

# ETL Project

Group 11 - Group Members:

Raymond Galan  
Jeongdae JD Kwak  
Rainer Perry

## Objective:

Creation of a database for the cost of living in major cities around the world, combined with a breakdown of various costs and figures related to living in those cities, including: meals, various consumer goods, rent in apartments of various sizes, gym memberships, tuition, and net salaries. We then compared the data to NYC and calculated ratios.

## Data Sets Used:

### Cost of Living index by City 2020 Mid-Year

<https://www.numbeo.com/cost-of-living/rankings.jsp>

### Cost of Living Worldwide

<https://www.kaggle.com/morriswongch/cost-of-living>

## Libraries Imported:

- Pandas
- SQL Alchemy
- Psycpg2

## EXTRACT

Using Pandas, we retrieved both datasets through their respective URLs into the following dataframes, using the `pd.read_html` method:

```
# retrieve data
url = 'https://www.numbeo.com/cost-of-living/rankings.jsp'

response = pd.read_html(url)
```

living\_cost\_df  
breakdown\_df

## TRANSFORM

1. The “living\_cost\_df” contained the cost of indexes, so we extracted the following columns and renamed them accordingly (using the .drop and .rename methods):

- Cost\_of\_Living\_Index'
- Rent\_Index
- Living\_Cost",
- Groceries\_Index",
- Restaurant\_Price\_Index
- Purchasing\_Power\_Index

```
# retrieve data
url = 'https://www.numbeo.com/cost-of-living/rankings.jsp'

response = pd.read_html(url)

# clean data
living_cost_df.drop('Rank', axis=1, inplace=True)
living_cost_df.head()
```

2. The “breakdown\_df” contained the specific details to be combined with the indexes, hence we extracted the most relevant columns as follows by creating a cleaned-up dataframe with selected columns only, including the following examples:

“Meal\_for\_1”, “Meal\_for\_2”, “Bottle\_of\_Wine”, “Gym\_membership”,  
“Average\_Monthly\_net\_Salary”, and others, a total of 17 items.

```
#rename columns
living_cost_df.rename(columns={
    "Cost of Living Index": 'Cost_of_Living_Index',
    "Rent Index": 'Rent_Index',
    "Cost of Living Plus Rent Index": "Living_Cost",
    "Groceries Index": "Groceries_Index",
    "Restaurant Price Index": "Restaurant_Price_Index",
    "Local Purchasing Power Index": "Purchasing_Power_Index"
}, inplace=True)
```

3. To set NYC as the standard to compare all other city data to, we created a new dataframe that contained the ratios for the individual items, computed with NYC as the standard.

```
# create new table to compare the price values to NYC by dividing and coming up with a ratio.
compare_nyc_df = clean_df.set_index("City")
nyc_value = compare_nyc_df.loc["New York, NY, United States", :].to_list()
compare_nyc_df = compare_nyc_df / nyc_value
```

## LOAD:

The final step was loading the data frames into Postgresql by creating an engine and a connection to SQL.

All three data frames were loaded under the names:

“Living Cost Index”

“Detailed Breakdown”

“Living Cost Compare to NYC”

```
# create engine variable
engine = create_engine('postgresql://postgres:[REDACTED]@localhost:5432/cost_of_living')
conn = engine.connect()
```

```
# check current tables
engine.table_names()
```

```
# create table for living_cost_index
living_cost_df.to_sql(name="living_cost_index", con=engine, index=False)
```

```
# create table for detailed_breakdown
clean_df.to_sql(name="detailed_breakdown", con=engine, index=False)
```

```
# create table for living_cost_compare_to_nyc
compare_nyc_df.to_sql(name="living_cost_compare_to_nyc", con=engine, index=True)
```

Dashboard Properties SQL Statistics Dependencies Dependents cost\_of\_living/postgres@PostgreSQL 11 \*

cost\_of\_living/postgres@PostgreSQL 11

Query Editor Query History

```
1 SELECT f.*, d.*, n.*
2 FROM living_cost_index f
3 JOIN detailed_breakdown d
4 ON f."City" = d."City"
5 JOIN living_cost_compare_to_nyc n
6 ON f."City" = n."City";
```

Data Output Explain Messages Notifications

City	Cost_of_Living	Rent_Index	Living_Cost	Groceries_Index	Restaurant_P	Purchasing_P	City	Meal_for_1	Meal_for_2	Bottle_of_W	Bottle_of_W	Cigarettes_1	One_Way_1	Monthly_R	Liter_of_G	Apartment_1
text	double precise	double precise	double precise	double precise	double precise	double precise	text	double precise	double precise	double precise	double precise	double precise	double precise	double precise	double precise	double precise
1 Zurich, Swit...	131.49	64.37	98.8	131.23	120.39	121.12	Zurich, Swit...	27.25	109.01	1.23	15.81	9.54	4.69	92.66	1.73	2100.94
2 New York, N...	100	100	100	100	100	100	New York, N...	21.5	100	2.09	15	15	2.75	127	0.73	3350.62
3 Oslo, Norway	95.78	40.23	68.72	88.4	96.2	86.05	Oslo, Norway	18.87	83.86	1.92	15.72	13.68	3.88	80.72	1.7	1342.91
4 Bergen, Nor...	94.78	31.59	64.01	85.87	96.17	93.49	Bergen, Nor...	18.92	84.08	1.84	13.66	13.66	3.99	84.08	1.69	1010.56
5 San Francis...	92.13	109.76	100.72	89.79	88.26	139	San Francis...	18	80	1.89	15	12	3	81	1.01	3429.71
6 Reykjavik, I...	91.81	42.83	67.96	81.91	102.07	74.76	Reykjavik, I...	17.72	113.43	1.55	17.72	10.38	3.4	90.74	1.62	1413.66
7 Tel Aviv-Yaf...	91.4	43.17	67.91	80.53	92.53	67.85	Tel Aviv-Yaf...	17.61	88.06	1.65	14.68	10.27	1.73	63.11	1.8	1480.75
8 Copenhagen...	90.85	46.33	69.17	68.37	102.96	91.35	Copenhagen...	20.32	101.62	1.39	9.38	8.52	3.75	78.17	1.7	1578.16
9 Tokyo, Japan	89.69	38.36	64.69	90.93	49.17	75.09	Tokyo, Japan	8.12	47.78	1.18	13.38	4.97	1.91	97.94	1.34	1199.55
10 Paris, France	88.83	45.38	67.67	86.8	76.97	69.54	Paris, France	17.46	69.84	1.15	9.31	11.64	2.21	87.3	1.8	1410.15
11 Boston, MA	88.61	75.11	87.05	89.78	84.14	107.58	Boston, MA	18	77.5	2.77	15	11.71	2.7	80	0.68	7610.46

[image: join of 3 tables in pgAdmin]