**Project Background**

This project delves into the utilization of MySQL in the context of data transformation, exploratory analysis, and automation, focusing on its application in handling the US Household Income dataset to derive meaningful insights and identify general patterns in the data. These patterns include outliers and features of the data that might be unexpected.

My role is a data analyst that uses MySQL to help the audience understand the data better by showing them tables that they want to see. The stakeholders are the general population that are curious about the salaries and geography of States and Cities and other insights that could also be found.

With the stakeholders in mind, I decided to showcase to them an up-to-date exploratory analysis of the data showing states and cities that have the highest average income and other categories that could be insightful.

**Summary & Key Insights**

This project outlined the utilization of MySQL for data transformation, exploratory analysis, and automation for data cleaning. Data transformation involved cleaning inconsistencies, null values, and missing values. Exploratory data analysis unveiled income distribution across states, income types, and cities. Finally, an automated system using events, stored procedures and triggers was implemented to clean data at specific intervals and upon data insertion.

This project leveraged MySQL to explore a dataset of US household income and geographic locations. Key findings include Texas having the largest land area and Michigan boasting the most water area. Analyzing income data, we discovered the District of Columbia has the highest average income within the top ten states, but also the lowest median income. Boroughs hold the top spot for average income by type, though with a low count. Tracks, on the other hand, have a significantly higher count but a much lower average income, hinting at possible data errors. Further analysis revealed Delta Junction City with the highest average salary, but there's a chance the data has a cap of 300000 for median salaries.

Overall, this project showcased the potential of MySQL for data exploration and management, uncovering valuable insights from datasets. However, the project acknowledges limitations in data quality and the need for potential further cleaning based on domain knowledge.

**Dataset Overview**

The dataset originally developed for real estate and business investment research. Income is a vital element when determining both quality and socioeconomic features of a given geographic location.

The database contains records on US Household Income Statistics & Geo Locations derived from over +36,000 files and covers 348,893 location records around the United States. The two datasets are Household & Geographic Statistics and Geographic Location. Household & Geographic Statistics have the statistical computations of the household incomes while the Geographic location have the geographic information.

**Data Preparation**

The datasets that were imported to MySQL are named as ushouseholdincome for the Geographic Location and ushouseholdincome\_statistics for the Household & Geographic Statistics under the ‘project’ schema.

Household & Geographic Statistics:

* State Name (character)
* Mean Household Income (double)
* Median Household Income (double)
* Standard Deviation of Household Income (double)
* Number of Households (double)

Geographic Location:

* Longitude (double)
* Latitude (double)
* State Name (character)
* State abbreviated (character)
* State\_Code (character)
* County Name (character)
* City Name (character)
* Name of city, town, village or CPD (character)
* Primary, Defines if the location is a track and block group.
* Zip Code (character)
* Area Code (character)
* Square area of land at location (double)
* Square area of water at location (double)

I conducted an initial exploration on the two datasets using the following queries, that will also be used to check the data after each query update.

A white background with black text

Description automatically generated

**Data Transformation**

Before transforming the data, I made a backup copy of the dataset csv file in case of an error.

Upon checking the tables, one of the columns in ushousegoldincome\_statistics.csv is in written as `ï»¿id`, so I wrote the following query to rename the column name to `id`.



A screenshot of a computer

Description automatically generated

*After changing the column name*

After checking the column names, I run a count on both tables to check for verification. The ushouseholdincome table returned with 32292 while ushouseholdincome\_statistics table returned with 32526. Since the difference is only minimal, I chose to ignore the difference.

To check for duplicates, I used the following query on both tables to check the column id count that is more than one.

A screen shot of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

*Duplicate count*

The ushousehold income returned 6 rows that are duplicates while the ushouseholdincome\_statistics returned 0 duplicates. Seeing as there are duplicates in ushouseholdincome, I used the following query to give the duplicated rows a unique id using row\_number() to delete them.

A computer screen shot of a program code

Description automatically generated

To check for inconsistencies in the data, I used the following format to select the column, group the column and get the count.

A close-up of a computer code

Description automatically generated

A screenshot of a computer

Description automatically generated

*Data inconsistency ‘georia’*

In the SELECT clause, replace ‘column\_name’ with a column name in the dataset to check each column for inconsistencies. In the FROM clause, replace ‘table\_name’ with the table you want to check. One of the inconsistencies I encountered is in the State\_Name column where the field inputted is ‘georia’, so I queried the following to replace the state name to ‘Georgia’. I did the same procedure for all other inconsistencies in the dataset.

A screenshot of a computer

Description automatically generated

To check for missing values and null values, I used the following format to select the columns to be checked and filter the null and missing values in the columns.

A white background with blue text

Description automatically generated



*Missing data in the Place column*

In the WHERE clause, replace ‘table\_name’ with a column name in the dataset to check each column for missing values and null values. In the FROM clause, replace ‘table\_name’ with the table you want to check. I found a missing value in the column ‘place’ and from exploring and analyzing the dataset, I deduced to replace the value with 'Autaugaville', in line with the data values close to it. I used the following query to filter by county and city to accurately select the row with the missing value and replace it with 'Autaugaville'.

A computer screen shot of text

Description automatically generated

There are some inconsistencies, null values, and missing values that can still be seen in the dataset, but I chose to just filter them out when doing the exploratory analysis. I prefer to not delete the null values and missing value with the idea being that the dataset could be updated in the future to fill in the missing data. I also lack the domain knowledge when it comes to fixing the remaining inconsistencies and I would need to consult someone who has the appropriate domain knowledge before I could proceed transforming the data further with confidence.

**Exploratory Data Analysis**

The goal of this project is to explore the data and derive any interesting insights and identify general patterns.

I wanted to start looking at the top ten largest land area and top ten largest water area by state.

A screenshot of a computer code

Description automatically generated

**A screenshot of a computer

Description automatically generated** **A screenshot of a computer

Description automatically generated**

*Top ten land area and water area by state*

As expected, Texas has the largest land area and Michigan has the largest water area.

To analyze the income data together with the geolocation data, I needed to join ushouseholdincome and ushouseholdincome\_statistics. The primary key and foreign key used to join the two tables together are the ‘id’ column from both tables. I used inner join to join the two tables. Doing so automatically filter’s out the missing rows that was mentioned previously when I did the count of all the rows for each table. To further filter out the missing values and null values, I added a WHERE clause.

A screenshot of a computer code

Description automatically generated

Further analyzing the data, I wanted to explore the top ten average income by states.

A screenshot of a computer

Description automatically generated A screenshot of a data

Description automatically generated

*Average income by state*

The state with the highest average income is the District of Columbia, but the median income is the lowest out of all states in the top ten.

Next I analyzed the top ten average income by type, but I had to add an additional filter, the HAVING clause, to filter out types that had a low count. Without the filter, the query comes back with outliers skewing the results. I suspected that there could be data errors in the types but as mentioned before, I do not have the domain knowledge to be confident enough to make changes.

A screenshot of a computer program

Description automatically generated

A screenshot of a computer

Description automatically generated  
*Average Income by type*

Borough has the highest average income in terms of type, but it does have a low count. Track has a significant number of counts, and the average income is far from Borough’s average income.

Analyzing the data further, I wrote a query to check the average salaries by city.

A screenshot of a computer program

Description automatically generated

A screenshot of a data

Description automatically generated

*Average salaries by city*

Delta Junction City has the highest average salary with a median salary of 300000. This is an odd finding since other rows also show similar values. I assumed that there is a cap of 300000 for the median salaries in the data.

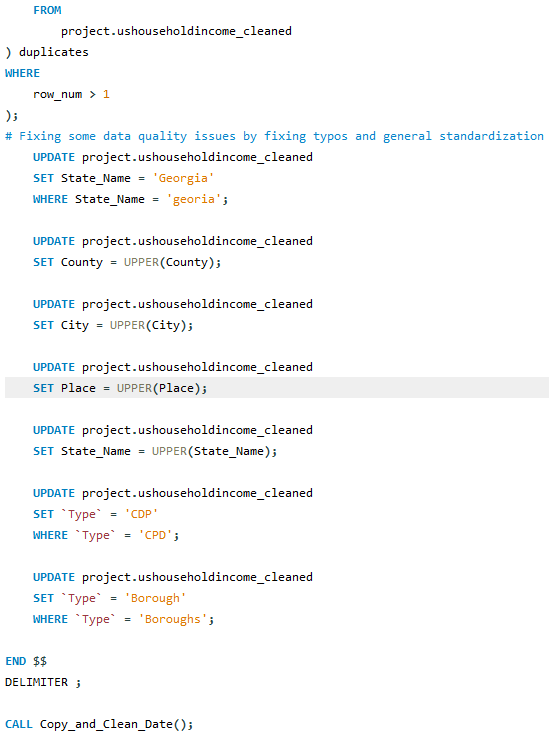
**Data Cleaning Automation**

In a scenario where the database gets updated, using a data cleaning automation helps with cleaning data at a specific period. Using the previous queries used in data transformation, I built an automation system using events and stored procedures. This would prove useful for dynamic data that is constantly changing.

I dropped the previously worked on ushouseholdincome dataset because it was already transformed and imported the original ushouseholdincome dataset from the backup copy to use to test the data cleaning automation.

Inside the stored procedure query, first, we create a new table called ushouseholdincome\_cleaned. This table mirrors the structure of an existing table named ushouseholdincome. We copy data from the original table into the new one, ensuring that each row includes a timestamp indicating when the data was inserted. Next, we address data quality issues. We correct misspelled state names (e.g., changing “georia” to “Georgia”), convert text values to uppercase for consistency (county, city, place, and state names), and standardize the Type column values (e.g., “CPD” becomes “CDP” and “Boroughs” becomes “Borough”). Overall, this procedure enhances data consistency, removes duplicates, and improves data quality for further analysis.

A screen shot of a computer program

Description automatically generated 

To run this stored procedure in a schedule, an event is needed named run\_data\_cleaning. This event is scheduled to run every 30 days. When triggered, it executes the stored procedure Copy\_and\_Clean\_Data(). The purpose of this event is likely to automate data cleaning tasks at regular intervals. It testing, I set the schedule every 2 mins.

A white background with black text

Description automatically generated

A trigger is created named Transfer\_clean\_date. It fires after an insert operation (i.e., when new data is added) on the specified table. When triggered, it calls the same stored procedure, Copy\_and\_Clean\_Data(). Essentially, this trigger ensures that data cleaning occurs whenever new records are inserted into the ushouseholdincome table.

A screen shot of a computer program

Description automatically generated

The provided SQL codes establishes an event and a trigger to maintain data quality and consistency in the ushouseholdincome dataset.

**Tools Used**

MySQL was used for the queries and code.