
Worldwide expenditure on public health

Trends based on the GDP and Income Groups of countries

Contents

| | | |
|-----------|--|-----------|
| 1 | Project Aims | 3 |
| 2 | Data Sources | 3 |
| 3 | Workflow Diagram | 4 |
| 4 | ER Diagram | 4 |
| 5 | Data Transformation | 5 |
| 6 | Inserting data into desired database using Python | 6 |
| 7 | Observing created database tables in phpMyAdmin | 10 |
| 8 | MySQL queries to obtain desired data from a database in csv | 12 |
| 9 | Creating a single file run for the project | 14 |
| 10 | Visualization in Tableau | 15 |
| 11 | Conclusion | 18 |
| 12 | Lessons learned, challenges and future work | 18 |

Abstract

This project tries to establish a connection between the GDP of a country, its GDP per capita, and the health expenditure per capita for its citizens. It also tries to establish if there is any correlation between the health expenditure per capita of a country, and the country's disability adjusted life years (DALY). DALY is the measure of the overall disease burden of a country. It tells us the number of 'healthy' life years lost due to death or disease. It takes both mortality and morbidity into account, and hence is a very good measure of the public health of a country.

The data for the countries was sourced from the World Bank and WHO. The data was available for download in a csv format, and Python coding language was used to convert the raw data schemas to something that was more appropriate for the project. The transformed data was stored in a database in phpMyAdmin. These databases were created using Python and MySQL queries. Python and MySQL queries were once again used to extract data (with desired relations), and this data could be exported to csv files for further investigation.

Since visualization makes understanding things much easier, Tableau was used to represent the data and the desired relationships. Tableau is a visualization software and tool. It can import data from databases, or text/csv files and visually represent it as filled maps, bar graphs, pie charts, treemaps, and so on.

To make the process of understanding relationships between data sets, and to establish connections between them, ER diagrams were made. This was the first step of the project, though as the project progressed, it became an iterative process, and the ER diagram changed from what it initially was.

1 Project Aims

This study was initiated due to a recent newspaper article (<http://in.reuters.com/article/2014/12/23/india-health-budget-idINKBN0K10Y020141223>). It was surprising to me that the health budget was slashed, when most people I know think it should be otherwise. Observing and representing the health expenditure trends of various countries, and the subsequent health consequences, seem to be very crucial to back claims that the health budget should be increased. Since the GDP and income groups of a country are critical features of expenditure capacity, it was important to represent this as well. Thus, this project asks two main questions,

- How much do countries spend on their citizens' health, given their GDP per capita?
- How does Disability Adjusted Life Years (DALY) of a country get affected by the health expenditure per capita?

2 Data Sources

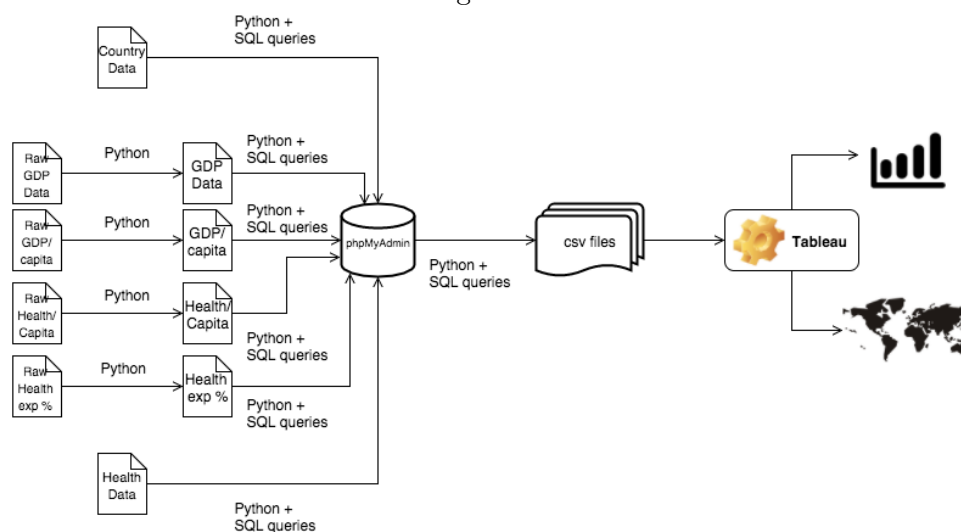
1. The GDP and monetary related data was sourced from the World Bank website. The list below gives a link to all the data sources used. The metadata for these also covers the income groups that the countries belong to.
 - GDP of countries over the years <http://data.worldbank.org/indicator/NY.GDP.MKTP.CD>
 - GDP per capita of countries over the years <http://data.worldbank.org/indicator/NY.GDP.PCAP.CD>
 - Percentage of GDP spent on Health Care <http://data.worldbank.org/indicator/SH.XPD.TOTL.ZS>
 - Health expenditure per Capita <http://data.worldbank.org/indicator/SH.XPD.PCAP>
2. The DALY data of countries was obtained from WHO <http://apps.who.int/gho/data/view.main.GHEASDALYRTCTRY>

3. Consistent countries names and codes was required to establish links and relationships across different data sets. This wasn't initially foreseen, but when I ran into entity resolution issues, such a 'country table linked with alternative country names' was required. The data for this was taken from <http://www.opengeocode.org/download/countrynames.txt>

3 Workflow Diagram

The workflow diagram described in Figure 1 depicts the overall workflow that was used for this project. Later sections of this report will describe each step in detail.

Figure 1:



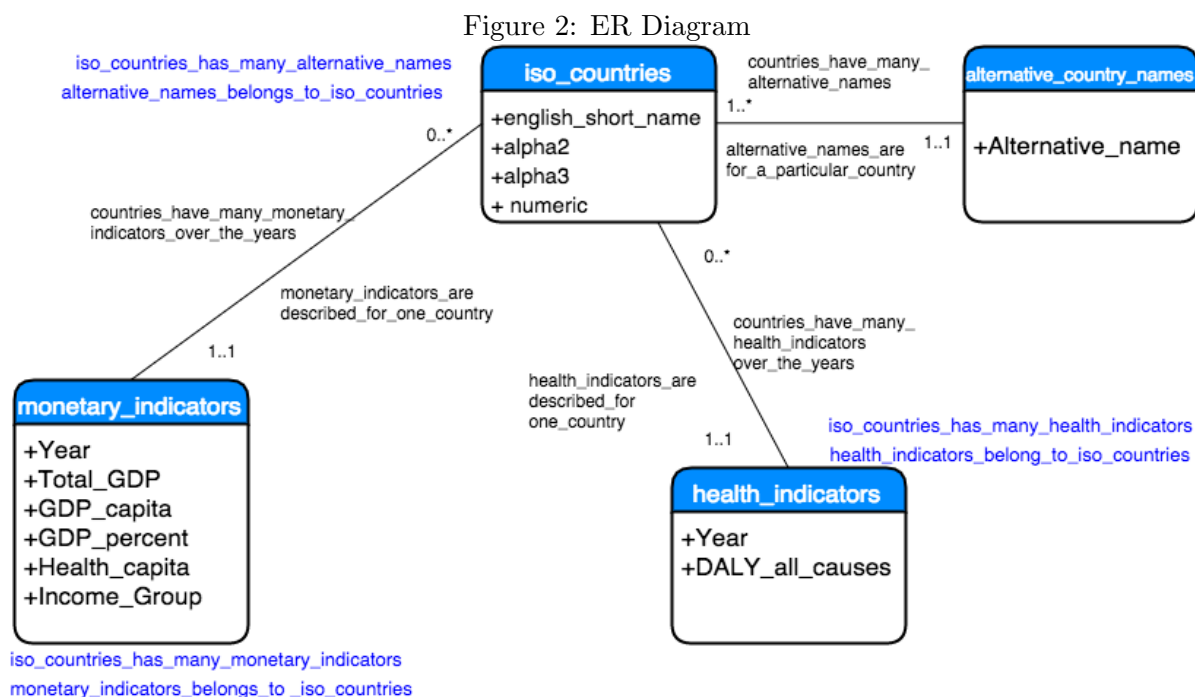
4 ER Diagram

An ER diagram was made to establish links between different tables, and to figure out how to have relationships between data sets.

The iso countries table was used to list all the standard iso names/codes that are used. The alternative country names table was used to account for the alternate names a country can have (eg, The United States of America is known as USA, United States, etc).

All monetary factors (related to expenditure, income group and GDP) were included in the monetary indicators table. The DALY was included in the health indicators table. Both monetary and health tables derived their country id as foreign keys from that table. As they country names in world bank and WHO were quite different, the iso country and the alternate country names would help create proper relations, and derive proper foreign country id.

See Figure 2 for a detailed ER diagram.



5 Data Transformation

To get the raw csv data into the format required by the tables described in the ER diagram, Python was used. The data from the World Bank was in a pivot table format. This was changed to normal rectangular csv by ‘popping’ the extra data fields, and later adding it each dict element in a list of dicts.

Figure 3: Data in pivot table format

```

"Country Name","Country Code","Indicator Name","Indicator Code","1960","1961","1962","1963","1964",
"Aruba","ABW","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"Andorra","AND","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,78617623.26772
"Afghanistan","AFG","GDP (current US$)","NY.GDP.MKTP.CD",537777811.911111,548888894.577778,54
"Angola","AGO","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"Albania","ALB","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"Arab World","ARB","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,25659088715.414,
"United Arab Emirates","ARE","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"Argentina","ARG","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,24450604877.6081,18272123664.4715,
"Armenia","ARM","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"American Samoa","ASM","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"Antigua and Barbuda","ATG","GDP (current US$)","NY.GDP.MKTP.CD",,,,,,
"Australia","AUS","GDP (current US$)","NY.GDP.MKTP.CD",18574308433.1952,19651696718.5575,1988
"Austria","AUT","GDP (current US$)","NY.GDP.MKTP.CD",6592693841.18495,7311749633.36229,775611
  
```

The data tables also had to be encoded in a utf8 format, and this was also written in the python script.

Figure 4: Data transformation from pivot table style to rectangular

```

import csv
import pprint

# Opening csv file to read from. This file is in a pivot table format
with open('GDP_country2.csv', encoding='utf8') as csvfile:
    #Opening the csv file to write to.
    with open('GDPcountries2.csv', 'w') as csvfileout:

        myCSVReader = csv.DictReader(csvfile, delimiter=',', quotechar='"')

        #defining headers of the desired, transformed csv
        new_headers = ["Country_Name", "Country_Code", "Year", "GDP_total"]
        myCSVWriter = csv.DictWriter(csvfileout, delimiter=',', quotechar='"', fieldnames = new_headers)
        myCSVWriter.writeheader()

        for rowDict in myCSVReader:
            #to view which format the initial dict is in
            pprint.pprint(rowDict)
            #It is in the following format--for each country, year:gdp, country name:---, country code:---

            #removing unnecessary rows in the dict
            Country_Name = rowDict.pop('\u0000Country Name')
            Country_Code = rowDict.pop("Country Code")
            Series_Name = rowDict.pop("Indicator Name")
            Series_Code = rowDict.pop("Indicator Code")

            #Now the row dict has only year:gdp
            for year,count in rowDict.items(): #
                print("Country_Name: {}, Country_Code: {}, Year: {}, GDP_total: {}".format(Country_Name, Country_Code, year, count))

            #adding all elements to a new dict, and writing that dict out in a new csv file
            #We now will have a normal, rectangular, csv file
            newRow = {"Country_Name" : Country_Name, "Country_Code" : Country_Code, "Year" : year, "GDP_total" : count}
            myCSVWriter.writerow(newRow)

print("Done")

```

The WHO data had commas in the numerical values. This meant that they can't be used for mathematical computing and calculations. Thus, it too had to be changed. This was discovered much later, and was definitely an iterative process.

Figure 5: Data transformation to remove comma, helps in converting column from string to int

```

import pymysql
import pprint
import csv

with open('DALY_per100000.csv') as csvfile:
    with open('DALY.csv', 'w') as csvfileout:
        myCSVReader = csv.DictReader(csvfile, delimiter=",", quotechar='"')
        new_headers = ["Country_Name", "Year", "All_Causes", "Communicable", "Noncommunicable_diseases", "Injuries"]
        myCSVWriter = csv.DictWriter(csvfileout, delimiter=',', quotechar='"', fieldnames = new_headers)
        myCSVWriter.writeheader()

        # move row by row through the file
        for row in myCSVReader:
            row["All_Causes"] = row["All_Causes"].replace(',', '')
            print(row["All_Causes"])
            myCSVWriter.writerow(row)

print("done")

```

6 Inserting data into desired database using Python

A database was created using Python and MySQL queries. The country data was inserted in two different tables, one that has iso standard names and codes. The other, that used the iso standard names as foreign keys, and for each country id had a list of potential alternative names.

Figure 6: Dict mapping for countries

```
#Creating a dict, here keys will be alternate country names,value will be english short name
alt_name_map = {}

# Now can query alternative names like this:
get_alt_sql = """
SELECT alternative_name, english_short_name
FROM iso_countries, alternative_country_names
WHERE iso_countries.id = alternative_country_names.iso_country_id
"""

cursor.execute(get_alt_sql)
results = cursor.fetchall()
for row in results:
    alt_name_map[row["alternative_name"]] = row["english_short_name"]
```

Figure 7: Country tables insert

```
cursor.execute("TRUNCATE iso_countries; TRUNCATE alternative_country_names")

country_sql = """
INSERT INTO iso_countries(`english_short_name`,`alpha2`,`alpha3`,`numeric`)
VALUES (
    %(english_short_name)s, %(alpha2)s, %(alpha3)s, %(numeric)s
)
"""

alt_sql = """
INSERT INTO alternative_country_names(iso_country_id, alternative_name)
VALUES (
    %(iso_country_id)s, %(alternative_name)s
)
"""

# Do inserts a bit quicker (see loading-very-large-csv.py)
cursor.execute("SET autocommit = 0")
```

Figure 8: Country tables insert (cont)

```

cursor.execute("SET autocommit = 0")

with open('countrynames.csv', encoding='utf8') as csvfile:

    # tell python about the specific csv format
    # CommentedFile skips lines with # chars
    myCommentWrapCSVReader = csv.reader(CommentedFile(csvfile), delimiter=";", quotechar="'")

    # move row by row through the file
    for row in myCommentWrapCSVReader:
        # Each row arrives in Python as a Dict
        print("-"*30)
        country_table = row[0:4] #alpha2, alpha3, numeric, english_short_name
        #pprint.pprint(country_table)

        # You want all the alt names as alternatives,
        # but you also want the english_short_name and the alphas
        # in the alternatives too.
        alternatives = row # all of the country codes map to the country

        # Now add a row for the country. # strip left spaces
        param_dict = { "english_short_name": country_table[3].rstrip(),
                      "alpha2": country_table[0].rstrip(),
                      "alpha3": country_table[1].rstrip(),
                      "numeric": int(country_table[2].rstrip())
                    }
        cursor.execute(country_sql, param_dict)
        # Get the id to use in the rest of the insertions
        new_country_id = cursor.lastrowid

        # Now for each alternative name
        # insert a row into the alternatives table, linked to country
        for alt_name in alternatives:
            param_dict = { "iso_country_id": new_country_id,
                          "alternative_name": alt_name.rstrip()
                        }
            cursor.execute(alt_sql, param_dict)

cursor.execute("COMMIT")

```

The monetary data included total GDP, GDP per capita, GDP percent on health, and health expenditure per capita. The country id as a foreign key was obtained using alternate country names table and iso table. This was done by connecting alternate country names and iso names using a 'mapping' dict. Thus, no matter the way in which the country name was written in World Bank Data, that name could be mapped to iso country name, and thus an accurate foreign id could be obtained.
















Figure 9: iso table in phpMyAdmin

| <input type="checkbox"/> | | | | <input type="checkbox"/> | id | english_short_name | alpha2 | alpha3 | numeric |
|--------------------------|--|--|--|--------------------------|-----------|---------------------------|---------------|---------------|----------------|
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 1 | Afghanistan | AF | AFG | 4 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 2 | Aland Islands | AX | ALA | 248 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 3 | Albania | AL | ALB | 8 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 4 | Algeria | DZ | DZA | 12 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 5 | American Samoa | AS | ASM | 16 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 6 | Andorra | AD | AND | 20 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 7 | Angola | AO | AGO | 24 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 8 | Anguilla | AI | AIA | 660 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 9 | Antarctica | AQ | ATA | 10 |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 10 | Antigua and Barbuda | AG | ATG | 28 |

Figure 10: monetary table in phpMyAdmin

| <input type="checkbox"/> | | | | <input type="checkbox"/> | id | Year | Country_id | Total_GDP | GDP_capita | GDP_percent_health | Health_capita | Income_Group |
|--------------------------|--|--|--|--------------------------|-----------|-------------|-------------------|------------------|-------------------|---------------------------|----------------------|---------------------|
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 871 | 2006 | 29 | 12550175101 | 3270 | 9 | 275 | Upper middle income |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 872 | 2005 | 21 | 30210091837 | 3126 | 7 | 215 | Upper middle income |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 873 | 2002 | 21 | 14594925393 | 1479 | 6 | 95 | Upper middle income |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 874 | 2013 | 21 | 73097619637 | 7722 | 6 | 463 | Upper middle income |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 875 | 2010 | 21 | 55220932614 | 5819 | 6 | 323 | Upper middle income |
| <input type="checkbox"/> | | | | <input type="checkbox"/> | 876 | 1996 | 21 | 14756861538 | 1452 | 6 | 89 | Upper middle income |

Figure 11: health table in phpMyAdmin

| | | | | id | Year | Country_id | DALY_all_Causes |
|--------------------------|---|------|---|------|---|------------|-------------------|
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 1 2012 1 68,970 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 2 2000 1 86,566 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 3 2012 3 29,903 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 4 2000 3 36,919 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 5 2012 4 34,790 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 6 2000 4 38,868 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 7 2012 7 93,709 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 8 2000 7 115,496 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 9 2012 11 26,808 |
| <input type="checkbox"/> |  | Edit |  | Copy |  | Delete | 10 2000 11 29,479 |

7 Observing created database tables in phpMyAdmin

After inserting data through Python queries, the data was observed to ensure the filling was done properly.

All the data was thus inserted into the monetary tables, deriving the foreign key from the iso country table, and matching year and country ids whenever required.

DALY table was filled in a similar manner

Figure 12: Example of monetary table insert

```

print("Processing")
with open('GDPcountries2.csv') as csvfile:
    myCSVReader = csv.DictReader(csvfile, delimiter=",", quotechar='"')

    # move row by row through the file
    for row in myCSVReader:
        #Proceed only if value is present in the alternate country tables, eliminates non
        #country names, such as the 'European Union'
        if( row["Country_Name"] in alt_name_map.keys()):

            # Mapping Country_Name from csv to english short name
            #Creates a new column in csv list of dicts, 'mapped_code" and assigns correct
            #value, using the appropriate 'key'
            row["mapped_code"] = alt_name_map[row["Country_Name"]]
            pprint.pprint(row) #Checking to see how the csv list of dicts has been changed
            print("Mapped {} to {}".format(row["Country_Name"],row["mapped_code"]))

            #To get foreign key for health_indicators from the iso_countries
            sql = "SELECT id FROM iso_countries WHERE english_short_name = %(mapped_code)s"
            cursor.execute(sql, row)

            #Checking if the country already exists, only one row if result will be obtained
            if (cursor.rowcount == 1):
                result = cursor.fetchone()
                Country_id = result['id']
                print("Got the country id {}".format(result['id']))

            #Skip null GDP values
            if (row["GDP_total"] == ""):
                pass
            #Skip null Year values
            elif (row["Year"] == ""):
                pass

```

Figure 13: Example of monetary table insert cont,

```

#Skip null GDP values
if (row["GDP_total"] == ""):
    pass
#Skip null Year values
elif (row["Year"] == ""):
    pass

else:
    # Now we have proper country id s. Insert all values.
    mont_sql = """INSERT INTO monetary_indicators(Year,Country_id,Total_GDP)
        VALUE (%(Year)s, %(new_country_id)s, %(GDP)s)"""

    param_dict = { "Year": row["Year"],
                    "new_country_id": Country_id,
                    "GDP": row["GDP_total"] }

    cursor.execute(mont_sql, param_dict)

print ("Done")

```

8 MySQL queries to obtain desired data from a database in csv

MySQL queries were made using Python, and these queries connected the DALY of countries, and the monetary indicators for a particular year.

1. The first query retrieved GDP per capita, health expenditure per capita, and the DALY for each country for the year 2012. The number of countries was 24, with 6 in each income group. The selection was made randomly to that there's no bias while observing trends. India was selected in addition, so that one can compare where it stands in terms of GDP/capita, Health expenditure per capita and DALY.
2. The second query retrieved India's GDP and GDP percentage of health expenditure over the years.

Figure 14: Example of MySQL query 1

```
#####
#Countries with GDP per capita and health expenditure per capita
#####

with connection.cursor() as cursor:
    sql = """SELECT DISTINCT iso_countries.english_short_name AS Country_Name,
                           monetary_indicators.GDP_capita,monetary_indicators.Health_capita,
                           monetary_indicators.Income_Group, health_indicators.DALY_all_Causes
        FROM iso_countries,monetary_indicators,health_indicators
        WHERE iso_countries.id = monetary_indicators.Country_id
              AND iso_countries.id = health_indicators.Country_id
              AND monetary_indicators.Year = 2012
              AND monetary_indicators.Income_Group = "Low income"
              AND monetary_indicators.Health_capita != ""
        ORDER BY RAND()
        LIMIT 6 """
```

Figure 15: Example of MySQL query 2

```
with connection.cursor() as cursor:
    sql7 = """ SELECT iso_countries.english_short_name AS Country_Name,
                   monetary_indicators.GDP_percent_health,
                   monetary_indicators.Total_GDP,
                   monetary_indicators.Year
        FROM iso_countries,monetary_indicators
        WHERE iso_countries.id = monetary_indicators.Country_id
              AND iso_countries.english_short_name = "India"
              AND monetary_indicators.Health_capita != ""
        ORDER BY monetary_indicators.Year """

    cursor.execute(sql7)
    results2 = cursor.fetchall()

    pprint.pprint(results2)

venues_keys = ['Country_Name', 'GDP_percent_health', 'Total_GDP', 'Year']

with open('percentage_table.csv', 'w') as csvfile:

    myCsvWriter = csv.DictWriter(csvfile, delimiter=',', quotechar='"', fieldnames = venues_keys)

    myCsvWriter.writeheader()
    for row in results2:

        myCsvWriter.writerow(row)

print("done")
```

Scipy was used to do a Spearman rank correlation between GDP per capita/Health expenditure and Health expenditure/DALY .

Figure 16: Rank Correlation with spearman

```
with connection.cursor() as cursor:

    sql = """SELECT monetary_indicators.GDP_capita
              FROM iso_countries,monetary_indicators,health_indicators
              WHERE iso_countries.id = monetary_indicators.Country_id
                    AND iso_countries.id = health_indicators.Country_id
                    AND monetary_indicators.Year = 2012
                    AND health_indicators.Year = 2012
                    AND monetary_indicators.Health_capita != "" """

    cursor.execute(sql)
    results = cursor.fetchall()
    list = [] #Making a list to store returned values of GDP per capita

    for row in results:
        for key in row:
            list.append(row[key]) #The column returned as a dict, storing key values from it

#get a list to store values of Health exp per capita
    sql2 = """SELECT monetary_indicators.Health_capita
              FROM iso_countries,monetary_indicators,health_indicators
              WHERE iso_countries.id = monetary_indicators.Country_id
                    AND iso_countries.id = health_indicators.Country_id
                    AND monetary_indicators.Year = 2012
                    AND health_indicators.Year = 2012
                    AND monetary_indicators.Health_capita != "" """

    cursor.execute(sql2)
    results2 = cursor.fetchall()
    list2 = []

    for row in results2:
        for key in row:
            list2.append(row[key])

correlation = scipy.stats.spearmanr(list,list2)[0]
print("\n\nThe rank correlation is {}".format(correlation))
```

Figure 17: Rank Correlation with spearman

```

sql3 = """SELECT health_indicators.DALY_all_Causes
            FROM iso_countries,monetary_indicators,health_indicators
            WHERE iso_countries.id = monetary_indicators.Country_id
                  AND iso_countries.id = health_indicators.Country_id
                  AND monetary_indicators.Year = 2012
                  AND health_indicators.Year = 2012
                  AND monetary_indicators.Health_capita != "" """

cursor.execute(sql3)
results3 = cursor.fetchall()
pprint.pprint(results3)
list3 = []

for row in results3:
    for key in row:
        list3.append(row[key])

pprint.pprint(list3)

correlation = scipy.stats.spearmanr(list,list2)[0]

print("\n\nThe rank correlation for GDP-Health per capita is {}".format(correlation))

correlation2 = scipy.stats.spearmanr(list2,list3)[0]

print("\n\nThe rank correlation for DALY and Health per capita is {}".format(correlation2))

```

Figure 18: Rank Correlation with spearman

```

The rank correlation for GDP-Health per capita is 0.9737784655746775

The rank correlation for DALY and Health per capita is -0.8645859821922042
sukanyam@holden:~/Project$ █

```

9 Creating a single file run for the project

I created a shell script file to run all my python files in a particular order.

Figure 19: Shell Script

```
python3 creating_tables
python3 iso_countries_alt_insert.py
python3 GDP_total_transform.py
python3 GDP_total_insert.py
python3 GDP_capita_transform.py
python3 GDP_capita_insert.py
python3 GDP_percent_transform.py
python3 GDP_percent_insert.py
python3 Health_capita_transform.py
python3 Health_capita_insert.py
python3 daly_insert.py
python3 project_sql.py
python3 sql_trial.py
```

10 Visualization in Tableau

I found Tableau to be very intuitive. A lot of ‘drag and drop’ elements, and prompts made this process easy. The layout was also similar to pivot tables in excel, and perhaps this made it simpler to understand Tableau. I actually didn’t go through any tutorials yet, probably because the visualizations I had to do were quite simple. In the future, I will probably refer YouTube tutorials.

I made the three following visualizations

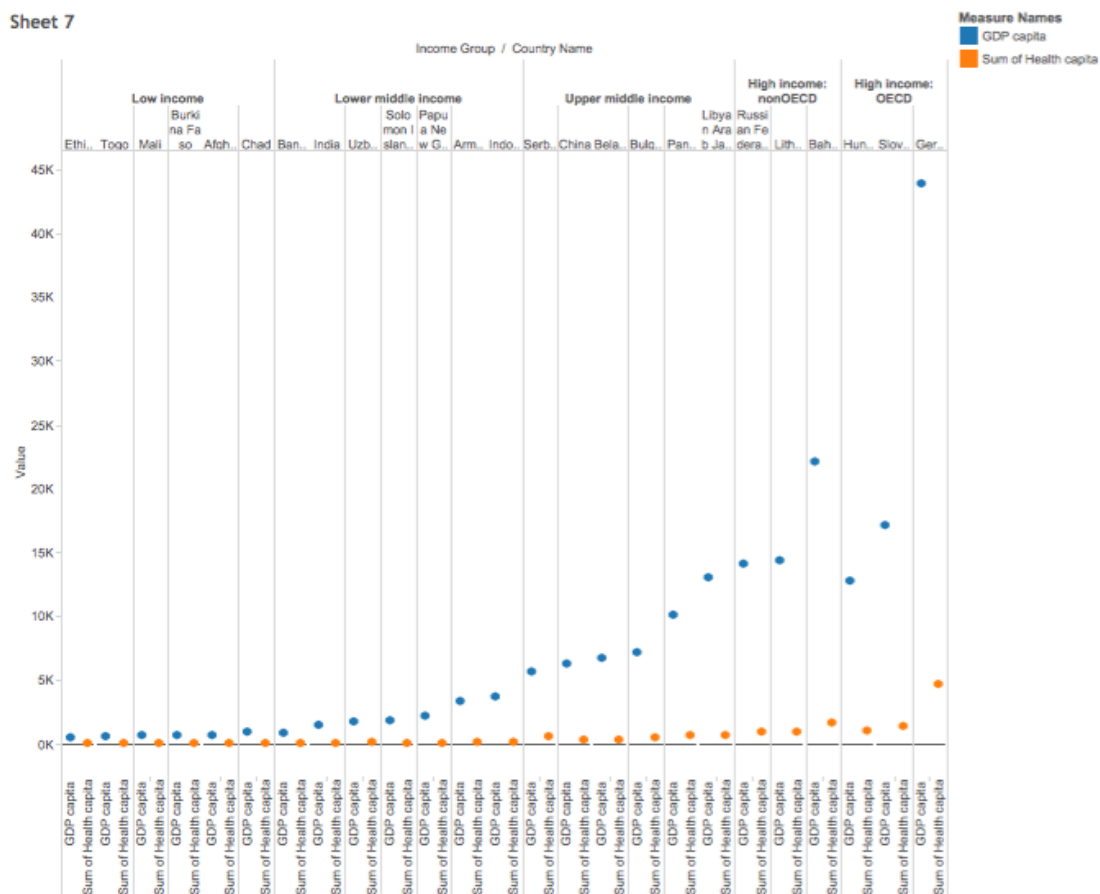
1. The first visualization depicted Health expenditure per capita and GDP per capita. These were each depicted in an ascending order, and compared. From the spearman correlation (0.97), and the visual observation, we can see that the relationship is proportional to quite an extent.

The slope of curve for GDP per capita is much higher than the health expenditure per capita.

In particular, India’s expenditure on health was observed and compared with some of the other countries. Afghanistan, with income group ‘Low income’ and a GDP per capita of 691, spent 58 usd on its citizens. India, with GDP per capita 1450, also spent 58 usd. But since the DALY is also an important factor to be considered, we look into it in the next visualization.

For this, side-by-side circles graph was used. I tried using side by side bar graph at first, but since the data range was too big, I found the circles to be better in terms of visual representation.

Figure 20: Variation of GDP per capita and Health expenditure per capita for different countries



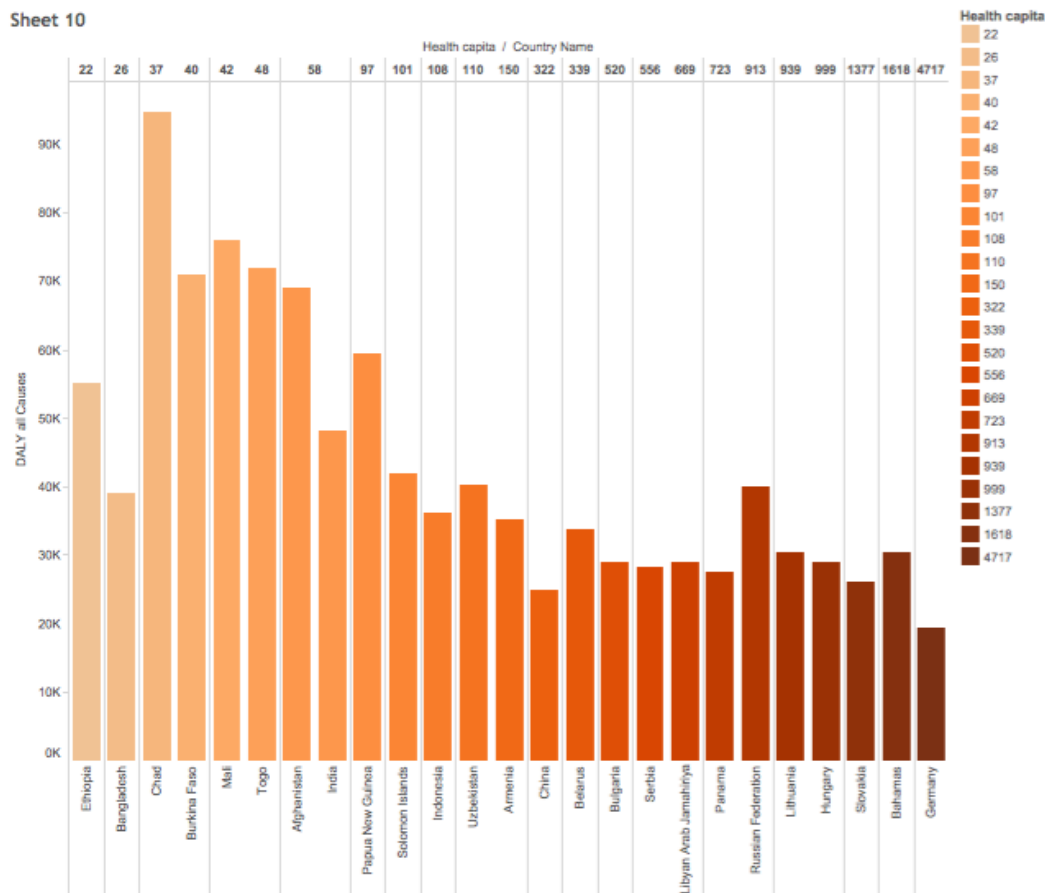
2. The second visualization depicted the relationship between DALY and Health expenditure per capita. Once again, we see a general trend in the relationship. Also, from the spearman value of (-0.86) we can conclude that they are inversly related. DALY seems to decrease, as the expenditure on health increases.

The slope of decrease in DALY isn't as high as the slope or amount of increase (not visual, but calculated from numbers) in Health expenditure.

In particular, India's DALY seems to correlate well with the amount spent. Most countries with expenditure more than India, have better values of DALY.

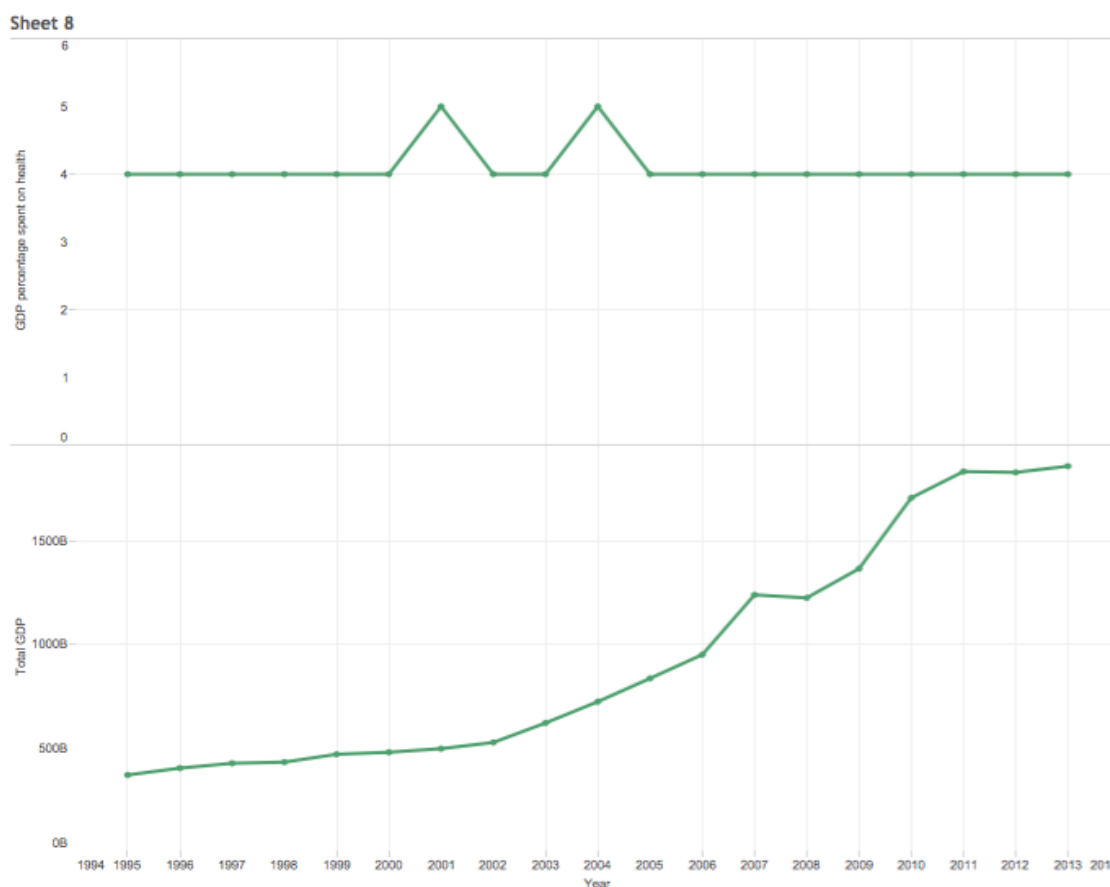
For this, bar graph was used. There was a gradient expressed through colours in the bar graph.

Figure 21: Variation of DALY and Health expenditure per capita for different countries



- The third visualization represents the slopes of GDP increase and increase in the GDP percentage on health for India. As one can see, the slope of GDP increase is much higher than percentage(which actually has a slope of almost zero)
For this, lines graph was used, as the data varies over time.

Figure 22: Variation over the years of total GDP and GDP percentage on health, India



11 Conclusion

For the answer to the first question, it was found that the GDP per capita correlates with the health expenditure (rank coeff 0.97). Some may argue that this reason justifies India's expenditure on health being less. But I don't think it justifies India's overall health budget percentage being slashed this year. I feel that India can spend a little more on its health budget. Since the GDP is increasing each year, it seems reasonable that the amount spent on health should also increase at least a little bit.

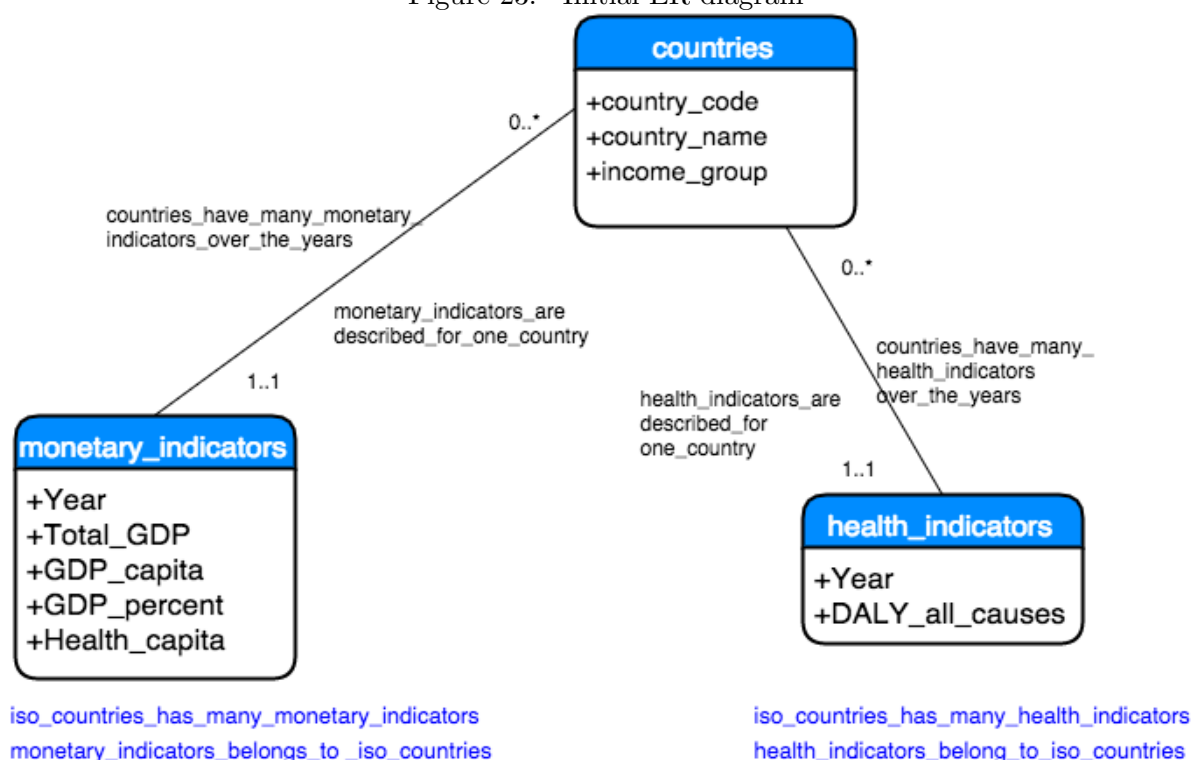
What is also surprising is that the slope of change of GDP per capita is much steeper than Health expenditure per capita. Perhaps all countries should think about how much they spend on health care. If anything, good public health adds to the happiness and economy of a country.

The answer to the second question is that as Health expenditure per capita increases, the DALY also decrease, ie, the health indicator of the country increases.

12 Lessons learned, challenges and future work

A great lesson I learned was that even though I planned the workflow of my project before I actually started working, I couldn't be completely faithful to it. This was due to data transformation, and changes in the ER diagram. Initially, I hadn't expected to face entity resolution issues. My ER diagram was much simpler. Please see below for an example

Figure 23: Initial ER diagram



In this diagram, it was assumed that countries were represented by only one name. But since countries have many alternative names, some extra data had to be accommodated. Hence, the current ER diagram that you saw in Figure 2 came to be.

Some of the other challenges I faced was with unicode decoding. I also realized the importance of working with a good and readily available data source, resolving issues along the way through others' help or through the internet, instead of trying to find an 'easier data source' to work with. More than anything, I am proud that I am no longer scared of Python or programming. I also realize that the work seems to 'go on'. Talking to Dr. Howison, and having group discussions really helped with dealing with challenges.

More than the data I am trying to analyse, which indeed is a topic very close to my heart, I am eager to learn more ways to handle data sets. In particular, I wish to improve the analysis methods I have chosen (Some of the techniques I have used right now aren't very concrete and I realize that). I want to start learning to use SciPy, as I think it would help me with data analysis. As I am from a humanities field, data and statistics are very important. I believe these skills will serve me well in the future