#### **Project 2 - Extract Transform Load**

|  |
| --- |

## ***Project Description***

In 2020-2021, the most important global issue has been the COVID-19 pandemic. Health professionals and researchers around the world have been working hard to make available the data related to cases, testing, and mortality. We are utilizing publically available Covid-19 data for our ETL project.

**At the end of ETL process, the prepared data would have answered these questions and more:**

* "Total reported cases" and "Total reported deaths" per county, state
* New cases and new deaths
* Case Fatality Rate (No. of confirmed positive cases vs no of deaths)
* Covid Mortality Rate (No. of confirmed positive cases vs no of covid deaths)
* Positive Cases per capita
* Flu cases as compared to Covid cases

**Brief introduction to ETL**

We are living in a world of data. The volume of data that is being generated and collected continues to increase at an exponential rate. As the amount of data grows, the importance of making use of that data grows as well. This data is immensely valuable for analytics, data science, and machine learning, not only to present valuable information and noticeable trends but also to derive business insights and predictions. The data that is being captured and stored cannot be used in its raw form. Processing the raw, messy data into clean, consistent and reliable data is a critical step before it can be used.

ETL, which stands for Extract, Transform, and Load, is the process used to

* Extract data from various sources
* Transform the data into a clean, usable format
* Load the data into target database systems that end-users can access and use for further analysis, to make business decisions and to solve problems.

How ETL works : Lets understand each step of the ETL process and how it relates to our project

## ***Extract:***

Raw data is extracted from various data sources, which can be structured or non structured. These sources can include but are not limited to:

* APIs, JSON, CSV, XML files
* RDBMS(MS SQl Server, MySql etc.) or NoSQL servers (MongoDB etc.)
* CRM (Salesforce Sales Cloud, Quickbase etc.) and ERP systems(Katana, Oracle NetSuite etc)
* Web pages

## **How we 'Extract'ed:**

## **Data Sources**

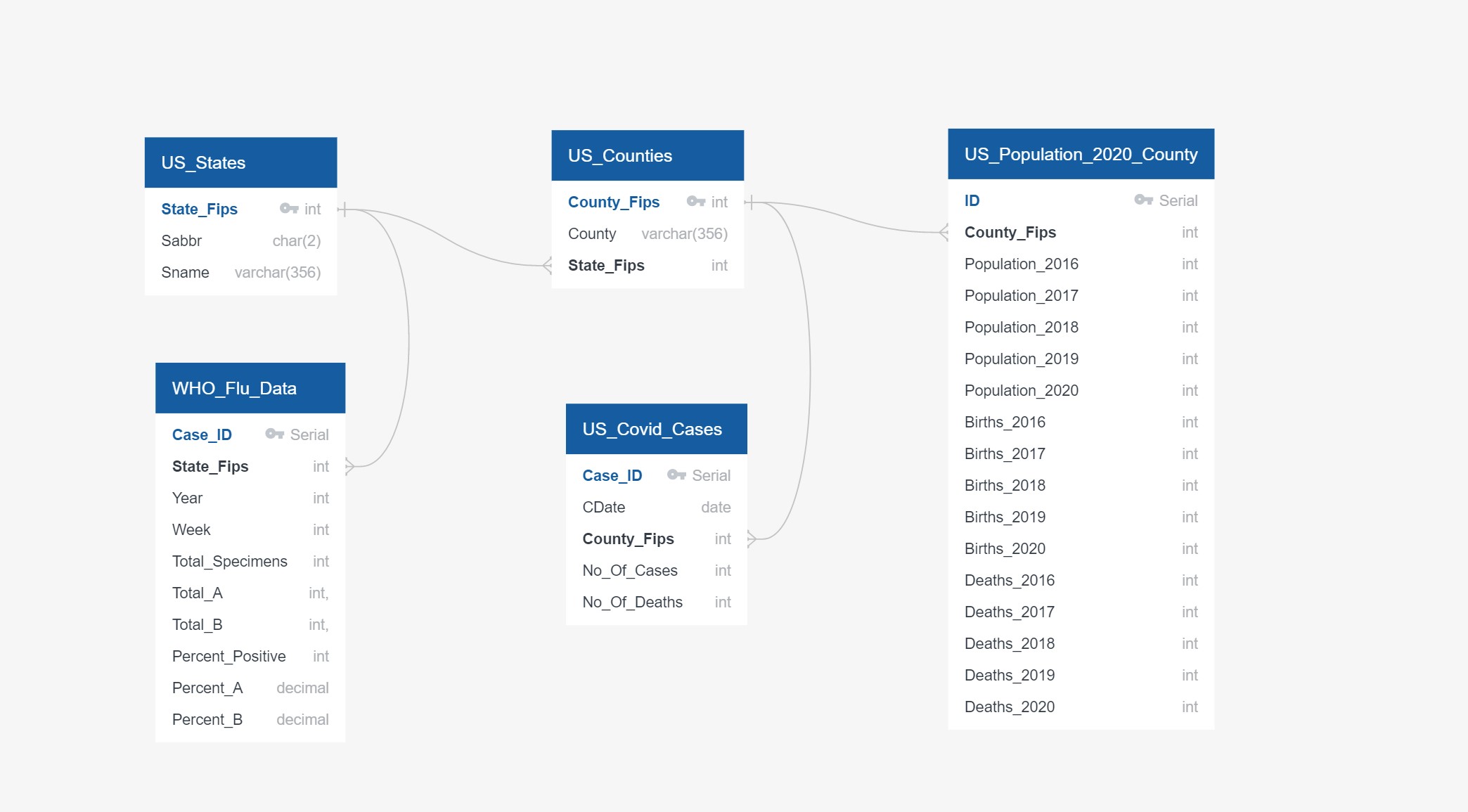
* Covid 19 Data raw feed : <https://raw.githubusercontent.com/nytimes/covid-19-data/master/us-counties.csv>
  + New York Times has collected a large amount of COVID-19 data for the United States, and they have made this data publicly available. We are connecting and extracting data by connecting to their raw feed.
* Census Data : <https://www2.census.gov/programs-surveys/popest/datasets/2010-2020/counties/totals/>
* State FIPS : <https://www.census.gov/library/reference/code-lists/ansi/ansi-codes-for-states.html>
* County FIPS : <https://www.nrcs.usda.gov/wps/portal/nrcs/detail/national/home/cid=nrcs143_013697>
* Flu data for Comparison: <https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html>

## ***Transform:***

The second step consists of transforming the raw data that has been extracted from the above-mentioned sources into a format that can be used by different applications. This can be compared to a staging area where the data gets cleansed, mapped, and transformed, often to a specific schema, so it meets operational requirements. This process can involve any of these methods:

* Formatting the data into tables or joined tables
* Performing calculations or summarizations
* Converting data to correct data types
* Deduplication
* Conducting tests/audits to ensure data quality and compliance
* Encrypting, or protecting data governed by industry standards or government regulations

### **How we 'Transform'ed:**

* After studying the raw data and identifying the columns from each table, created an Entity Relationship Diagram (ERD) to view a snapshot before rearranging data into required schema
* Prior to inputting the data into PostgreSQL, we formatted the data using pandas in Python to reorganize and merge essential dataframes specifically catered for our final tables in PostgreSQL(via PgAdmin)
  + Ex. We merged the counties and state dataframes on their state abbreviations in order create one dataframe with both state and county FIPs
* Using the strip function, we systematically got rid of any extra space within the cells for each table to maintain consistency and prevent any issues when transforming the data to its final version.
* After renaming, merging, and selecting only the essential columns, proceeded to export all data frames as CSVs to be imported into PostgreSQL.
* In PgAdmin, we created a database named “ETL” in which we use the query tool to build all the necessary tables for our project. These tables
* We identified, cleaned, formatted and redistributed the data that we gathered from multiple datasets into the new PostgreSQL database, making sure it was executed in a sequence that data is loaded into relational tables accurately.

**Detailed Transformations**

**Table 1 : US\_States**

* Read from source to Pandas Dataframe
* Trimmed Leading and Trailing Spaces
* Made sure There were no duplicates
* Identified 'State\_Fips' as primary key
* Wrote to US\_States.csv that can be imported in PostgreSQL

**Table 2 : US\_Counties**

* Read from source to Pandas Dataframe
* Trimmed Leading and Trailing Spaces
* Made sure there were no duplicates
* Assigned State\_Fips to each record by merging with States Dataframe using state abbreviation to compare
* To handle the issue of incoming data from sources with unknown counties but known States, created a unique code based on the state\_fips and assigned StateName as County Name. This observation came from the US Census Data Table from where we used the same process for our county table.
* Created a new dataframe with 'County\_Fips','County','State\_Fips' columns
* Identified 'County\_Fips' as primary key
* Identified 'State\_Fips' as foreign key
* Wrote to Us\_Counties.csv that can be imported in PostgreSQL

**Table 3 : US\_Covid\_Data**

* This was the data that required most transformation
* Read from source raw feed to Pandas Dataframe
* Checked for Nulls in any columns
* If there are records that have a valid state but an unknown county, kept the records and assigned the fips\_county for these records (with the special id's mentioned above in the county table)
  + Column 'deaths' can contain Nulls but not Fips
  + Identified these records that has Fips as Null
* Moved them to a seperate Dataframe
* Trimmed Leading and Trailing Spaces for required fields
* Merged with county table and assigned the County\_Fips
* Created a dataframe for records with No Nulls
* Concatenated the clean dataframes
* Replace Null with zeros for column 'Deaths'
* Converted fips and deaths columns to Int
* Selected fields 'date', 'county\_fips', 'cases', 'deaths' for final dataframe
* Wrote the final dataframe to US\_Covid\_Data.csv that can be imported in PostgreSQL

**Table 4 : Us\_Census\_Data**

* Read from source to Pandas Dataframe
* The original table had 150+ columns
* Removed records with for State Totals as they are redundant
* State totals can be calculated from county information
* Created Fips\_County Field using State and County Fields that were both stored separately as integers
  + converted state and county fields from int to str
  + added leading zeros to State and County get the format required for fips\_county
  + concatenated state and county to make County\_Fips Code that follows the Fips\_County format
* created final dataframe with selected fields
  + 'fips','POPESTIMATE2016','POPESTIMATE2017','POPESTIMATE2018','POPESTIMATE2019', 'POPESTIMATE2020', 'BIRTHS2016','BIRTHS2017','BIRTHS2018','BIRTHS2019','BIRTHS2020', 'DEATHS2016','DEATHS2017','DEATHS2018','DEATHS2019','DEATHS2020'
* Wrote to US\_Census\_Data.csv that can be imported in PostgreSQL

**Table 5 : WHO\_Flu\_Data**

* Read from source to Pandas Dataframe
* Dropped a column not required
* Records with missing data came in with 'X' instead of Null. Replaced 'X' with -1 as they were numeric fields in the destination table.
* Trimmed leading/trailing spaces for columns that are used in join conditions for merging dataframes
* Added State\_Fips
  + 'State\_Fips','YEAR','WEEK','TOTAL SPECIMENS','TOTAL A','TOTAL B','PERCENT POSITIVE', 'PERCENT A','PERCENT B'
* Wrote to WHO\_Flu\_Data.csv that can be imported in PostgreSQL

## 

## 

## 

## 

## 

## 

## ***Load:***

This last step involves moving the transformed data to a target data warehouse. Initially, the final data is loaded once, and thereafter periodic loading of data happens to keep the database up to date. Most of the time the ETL process is automated and batch-driven. Typically, ETL is scheduled to trigger during off-hours when traffic on the source systems and the destination systems is at its lowest.

## **How we ‘Load’ed:**

* After structuring the data frames and exporting as CSV to a designated files, we were able to import the CSV to the defined tables in PgAdmin using the import tool
* For certain tables, we adjusted the variable types to match between the CSV as well the tables to create the final loaded data.

Detailed Procedure for Loading

* RDBMS Used : PostgreSQL
* Created a PostgreSQL database 'ETL'
* Followed ERD created earlier, developed SQL DDL queries with correct data types, primary keys, foreign keys, and other constraints
* Created tables making sure they were created in correct order to handle foreign keys.
* Executed these queries on pgAdmin to create tables
* Imported each CSV file into the corresponding SQL table making sure data is imported in the same order that the tables were created and account for the headers when importing to avoid errors.
* Created a view with most used tables for easy querying of data
* Examples of some queries we executed to test Data:
  + Latest Covid19 numbers for All States

**Purpose**

With COVID-19 affecting close to every aspect of our lives, directly or indirectly we can use this particular data to inquire about certain questions/topics within the field of medicine:

* **Have the births increased or decreased during the time of Covid**
* **Case positivity Rate**
* **Comparison of covid with other seasonal viruses**
* **Effects of covid on hospital and ER visits for non covid related health issues**
* **Effects of covid on on going treatments for chronic illnesses**
* **Effect of covid on mental health**