



ETL PROJECT

Happiness, Countries and Regions

Abstract

This report outlines the process of Extracting, Transforming and Loading data using postgres SQL and Python

Arisbel Batista

Kritika Agarwal

James Rydlewski

Contents

Introduction	2
Background	2
Extracting	2
Reviewing CSV's	3
Transforming Data	4
Creating the postgres SQL Schema	6
Loading data.....	7
Rudimentary analysis.....	8
Discussion.....	8
Modifications	9
Conclusion.....	10

Introduction

Data can be found from a range of sources but it is not always in a readily useable format. Often data will be in comparable but slightly different forms which can make it challenging to make relations between varying data sets. Extracting Transforming and Loadinf (ETL) Is a critical element of data analytics as it enables databases to be built where relationships can be made in a more easily used format.

Background

To complete this ETL project it has been determined that country data will be used to build a database where relationships can be shown between metrics available for them. The data will be sourced via downloaded CSV files on Kaggle but in practice this could also be obtained via API calls or Web scraping.

Extracting

Kaggle is a useful source for data as it is open source and grants the end user the control to adjust headings to make it possible to combine data into a more useable format. The process of sourcing data to make these adjustments is the Extracting phase of an ETL assignment. Our group after reviewing Kaggle found that Country data is readily available and would be a good place to start our data journey.

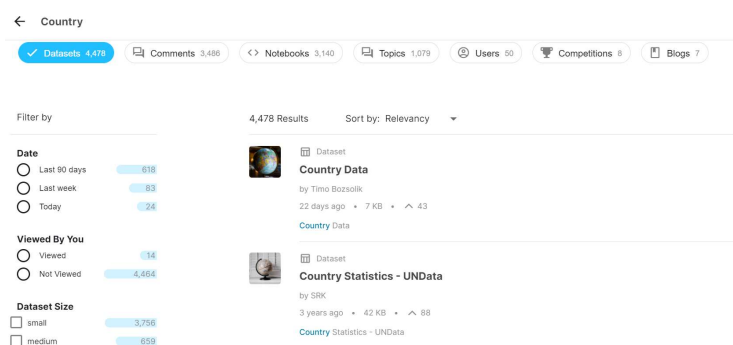
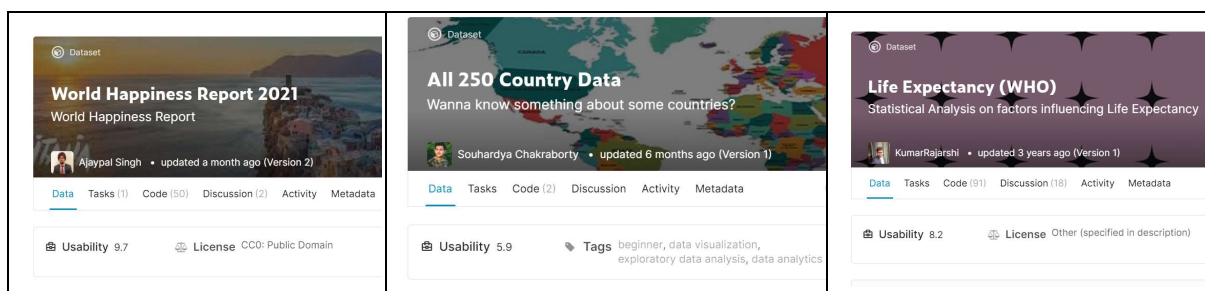





Figure 1 Screen grab of Kaggle data sets relating to country. This shows over 4000 data sets are available with this search!

The country data sets that we wished to combine to allow greater insights were:



-  The World Happiness Report
-  250 Country Data
-  Life Expectancy Data (WHO)¹

¹ Life Expectancy data was downloaded as a CSV but had been developed through an API search by another data scientist.

Reviewing CSV's

The csv's that we had obtained from Kaggle were opened to review what data that we wished to keep during the transformation stage of the project. At this stage it was observed that the data would need to be cleaned to facilitate better joins to be made once the operational database had been formed.

Country	n/year	Life Ladder	Log GDP per capita	Social support	Healthy life expectancy at birth	Freedom to make life choices	Generosity	Perceptions of corruption	Positive affect	Negative affect	
Afghanistan	2008	3.724	7.37		0.451	50.8	0.718	0.168	0.882	0.518	0.258
Afghanistan	2009	4.402	7.54		0.552	51.2	0.679	0.19	0.85	0.584	0.237

Figure 2 Happiness CSV

name	region	subregion	population	area	gini	Real Grow	Literacy Rate(%)	Inflation(%)	Unemployment(%)
Afghanistan	Asia	Southern	27657145	652230	27.8	3.1% (201	28.1% (2000 est.)	6.8% (2013 est.)	35% (2008 est.)

Figure 3 250 Country Data Set CSV

Country	Year	Status	Life expect	Adult Mor	Infant dea	Alcohol	percentag	Hepatitis	Measles	BMI	under-five	deaths	Polio	Total expen	Diphtheri	HIV/AIDS	GDP	Population	thinness	thinness 5-9 ye	Income comp	Schooling
Afghanistan	2015	Developin	65	263	62	0.01	71.27962	65	1154	19	83	6	8.16	65	0.1	584.2592	33736494	17.2	17.3	0.479	10.1	

Figure 4 Life Expectancy CSV

The transformations that were noted at this stage were:

- 📁 Making a common heading to be used as the primary key, This would be Country and required updating 'name' and 'country_name' from '250 Country Data set' and 'Happiness CSV' respectively.
- 📁 Removing irrelevant rows of data containing years that did not overlap. The happiness data was available over a 11yr period from 2008 to 2019, however, life expectancy was only from 2015. It was determined that the "cleaned" data would concentrate on 2015 to generate more relevant connections.
- 📁 Each Table had multiple columns which could be broken down for the data base to be created. Through discussion it was determined that many of these columns would be removed during the transformation stage of the assignment. The removed columns could be placed in additional data frames for further analysis but during this report it was determined that 3 would be sufficient.

Transforming Data

The data sets were transformed in python using Jupyter lab. The first stage of this is to import dependencies and establish a connection postgres SQL.

```
1]: import pandas as pd
    from sqlalchemy import create_engine
    # Import PostgreSQL username and password
    #from config import username, password

    from sqlalchemy import create_engine, inspect
    import pandas as pd
    from sqlalchemy.ext.automap import automap_base
    from sqlalchemy.orm import Session

    engine = create_engine('postgresql://postgres:[REDACTED]@localhost:5432/ETL Project')
    connection = engine.connect()
```

Figure 5 Dependencies and Connection to Postgres

The next step was Extracting the CSV's as Data frames.

Extract CSVs into DataFrames

```
[5]: # Import csv into economics_df
    economics_file = "250 Country Data.csv"
    economics_df = pd.read_csv(economics_file)

[6]: # Import csv into WHO_df
    WHO_file = "Life Expectancy Data.csv"
    WHO_df = pd.read_csv(WHO_file)

[7]: # Import csv into happiness_df
    happiness_file = "world-happiness-report.csv"
    happiness_df = pd.read_csv(happiness_file)
```

Figure 6 CSV's to DataFrames

Transform DataFrames

```
8]: # Split columns in economics_df that have mixed values
    economics_df[['Literacy Rate (%)', 'year']] = economics_df['Literacy Rate(%)'].str.split('%', expand=True)
    economics_df[['Inflation (%)', 'year', 'year']] = economics_df['Inflation(%)'].str.split('%', expand=True)
    economics_df[['Unemployment (%)', 'year', 'year']] = economics_df['Unemployment(%)'].str.split('%', expand=True)
    economics_df

# Rename and select the column headers
economics_transformed = economics_df.rename(columns={"name": "Country",
                                                    "region": "Continent",
                                                    "area": "Area",
                                                    "Literacy Rate (%)": "Literacy Rate",
                                                    "Inflation (%)": "Inflation",
                                                    "Unemployment (%)": "Unemployment"})
economics_df_cols = ["Country", "Continent", "Area", "Literacy Rate", "Inflation", "Unemployment"]
economics_transformed = economics_transformed[economics_df_cols].copy()
economics_transformed.head()
```

Figure 7 Transformations made on 250 Country Data

[8]:

Unnamed: 0	name	region	subregion	population	area	gini	Real Growth Rating(%)	Literacy Rate(%)	Inflation(%)	Unemployment(%)	Literacy Rate (%)	year	Inflation (%)	Unemployment (%)	
0	0	Afghanistan	Asia	Southern Asia	27657145	652230.0	27.8	3.1% (2013 est.)	28.1% (2000 est.)	6.8% (2013 est.)	35% (2008 est.)	28.1	None	6.8	35
1	1	Åland Islands	Europe	Northern Europe	28875	1580.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

Figure 8 Economics DF Before Transformation

	Country	Continent	Area	Literacy Rate	Inflation	Unemployment
0	Afghanistan	Asia	652230.0	28.1	6.8	35
1	Åland Islands	Europe	1580.0	NaN	NaN	NaN
2	Albania	Europe	28748.0	98.7	1.7	16.9
3	Algeria	Africa	2381741.0	79	3.9	10.3
4	American Samoa	Oceania	199.0	97	NaN	NaN

Figure 9 Economics DF after Transformation

This process was repeated for each of the data frames imported as shown by the following screen grabs

```
[ ]: # Filter WHO_df for 2015 year
WHO_df = WHO_df[WHO_df.Year == 2015]

[ ]: # Rename and select the column headers
WHO_transformed = WHO_df.rename(columns={"Country": "Country",
                                         "Status": "Status",
                                         "Life expectancy ": "Life_expectancy",
                                         "Hepatitis B": "Hepatitis_B",
                                         "Polio": "Polio",
                                         "GDP": "GDP",
                                         "Population": "Population"})
WHO_df_cols = ["Country", "Status", "Life_expectancy", "Hepatitis_B", "Polio", "GDP", "Population"]
WHO_transformed = WHO_transformed[WHO_df_cols].copy()
WHO_transformed.head()
```

[1]:

	Country	Status	Life_expectancy	Hepatitis_B	Polio	GDP	Population
0	Afghanistan	Developing	65.0	65.0	6.0	584.259210	33736494.0
16	Albania	Developing	77.8	99.0	99.0	3954.227830	28873.0
32	Algeria	Developing	75.6	95.0	95.0	4132.762920	39871528.0
48	Angola	Developing	52.4	64.0	7.0	3695.793748	2785935.0
64	Antigua and Barbuda	Developing	76.4	99.0	86.0	13566.954100	NaN

Figure 10 Life Expectancy Transformations

```
: # Filter happiness_df for 2015 year
happiness_df = happiness_df[happiness_df.year == 2015]

: # Rename and select the column headers
happiness_cols = ["Country name", "Life Ladder", "Social support", "Freedom to make life choices", "Perceptions of corruption"]
happiness_transformed = happiness_df.rename(columns={"Country name": "Country",
                                                    "Life Ladder": "Happiness rating",
                                                    "Social support": "Social support",
                                                    "Freedom to make life choices": "Freedom to make life choices",
                                                    "Perceptions of corruption": "Perception of corruption"})
happiness_df_cols = ["Country", "Happiness rating", "Social support", "Freedom to make life choices", "Perception of corruption"]
happiness_transformed = happiness_transformed[happiness_df_cols].copy()
happiness_transformed.head()
happiness_transformed.set_index("Country", inplace=True)
happiness_transformed.head()
```

	Happiness rating	Social support	Freedom to make life choices	Perception of corruption
Country				
Afghanistan	3.983	0.529	0.389	0.881
Albania	4.607	0.639	0.704	0.885
Argentina	6.697	0.926	0.881	0.851
Armenia	4.348	0.723	0.551	0.901
Australia	7.309	0.952	0.922	0.357

Figure 11 Happiness Transformations

Once the transformations were complete a schema was created in postgres to enable the data to be pushed into a centralised database.

Creating the postgres SQL Schema

Whilst completing this project it was discovered that if the headings contain no special characters such as %, () or [] dataframes could be pushed directly in the data base that had been created in postgres. Even though this was possible the steps taken to create tables is shown below. It was determined to set country as the primary key.

```
-- getting rid of preexisting tables
DROP TABLE IF EXISTS economics;
DROP TABLE IF EXISTS happiness;
DROP TABLE IF EXISTS who;

-- making sql locations for csvs and loading them in
CREATE TABLE "happiness" (
  "Country" text NOT NULL,
  "Happiness rating" double NOT NULL,
  "Social support" double NOT NULL,
  "Freedom to make life choice" double NOT NULL,
  "Perception of corruption" double NOT NULL,
  CONSTRAINT "primarykey_happiness" PRIMARY KEY ("Country")
)

CREATE TABLE "economics" (
  "Country" text NOT NULL,
  "Continent" double NOT NULL,
  "Area" double NOT NULL,
```

Figure 12 Creating Tables in SQL

Once this schema had been created it was possible to load the dataframes into postgres.

Loading data

The following screengrab show the code required for the final steps in this project of loading the data into postgres.

Create database connection

```
] : # Create Engine
# engine = create_engine(f"postgresql://{username}:{password}@localhost:5432/happiness_db")
# connection = engine.connect()#

engine = create_engine('postgresql://postgres:[REDACTED]@localhost:5432/ETL Project')
connection = engine.connect()

] : # Confirm tables
engine.table_names()

] : ['economics', 'happiness', 'WHO', 'who']
```

Figure 13 Creating Database connection

The tables that had been created in postgres then had the data pushed into them from jupyter lab

<pre>conn = engine.raw_connection() cur = conn.cursor() output = io.StringIO() df.to_csv(output, sep='\t', header=False, index=False) output.seek(0) contents = output.getvalue() cur.copy_from(output, 'table_name', null='') # null values become '' conn.commit()</pre>	<pre>WHO_transformed.to_sql(name='who', con=engine, if_exists='append', index=True) happiness_transformed.to_sql(name='happiness', con=engine, if_exists='append', index=False) economics_transformed.to_sql(name='economics', con=engine, if_exists='append', index=True)</pre>
--	--

Figure 14 Pushing Data into SQL Tables

To check that this data had successfully been exported it was reimported into the python code from the postgres database.

```
conn = engine.connect()

economics_import = pd.read_sql("SELECT * FROM economics", conn)
happiness_import = pd.read_sql("SELECT * FROM happiness", conn)
who_import = pd.read_sql("SELECT * FROM who", conn)
```

Figure 15 Reimporting data from postgres

Rudimentary analysis

The significance of this project was to demonstrate the ETL process. This has been shown in the previous steps. The data that has been used can then be analysed further by merging on country columns and selecting which data is required for analysis.

The means of analysing the data is merging on country using pandas. An example of this is shown below with a scatter diagram to visualise any link between life expectancy and happiness.

```
] : # joining for analysis
# Happiness vs Literacy Rate
# Happiness vs Inflation
# Happiness vs Life expectancy

Happiness_vs_Literacy_Rate = pd.merge(happiness_import,economics_import, on="Country", how="inner")
Happiness_vs_Life_expectancy = pd.merge(happiness_import,who_import, on="Country", how="inner")
Happiness_vs_Life_expectancy.head()
```



Figure 16 Happiness vs Life expectancy scatter plot after reimporting and merging using pandas

Discussion

This project has shown that data can be obtained from a number of sources and relational data bases can be made after cleaning the data to ensure that it can be joined with other data sets. For future projects after obtaining raw data it would be desirable to generate a unique identifier that could then provide alternative means of joining data beyond that of the columns initially obtained. It could also be prudent to make data tables with fewer columns using this unique identifier to facilitate future joins and make data analysis easier for those using the data base. Whilst the initial project is functional it lacks useability owing to this. To address this modifications have been made to the initial approach.

Modifications

In the initial foray into ETL a database was created utilising the primary key of Country in the created tables. It was found that there were inconsistencies in the naming standards of countries which led to dramatic reductions in available data when subsequent merges were made. TO address this an alternat approach to cleaning the data has been performed which yielded a unique country identifier. By importin “pycountry” it was possible to iterate through the data and smooth out some of the spelling or grammatical issues surrounding the country name. this would then make it easier to join and also grant a country code to use as a unique identifier.

```
143 rows x 5 columns

[7]: # Define a function to add the alpha_2 id code corresponding to each country name
def findCountry (country_name):
    try:
        return pycountry.countries.get(name=country_name).alpha_2
    except:
        return ("Not found")

[9]: happiness_transformed["id"] = happiness_transformed.apply(lambda row: findCountry(row.country), axis = 1)
happiness_transformed

[9]:
```

	country	happiness_rating	social_support	freedom	corruption	id
7	Afghanistan	3.983	0.529	0.389	0.881	AF
19	Albania	4.607	0.639	0.704	0.885	AL
46	Argentina	6.697	0.926	0.881	0.851	AR
61	Armenia	4.948	0.792	0.661	0.901	AM

Figure 17 Using Pycountry to generate unique identifiers

At this stage all data frames were merged on the new “ID”column. By isolating the unique values in this column and extracting the matching countries a new dataframe featuring Country and ID was made. This dataframe is showing the overlap of countries that occurs in all the dataframes which will make future analysis easier as there will be fewer omitted values.

```
2922 Zimbabwe Developing 67.0 87.0 88.0 118.693830 15777451.0 ZW

183 rows x 8 columns

[16]: # Merge on country to generate a unique country id
merged1 = pd.merge(happiness_transformed, WHO_transformed, on = "id", how='left')
merged2 = pd.merge(merged1, economics_transformed, on = "id", how='left')
merged2

[16]:
```

	country_x	happiness_rating	social_support	freedom	corruption	id	country_y	status	life_expectancy	hepa
0	Afghanistan	3.983	0.529	0.389	0.881	AF	Afghanistan	Developing	65.0	
1	Albania	4.607	0.639	0.704	0.885	AL	Albania	Developing	77.8	
2	Argentina	6.697	0.926	0.881	0.851	AR	Argentina	Developing	76.3	

```
country_alpha = country_alpha.reset_index()
country_alpha

[19]:
```

	id	country
0	AF	Afghanistan
1	AL	Albania
2	AR	Argentina
3	AM	Armenia
4	AU	Australia
...
3631	UY	Uruguay
3632	UZ	Uzbekistan
4101	YE	Yemen
4102	ZM	Zambia
4103	ZW	Zimbabwe

126 rows x 2 columns

Figure 18 Figures illustrating the creation of a unique identifier dataframe to be used for future joins

Once this new dataframe of country data was created it was added to the country schema in postgres to be included for further analysis.

	Data Output	Explain	Messages
	ID [PK] text		country text
1	AE		United Arab...
2	AF		Afghanistan
3	AL		Albania
4	AM		Armenia
5	AR		Argentina
6	AT		Austria
7	AU		Australia
8	AZ		Azerbaijan
9	BA		Bosnia and ...
10	BD		Bangladesh
11	BE		Belgium
12	BF		Burkina Faso
13	BG		Bulgaria
14	BH		Bahrain

Figure 19 Unique identifier data table in Postgres

Conclusion

This report has demonstrated the intricacies required in building a relational database. The tables used were relatively well organised and consistent with each other which certainly streamlined the database building process. In practice this is not too likely to occur, and it has been useful to learn methods to clean data prior to building a database. The transformations element of this assignment has been challenging as despite the initial assumption of merging on country it has been shown that a unique identifier was required to facilitate analysis. These modifications have been included in the report which reflect the need to be adaptable when obstacles are encountered.