	61.753258
2	Albuquerque	NM	31.929621	15.2	47.129621
3	Amarillo	TX	35.764848	17.1	52.864848
4	Ames	IA	32.639848	18.3	50.939848
...	...	...	...	...	...
171	Wichita	KS	41.936515	17.0	58.936515
172	Williamsport	PA	34.723258	13.1	47.823258
173	Worcester	MA	42.386439	16.9	59.286439
174	Yakima	WA	33.035076	13.7	46.735076
175	Youngstown	OH	33.216742	13.3	46.516742

176 rows × 5 columns

```python
total_score.to_csv("score_table.csv", index=False, header=True)
```

So “Top 10 in the US to make money” are:

	city	state	income score	employmen	total_score
1	city	state	income score	employmen	total_score
2	San Jose	CA	82.35	17.10	99.45
3	Seattle	WA	57.30	15.10	72.40
4	New York	NY	55.31	16.50	71.81
5	Washington	DC	54.43	17.20	71.63
6	Charlottesvil	VA	52.78	17.60	70.38
7	Midland	MI	53.38	16.60	69.98
8	Denver	CO	50.23	15.90	66.13
9	Minneapolis	MN	48.66	17.10	65.76
10	Philadelphia	PA	50.08	15.40	65.48
11	Baltimore	MD	48.30	17.00	65.30

*The calculation is just for demonstration, the final calculation formula should be developed by Point of Living magazine.