Project Report

Extract, Transform and Load

Formal Specification Document

1.1 Database - Introduction

For this exercise a structured database has been selected, postgresSQL, and pgAdmin 4. The reason being that a structured database provides the necessary functionality required to create and analyse the data. The formal structure of tables created within pgAdmin means that there is a lower chance of improper loading of data into the database by future users.

1.2 Database Identification

Database Name: econ_db

Table Name: Inflation
Primary Key: Country

Table Name: Foreign_Investment
Primary Key: Country

Table Name: Government_Debt
Primary Key: Country

1.3 Database Naming Conventions

As this database is created to store data used for historical economic analysis, the naming conventions for the tables are to be created in a descriptive and clean manner, with '_' as the placeholder for spaces between words. Capital lettering at the start of each word should be kept.

1.4 PostgresSQL Schema Information

```
 CREATE TABLE "Inflation" (

    "Country" VARCHAR PRIMARY KEY,

    "2001" INT   NOT NULL,
    "2002" INT   NOT NULL,
    "2003" INT   NOT NULL,
    "2004" INT   NOT NULL,
    "2005" INT   NOT NULL,
    "2006" INT   NOT NULL,
    "2007" INT   NOT NULL,
    "2008" INT   NOT NULL,
    "2009" INT   NOT NULL,
    "2010" INT   NOT NULL

);

CREATE TABLE "Foreign_Investment" (

    "Country" VARCHAR PRIMARY KEY,
    "2001" INT   NOT NULL,
    "2002" INT   NOT NULL,
    "2003" INT   NOT NULL,
    "2004" INT   NOT NULL,
    "2005" INT   NOT NULL,
    "2006" INT   NOT NULL,
    "2007" INT   NOT NULL,
    "2008" INT   NOT NULL,
    "2009" INT   NOT NULL,
    "2010" INT   NOT NULL

);

CREATE TABLE "Government_Debt" (

    "Country" VARCHAR PRIMARY KEY,
    "2001" INT   NOT NULL,
    "2002" INT   NOT NULL,
    "2003" INT   NOT NULL,
    "2004" INT   NOT NULL,
    "2005" INT   NOT NULL,
    "2006" INT   NOT NULL,
    "2007" INT   NOT NULL,
    "2008" INT   NOT NULL,
    "2009" INT   NOT NULL,
    "2010" INT   NOT NULL

);
```

1.4.1 Physical Design

Database Model

Figure 1

1.4.3 Physical Structure

The illustrations in *Figure 1* details how the tables will be structured. All three tables are constructed with a primary key (Country) to allow , if necessary, for any future relationship based tables to be created. The remaining columns range from 2001 to 2010, this is imperative to our future analysis needs.

The structures for new tables are created to keep unit and continuity with any future data amendments.

If new tables are created and analysis via modelling is to occur comparatively against the current tables, new tables must follow the same structures detailed above.

Country - VARCHAR

Year(2001-2010) - INT

2.0 Data Extraction, Transformation and Loading

2.1 Database - Creation

Initially start by creating the database inside of pgAdmin.

The name chosen for the database is ‘econ_db’.

NOTE: It is vital to take note of the access details such as username and password used to access pgAdmin - this is needed to update the config.py

2.2 Database - Table Creation

Run the commands located inside ‘postgresSQL_econ_db_schema.sql’ to create the tables needed. These table creation commands are also located in section *1.5 PostgresSQL Schema Information.*

2.3 Data Extraction

The data used is in csv format.

In order to extract the data from the csv files, Python is used and the library is Pandas.

Starting off by using the ‘read_csv’ function inside Pandas to read in the csv’s as Dataframes, the relevant data can be extracted specific to our needs. The years ranging from 2001 to 2010 are selected for the analysis.

2.4 Data Transformation

2.4.1 Annual Inflation and Foreign Direct Investment Datasets

- Narrow Down Dataset

In our Dataframe’s **annual_inflation_raw_df** and **foreign_dir_inv_raw_df** the extraction of the necessary data from the raw dataset is straightforward as the tables are already in the preferred format. By selecting the columns we require:

"Inflation, GDP deflator (annual %)","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010"

and

"Foreign direct investment, net inflows (% of GDP)","2001","2002","2003","2004","2005","2006","2007","2008","2009","2010"

This extracted data can be stored into a new Dataframe that will be used after the transformation to load into the live database.

- Update Column Names

Once the dataset has been narrowed down to include the years required, update the column names - primarily 'Inflation, GDP deflator (annual %)' and 'Foreign direct investment, net inflows (% of GDP)'. Both of these are to be changed to 'Country' as they both refer to the name of the country the relevant row provides data for in each table.

- Drop NaN

Once the naming conventions for all columns are correctly entered, we are able to further clean the data by dropping the NaN values. Both these datasets contain entries for countries where partial or no data is present. These rows aren't useful to us in our analysis and modelling hence opted to drop them entirely.

- Set Index

The primary key for the tables created in pgAdmin is 'Country' and therefore the index on both tables (annual_inflation_df and foreign_dir_inv_df) is set to 'Country'.
This is vital as it signals that the column Country to be the primary key when loading the Dataframe into the postgreSQL database using SQLAlchemy.

2.4.2 Government Debt Dataset

This dataset requires extra transformation as in its original state it does not meet our table row/column structure.

- Narrow Down Dataset

Start by selecting the appropriate columns need: "Country" , "TIME" and "Value"

- Format Table Layout

As the table is not in the preferred format, the pivot_table function is used to reconfigure the layout. The pivot_table function takes the tables rows and columns and stacks them to form a new table structured in a set out way. "Value" is used as the row data, "TIME" as the column names and "Country" as the initial column.

The index is reset using the reset_index function.

- Narrow Down Dataset - Years

Now that the table is in the preferred format, select the necessary columns (years) required to be included in the dataframe.

- Set Axis

Restructuring the table, in the current format, has thrown off the column axis and it will not be ready to load into postgresSQL. Use the function set_axis selecting all the appropriate columns.

- Set Index

Like the previous Dataframes the index for govt_debt_df is set to 'Country' so that it recognises the primary key correctly when loaded into the postgres database.

- Fill NaN with 0

The final data cleaning step is to fill all the NaN values present with a 0. This option was chosen as a majority of the dataset is whole with some slight exceptions. Instead of losing/dropping a row of data, filling any NaN values with 0 enables data to be retained whilst still meeting the requirements of the tables inside the Database when loaded into the postgres database.

2.5 Data Loading

As the Dataframes are now cleaned and transformed ready to be loaded into the postgreSQL database, the library SQLAlchemy is utilised.

- PostgreSQL Database connection via SQLAlchemy

Initially connecting to our postgres database using a connection string composed of our server's Host Name (host), Username (username), Password (password), Port Number (port) and Database name (database_name).

Once a connection to the postgreSQL server is made, inspect the Database and inspect the tables. This step verifies that the tables are visible and that our connection to the database is true.

- Load Dataframes

Finally, load the Dataframes created into their respective tables inside of our postgreSQL database.

- Verifying Tables

Open pgAdmin and run the following commands to verify the data exists in the tables created.

SELECT * FROM “Inflation”;

	Country [PK] character varying	2001 integer	2002 integer	2003 integer	2004 integer	2005 integer	2006 integer	2007 integer	2008 integer	2009 integer	2010 integer
1	Albania	3	3	3	6	3	2	2	4	2	3
2	Algeria	1	2	8	11	16	11	7	15	-11	16
3	Angola	108	121	103	43	26	13	13	20	-7	22
4	Antigua and Barbuda	-3	1	0	0	6	0	23	3	0	4
5	Argentina	-1	31	10	9	9	13	14	19	10	15
6	Armenia	4	2	5	6	3	5	4	6	3	9
7	Australia	5	3	3	3	4	5	5	5	4	0
8	Austria	2	1	1	2	2	2	2	2	1	2
9	Azerbaijan	3	3	6	8	16	11	21	28	-19	14
10	Bahamas	0	4	1	1	5	1	3	1	-2	1

SELECT * FROM “Foreign_Investment”;

	Country [PK] character varying	2001 integer	2002 integer	2003 integer	2004 integer	2005 integer	2006 integer	2007 integer	2008 integer	2009 integer	2010 integer
1	Afghanistan	0	1	1	3	4	3	3	3	1	0
2	Albania	5	3	3	5	3	4	6	7	8	9
3	Algeria	2	2	1	1	1	2	1	2	2	1
4	Angola	24	15	25	7	-4	0	-1	2	3	-4
5	Antigua and Barbuda	12	8	19	9	22	31	26	13	6	8
6	Argentina	1	2	1	3	3	3	2	3	1	2
7	Armenia	3	5	4	7	5	7	8	8	9	6
8	Australia	2	4	2	6	-5	4	5	4	3	3
9	Austria	3	0	3	1	27	1	17	2	3	-7
10	Azerbaijan	4	22	45	41	13	-3	-14	0	1	1

SELECT * FROM “Government_Debt”;

	Country [PK] character varying	2001 integer	2002 integer	2003 integer	2004 integer	2005 integer	2006 integer	2007 integer	2008 integer	2009 integer	2010 integer
1	Australia	10	9	8	7	6	6	5	5	8	11
2	Austria	61	60	61	62	62	60	58	59	65	66
3	Belgium	99	98	95	93	92	88	85	90	95	97
4	Canada	40	38	36	32	30	28	25	29	36	36
5	Chile	15	16	13	11	7	5	4	5	6	9
6	Czech Republic	15	16	19	21	23	25	25	27	32	37
7	Denmark	52	52	50	47	39	33	28	32	38	40
8	Estonia	3	4	3	3	2	2	1	2	4	3
9	Finland	44	41	44	42	38	36	31	29	38	42
10	France	48	50	52	53	53	52	52	53	61	67