

Database Project Report

Database Management for Data Scientists, FS24

United States County Level Income Response to Corporate Taxation

Submitted by The Federal SQL Consultants (formerly "ACA")

Members:

Aishwarya Patil (aishwarya.patil@stud.hslu.ch)

Carlos Leon (carlos.leon@stud.hslu.ch)

Alec Vayloyan (alec.vayloyan@stud.hslu.ch)

Submitted for 16.06.2024

Table of Contents

1. Introduction & Context.....	3
2. Project Idea & Use Case	3
2.1. Decision support for value Generation	3
2.2. Calculation of key figures used for decision support	3
2.3. Data sources used	4
2.3.1. Data Quality	4
2.4. Database technology with rationale.....	4
2.5. VM / MYSQL	5
3. Data Model & Database Schema.....	5
3.1. Data model according to ER technique v. Chen (1976)	5
3.2. Database schema in database DDL Syntax + Relationship Between Model and Schema	6
3.3. Verbal description of the most important data classes and attributes.....	9
3.4. Normalization.....	10
4. Loading & Transforming the Data	11
4.1. Data loading	12
4.2. Transforming	13
5. Analyzing & Evaluating Data	14
5.1. Queries in database syntax	14
5.2. Show connection between queries and use case	16
6. Query Optimization.....	16
7. Visualization & Decision Support	20
8. Conclusions & Lessons Learned	22
8.1. Business Conclusion:.....	22
8.2. Lessons Learned	23
9. Appendix.....	24
9.1. AI Disclosure.....	24
9.2. Login Information	24
9.3. Self-Assessment	25

1. Introduction & Context

Increasing household income is a crucial policy goal for governments due to its positive impacts on individuals and society. Higher average income levels lead to an improved material standard of living, enhanced personal security, and increased tax revenue for government. It is therefore an important aspect on which voters judge politicians. To achieve higher incomes, politicians often propose business-friendly policies based on the concept of trickle-down economics. This theory suggests that allowing corporations to retain more profits and operate with greater freedom benefits everyone. While most of the increased economic value may go to company managers and shareholders (only 21% of American households company stock), proponents argue that the broader population can still gain from fostering corporate success.

One key benefit is that companies are attracted to locations with favorable business environments, especially those with low **corporate tax rates**. This relocation leads to local employment growth and economic stimulation, particularly in smaller towns and cities that often rely heavily on a single major employer. Furthermore, with more money to invest in new products this effect is compounded. Not only are companies attracted to a location but the increased investment means more is produced at these locations than would otherwise be. Opponents argue that the broader population will not meaningfully benefit from giving companies lower taxes in a region. They say companies will just pay back increased dividends to shareholders and at best, move the “mailbox” to a low tax region. As such, policies designed to increase household income may suffer from **distributional differences**, meaning that the policy’s effects do not increase all groups if income the same. According to opponents, the wealthiest would get wealthier while the poor would suffer from underfunded social services (which require tax revenue).

Our client, a **state government** in the United States, is considering updating their tax code to be more business-friendly to facilitate an increase in income levels and wants to know not only what kind of effects would there be on the overall county average wealth, but also if there are distributional issues that should be taken into account. **The key question is this: Do areas within a state (counties) with differing wealth levels benefit (or not benefit) equally from changes in the corporate taxes at the state level?** Henceforth, when we refer to the persona government / client, **we are referring to someone within this organization with the power to make changes.**

2. Project Idea & Use Case

2.1. Decision support for value Generation

The database will be able to generate value insight regarding relationships and optima around factors of corporate taxation if the goal is increasing household income, while also considering *where* these increases or decreases in household income are happening. Unfortunately, tax data is available only at the state level while incomes should be evaluated at the county level. Looking at the county level incomes allows us to separate counties according to their levels of wealth. Meaning distributional differences are considered, something which is often not done in studies evaluating policies. **This insight allows decision makers to make data driven choices, generating value.**

2.2. Calculation of key figures used for decision support

The key figure will be a table showing the average household wealth of counties of different wealth levels for different levels of corporate taxation. The hope is that distributional issues as well as optima can be discovered, generating significant value for politicians who can make more educated choices.

2.3. Data sources used

Table 1 below shows an overview of variables and respective sources used

Variables	Source	Notes
<u>Per US State in 2020:</u> Property Tax Rank Unemployment Insurance Tax Rank Sales Tax Rank Individual Income Tax Rank Individual Income Tax Rank Corporate Tax Rank	Tax Foundation State Economic Freedom Ranking 2020 (Think Tank)	The think tank's mission is political, however their <u>relative</u> ranking of US States should be accurate.
<u>Per US County in 2021</u> FIPS Code ¹ Area Name State Median Household Income Unemployment rate Number unemployed annual average	Economic Research Service (Governmental Agency)	Government Census data, this is the gold standard.
<u>Per City in Jan 2024</u> FIPS Code Population Density Military (T/F) Incorporation (T/F) Ranking (1-5)	Simplemaps.com (Commercial)	Free demo account used. Ranking: refers to a score generated by the authors on the city's importance nationally.

Table 1: Description of Variables and Sources used in Project

2.3.1. Data Quality

The data set has a number of missing values, particularly for Alaska and Puerto Rico. These are dealt with by removing them. While this does involve a loss of data, these states have so few counties anyways, they could not be reliably imputed by taking standard imputation methods anyways.

2.4. Database technology with rationale

We live in volatile times and need to be able to update information to reflect changes. Such data may not be available at the desired levels, so having a database allows analysts to update at different levels. For example, COVID-19 data was available at county levels but transport data (like density of airports) is only available at the state level oftentimes. Such adaptability is the main reason to use DB technology over another tool such as statistical programming languages or Excel. The initial setup may be more complicated but in the long term it is more resilient to more information being added.

The **DBM Does provide intrinsic advantages** not present in other solutions: such as consistency checks, efficiency benefits, uniform retrieval and multiuser access. All of which can facilitate the generation of **value** through transforming information to optimize decision making.

¹ FIPS: Federal Information Processing System. A unique code for each county of the US

2.5. VM / MYSQL

We will use the VM provided to us by the lecturers and the MYSQL language which combines both pure SQL and Not Only SQL (NOSQL).

3. Data Model & Database Schema

3.1. Data model according to ER technique v. Chen (1976)

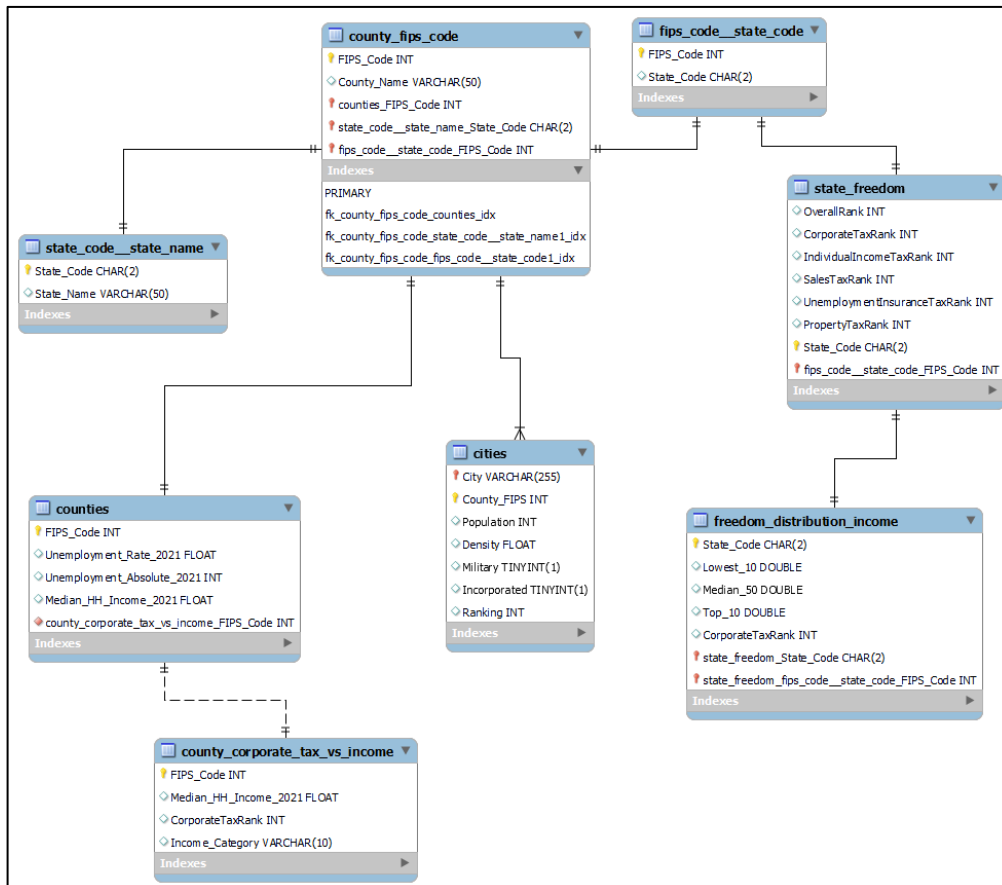


Figure 1: ER Diagram

—	One
— <	Many
—	One (and only one)
— ○	Zero or one
— <=	One or many
— ○<=	Zero or many

Symbol	Meaning
	Primary Key
	Attribute
	Relationships, not relevant for Chen Notation (ignore in diagram)

Figure 2: Cardinality²

Table 2: Key Symbols ER

² Credit: [LucidCharts](#)

Figure 1 shows the Entity Relationship Diagram of the DB. It is important to note the cardinality. All relationships between keys are 1:1, except for the cities, where one county can have multiple cities in it. For the sake of space the **relationship** between tables are written in text here:

All relationships are of **type** “relates to”, since they just add information but do not signify any conceptual difference in the relationship. With the exception of: cities, which “belong to” counties.

3.2. Database schema in database DDL Syntax + Relationship Between Model and Schema
Disclaimer: Although these scripts are displayed in the order they were run, the Loading part of the process is delved into later in this report but takes place in between some of the table creation steps. This is why some snippets may appear as if some data is already loaded, because it was. Figures in Tables have no captions, the text related the image in the same box.

In this section, we will go over how each table in the database was created, as well as the purpose for each of them, the steps are described in Table 3 below.

County_FIPS_Code	
<pre>5 # TABLE 1: COUNTY FIPS AND COUNTY NAME 6 CREATE TABLE County_FIPS_Code(7 FIPS_Code int primary key, 8 State_Code char(2), 9 County_Name VARCHAR (50) 10);</pre>	<ul style="list-style-type: none">- This snippet creates the first table in our database, which will house the FIPS Code as its primary key, the State Code, which is an abbreviation of the states’ names, and the County name.- We will be able to use this table further on to create relationships between other tables, as well as for querying for data housed in separate tables that may be accessible through these variables.
<pre>22 UPDATE County_FIPS_Code 23 SET FIPS_Code = 1000 24 WHERE FIPS_Code = 0;</pre>	<ul style="list-style-type: none">- We were having issues loading in the first row of data to this table, perhaps because of the separator that was used in the .csv file, so we needed to find a workaround to include our entire dataset in the loading process.- After some investigation, we realized only the first row was missing from the dataset, so we decided to manually add it.
State_Code__State_Name	
<pre>27 # TABLE 2: STATE CODE AND STATE NAME 28 CREATE TABLE State_Code__State_Name AS 29 SELECT State_Code, County_Name 30 FROM County_FIPS_Code 31 WHERE FIPS_Code%1000 = 0;</pre>	<ul style="list-style-type: none">- In this table, we are interested in creating a relationship between State Codes and State Names. The FIPS Codes are standardized so that each state has a FIPS code of its own, and a state’s FIPS code is always a multiple of 1000.

- This is why we have used the modulo operator to only include FIPS codes that are multiples of 1000, as to only include state names in this table

```
33 • ALTER TABLE State_Code__State_Name CHANGE County_Name State_Name varchar(50);
```

- Previously we used another table's "County_Name" column to fill in this table, but since we are only including state names in this table, we change the column's name for clarity.

```
36 • delete from County_FIPS_Code where FIPS_CODE % 1000 = 0;
```

- Now that we have the States with their FIPS codes in a separate table, we can remove them from the County_FIPS_Code table using the same criteria (%1000) as before. We do this to avoid redundancy, and to keep county and state data separate.

```
103 • Alter table state_code__state_name add primary key (State_Code);
```

- Later, we add State_Code as the primary key for this table

FIPS_Code__State_Code

```
40 #TABLE 3 FIPS CODE - STATE CODE
```

```
41 • CREATE TABLE FIPS_Code__State_Code AS
```

```
42 SELECT FIPS_Code, State_Code
```

```
43 FROM County_FIPS_Code;
```

- This table is meant to establish a relationship between the FIPS Code and the State Code, to be used for querying and harnessing data from other tables.

- The states' FIPS codes, names and state codes have been separated into two tables to avoid risking loss of data.

```
45 • Alter TABLE FIPS_Code__State_Code
```

```
46 ADD PRIMARY KEY (FIPS_code);
```

- The FIPS Code is set as the primary key for this table. This will ensure that execution is faster by avoiding having to do a full table scan when looking for data, given the relationships that have been created.

```
50 • alter table County_FIPS_Code drop column State_Code;
```

- Since we added a "copy" of this column as a primary key, it is not necessary to keep it, as it would be a duplicate, so we drop it from the table.

State_Freedom

```
9 # TABLE 4 STATE FREEDOM
```

```
10 • CREATE TABLE State_Freedom (
```

```
11 State varchar(255) PRIMARY KEY,
```

```
12 OverallRank INTEGER,
```

```
13 CorporateTaxRank INTEGER,
```

```
14 IndividualIncomeTaxRank INTEGER,
```

```
15 SalesTaxRank INTEGER,
```

```
16 UnemploymentInsuranceTaxRank INTEGER,
```

```
17 PropertyTaxRank INTEGER);
```

- The State_Freedom table is meant to house information regarding the Tax Freedom (as a rank from 1-52) of each state, as well as other variables relevant to this metric. Further along, the "State" column, used as a primary key, will be replaced by State_Code, which will also be assigned as the key for this table.

```

88 • Create Table state_freedomB as
89     select a.OverallRank,
90     a.CorporateTaxRank,
91     a.IndividualIncomeTaxRank,
92     a.SalesTaxRank,
93     a.UnemploymentInsuranceTaxRank,
94     a.PropertyTaxRank,
95     b.State_Code
96     from state_freedom as a
97     join state_code__state_name as b
98     on a.State = b.State_Name;

```

- Here we create a duplicate of the State_Freedom table with the purpose of changing the state name in the original, for the state code instead.

```

100 • drop table state_freedom;
101 • alter table state_freedomB rename state_freedom;
102 • alter table state_freedom add primary key (State_Code);

```

- In these lines we can see that, after duplicating state_freedom, we drop the original, rename the copy, and add the state code as the primary key.

Counties

```

19     # TABLE 5 COUNTIES
20 • CREATE TABLE Counties(
21     FIPS_Code INTEGER PRIMARY KEY,
22     Unemployment_Rate_2021 float,
23     Unemployment_Absolute_2021 INTEGER,
24     Median_HH_Income_2021 float ) ;

```

- This specific County table houses the gathered data regarding unemployment at the county level, for the year 2021. These are represented as percentages and as an absolute value. Median Household Income at the county level is also included.

```

62 • UPDATE counties
63     SET FIPS_Code = 1000
64     WHERE FIPS_Code = 0;

```

- This snippet was meant to fix an error, where a state's FIPS code was changed to 0 when the data was loaded, so we had to manually add it in again.

```

80 • delete from counties where FIPS_Code % 1000 = 0;

```

- Here, we remove the states (%1000) from the counties table, so as to only have county data in them.

CITIES


```

26  # TABLE 6 CITIES
27  • CREATE TABLE cities (
28      City varchar(255),
29      County_FIPS INT,
30      Population INT,
31      Density float,
32      Military BOOLEAN,
33      Incorporated BOOLEAN,
34      Ranking INT,
35      PRIMARY KEY (City, County_FIPS)
36  );

```

- The Cities table is created to contain data regarding cities, such as their name, their county's FIPS code, population, density, whether there is a military base, whether it is incorporated into another city/county, and its freedom ranking.
- The term "Incorporated" is an administrative term used in the united states for areas that may be a bit removed from more metropolitan regions. Specifically, those cities that are not "incorporated" do not have the same rights in local government as the opposites do. Take for example, an area on the outskirts of Chicago, IL; if not incorporated, they must abide by city laws, but have no say in city politics.
- The primary codes used are both the city's name and the county's FIPS code.

Table 3: Creating Tables in DB

3.3. Verbal description of the most important data classes and attributes

The most important attributes in this table are the FIPS Codes, the Corporate Tax Ranking of the States and the Median Household Income.

- FIPS, or Federal Information Processing System Codes are unique identifiers for each county in the United States. This is a class integer.
- Corporate Tax Ranking is the ranking of the state when it comes to corporate taxes. This is a class integer.
- Median Household Income is the measure of a county's wealth. This is a class float.
- The above information can be combined using the DBMS system to answer the business question.

3.4. Normalization

Table Name (See 3.1)	Form	Justification for Normal Form		
		1N	2N (1N is a precondition)	3N (2N is a precondition)
county_fips_code,	3N	All attributes atomic, there are no sets, list or tuples.	For each key (FIPS_Code) there is one County_Name	No non-key attribute is transitively dependent on any key attribute.
counties	3N		For each key (FIPS_Code), there is exactly 1 number for each economic measure	
State_code__state_name	3N		For each key (State_Code), there is exactly one State_Name	
fips_code__state_code	3N		For each key (FIPS_Code), there is exactly one State Code	
cities	3N		For each concatenated key (city name, county_FIPS), there is exactly one value for each of the cities attributes	
state_freedom	3N		For each state_code, there is exactly one value for each of the indicators of economic freedom.	

Table 4: Table Showing Normal Forms of Tables in Database

Table 4 shows the normal forms of the tables. Such a form is necessary to ensure, among other things that no information is lost if information from the database is removed. For 3N, a **transitive dependency** would be the case for example if the name of the state was stored in the fips_code__state_code table, since each state name would be directly dependent on a state code, which would functionally depend on a county_FIPS_code.

4. Loading & Transforming the Data

For our case, we made the decision to load the data first, and transform it once it was in the database. During a previous module, we had practice with the ETL process, which is one of the reasons why we opted for implementing an ELT process in this case. In our minds, we had the chance to compare both processes, see their advantages and disadvantages, and learn from them. During this process we learned that an ELT process can be especially useful, since the server or VM offers better computational power than most personal machines, but it also comes with the risk of someone committing a mistake, which would permanently impact the entire database. While transforming before the loading process, though, each time we wanted to make an edit, at least as was in our case, all the code had to be re-ran, not just one line, which also makes ELT more comfortable.

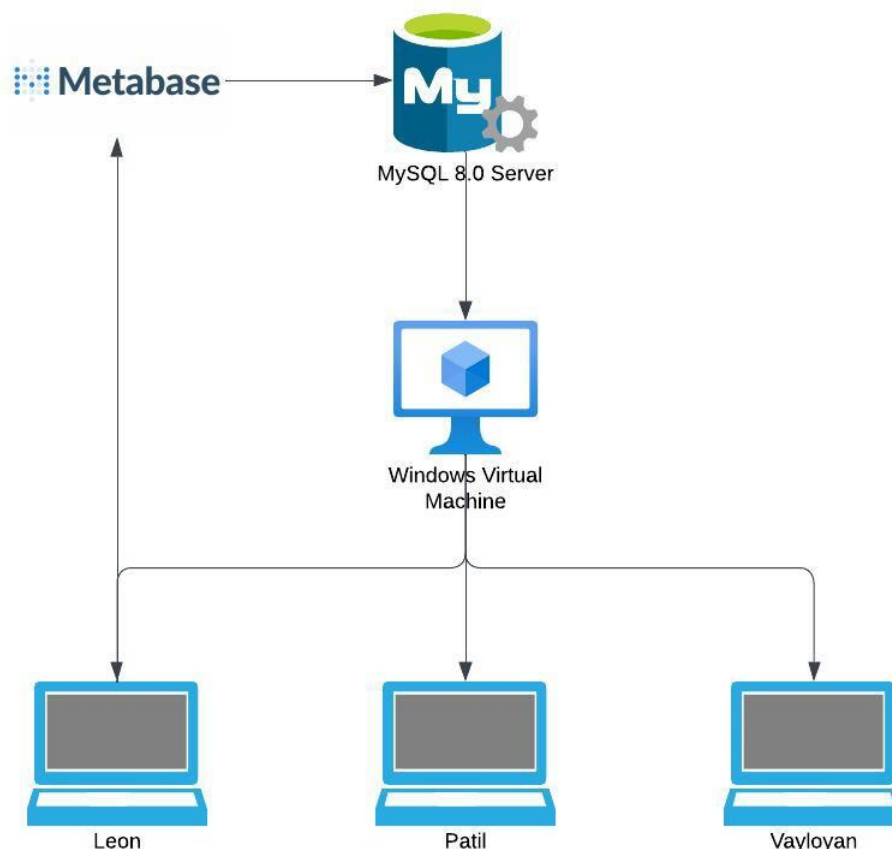


Figure 3: Database System Architecture

In Figure 3 we can see how our system is set up. Once we got the virtual machine running, we proceeded to load it with our data and organize our database, which we will go more in depth later on. Our main component is the **MySQL 8.0** server at the top of the diagram, which houses our database and allows us to access and manipulate it as we see fit for our purpose.

Below the database we can see the **Windows Virtual Machine** component. This is our primary access point to the Database, and it is where the server lives. All three of us students have equal access to the database, as can be appreciated on level lower in comparison to the virtual machine. We all work locally as well as on the server, so as to have a kind of 'version control', where we can all see what the latest changes have been and have our local environments up to date.

The final component would be **Metabase**, which is the software we decided to use for visualization in our project. It is directly connected to our **MySQL server**, but not directly to our database. As best practices instruct, we have made a separate schema within the server, which has only views of the main database, and that is what we connected Metabase to. This ensures that Metabase, and any other software or user connected to this second schema, will not have the privileges necessary to edit or manipulate our data, thus preventing any errors in the future.

4.1. Data loading

Table 5 below shows the steps in loading the data.

County_FIPS_Code
<pre> 12 • SET GLOBAL local_infile = true ; 13 • LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/DBM_Unemployment_csv.csv' 14 INTO TABLE County_FIPS_Code 15 FIELDS terminated BY ';;;' 16 LINES TERMINATED BY '\n' 17 IGNORE 0 LINES 18 (FIPS_Code, State_Code, County_Name, @dummy, @dummy, @dummy, @dummy); </pre> <ul style="list-style-type: none"> - During the uploading, due to our .csv files' structure, we had to specify the separator, or field terminator. - In this step, we populate the County_FIPS_Code table with our data, specifying which fields should be filled in.
State_Freedom
<pre> 40 • SET GLOBAL local_infile = true; # must be executed at the start of running each file 41 • LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/DBM_Econ_Freedom_csv.csv' 42 INTO TABLE state_freedom 43 FIELDS terminated BY ',' ESCAPED BY '\t' 44 IGNORE 1 LINES; </pre> <ul style="list-style-type: none"> - Next, we again use the LOAD command to populate the State_Freedom table. In this case, we did not have the issue where the separator was `;;;`, but we specified the field terminator as `,` anyways.
Counties
<pre> 46 • SET GLOBAL local_infile = true; 47 • LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/DBM_Unemployment_csv.csv' 48 INTO TABLE counties 49 FIELDS terminated BY ';;;' ESCAPED BY '\t' 50 IGNORE 0 LINES 51 (FIPS_Code, @dummy, @dummy, @Unemployment_Absolute_2021, @Unemployment_Rate_2021, 52 @Median_HH_Income_2021) 53 SET 54 Unemployment_Absolute_2021 = NULLIF(@Unemployment_Absolute_2021, ""), 55 Unemployment_Rate_2021 = NULLIF(@Unemployment_Rate_2021, ""), 56 Median_HH_Income_2021 = NULLIF(@Median_HH_Income_2021, ""); </pre> <ul style="list-style-type: none"> - For the Counties table, we load the data again with different columns from the same file as the County_FIPS_Table, specifying which columns should be taken. - In an effort to keep the database clean, we used NULLIF Statements to ensure that empty values will not be read in to the table, and will be replaced with NULL values in their stead.

Cities	
72 •	LOAD DATA LOCAL INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/cities_csv.csv'
73	INTO TABLE cities
74	FIELDS terminated BY ';;;' ESCAPED BY '\\t'
75	IGNORE 0 LINES
76	(City, County_FIPS, @dummy, Population, Density, Military, Incorporated, Ranking);
- For the Cities table, once again we had to specify the ';;;' field terminator. This is the final LOAD command in our scripts, and all the data is now ready for transformation and querying.	

Table 5: Loading Information into DBM

4.2. Transforming

Table 6 below shows the transformations done on the loaded data.

<pre>select count(*) from counties where Median_HH_Income_2021 = 0;</pre>	<table> <tr> <td></td><td>count(*)</td></tr> <tr> <td>▶</td><td>82</td></tr> </table>		count(*)	▶	82
	count(*)				
▶	82				
-To make further analysis better, we remove the rows with value 0 for the median household income.					
<pre>delete from counties where Median_HH_Income=0</pre>	<table> <tr> <td></td><td>count(*)</td></tr> <tr> <td>▶</td><td>0</td></tr> </table>		count(*)	▶	0
	count(*)				
▶	0				
- using this query we remove rows where the value was 0					
<p>Note: For analyzing different counties with another economic indicator (we focus on Median Household Income), this process would have to be repeated for the respective indicator. We choose not to remove those rows to keep as many rows as possible for the variable of interest in this project, median household income.</p>					

Table 6: Transformations

5. Analyzing & Evaluating Data

5.1. Queries in database syntax

To answer the business question, we need to be able to compare areas with different business friendliness indexes and their associated economic performance. The code snippets in below outline how this was done.

Table 7: Query

```
19 • create temporary table one_one as
20 select c.FIPS_Code, Unemployment_Rate_2021, Unemployment_Absolute_2021,
21 Median_HH_Income_2021, State_Code,
22 PERCENT_RANK() OVER (PARTITION BY fs.State_Code ORDER BY c.Median_HH_Income_2021 ASC)
23 AS income_rank
24 from counties as c
25 join fips_code__state_code as fs
26 on c.FIPS_Code = fs.FIPS_Code;
```

	FIPS_Code	Unemployment_Rate_2021	Unemployment_Absolute_2021	Median_HH_Income_2021	State_Code	income_rank
▶	2158	19.5	461	38203	AK	0
	2290	12.3	299	44934	AK	0.034482758620
	2164	8.7	57	51205	AK	0.068965517241
	2050	13.1	900	54849	AK	0.103448275862

- This code produces the table seen above. It uses a join to add the county's state code to the table. Using this state code then, the window function percent_rank is applied on the column Median_HH_Income to assign each county a score from 0-1. This score represent the percentile of the county's income relative to the other counties in the same state.

```
create temporary table lowest_10 as
select State_Code, avg(Median_HH_Income_2021) as Lowest_10
from one_one
where income_rank <= 0.1
group by State_Code;
```

	State_Code	Lowest_10
▶	AK	44780.666666666664
	AL	31692.571428571428
	AR	35177.375
	AZ	42666
	CA	49597

The next step is to create a temporary table which takes the mean of the household income for counties in each state which fall in the bottom 10% of counties in the state, as measured by household income. An aggregation (avg) is used, which requires a grouping, which is the state code here. An intermediate filter is introduced to apply this calculation only to counties which have an income rank below 0.1.

The previous logic is applied to the middle 50% of household income counties as well as the top 10%, producing the following two query results. The code is identical to the previous cell except for the where statement. See below for which where statements were used.

Median_50 = where income_rank >= 0.25 and income_rank <= 0.75

Top_10 = where income_rank >= .9

State_Code	Median_50
AK	70787.92857142857
AL	47071.666666666664
AR	45325.67567567567
AZ	54934.857142857145
CA	70699.85714285714

State_Code	Top_10
AK	91159.66666666667
AL	70187.71428571429
AR	64253.125
AZ	73949.5
CA	121687.66666666667

```

62 • create table freedom_distribution_income as
63     select
64         a.*,
65         b.Median_50,
66         c.Top_10,
67         d.CorporateTaxRank
68     from
69         lowest_10 as a
70     join
71         middle_50 as b on a.State_Code = b.State_Code
72     join
73         Top_10 as c on a.State_Code = c.State_Code
74     join state_freedom as d on a.State_Code = d.State_code;

```

State_Code	Lowest_10	Median_50	Top_10	CorporateTaxRank
AK	44780.666666666664	70787.92857142857	91159.66666666667	26
AL	31692.571428571428	47071.666666666664	70187.71428571429	22
AR	35177.375	45325.67567567567	64253.125	39
AZ	42666	54934.857142857145	73949.5	13
CA	49597	70699.85714285714	121687.66666666667	32

The next step is to combine the different county household income brackets (bottom 10%,middle 50%, top 10%) into a single table and add the information on economic freedom (here: corporate tax rate). This output is created in a permanent table since it will be used later for visualizations.

The table selects State Code and the values from lowest 10% from the temporary table lowest_10, this is combined with the values for middle 50% and top 10% and finally the Corporate Tax Rank. All of these tables have the State Code in common which makes it easy to use these as a key to combine them.

```

82 • create temporary table one_four as
83     select State_Code, Lowest_10, Median_50, Top_10,
84         CorporateTaxRank, NTILE(3) OVER (ORDER BY CorporateTaxRank) AS TaxRankGroup
85     from freedom_distribution_income;

```

State_Code	Lowest_10	Median_50	Top_10	CorporateTaxRank	TaxRankGroup
AK	44780.666666666664	70787.92857142857	91159.66666666667	26	2
AL	31692.571428571428	47071.666666666664	70187.71428571429	22	2
AR	35177.375	45325.67567567567	64253.125	39	3
AZ	42666	54934.857142857145	73949.5	13	1
CA	49597	70699.85714285714	121687.66666666667	32	2

This step uses another window function to assign a rank of 1,2,3 to each state based on their ranking in the Corporate Tax Rank. This is applied to the table created in the previous step. The intermediate result is saved as a temporary table.

94	•	<code>create table business_question_one as</code>	The left query creates a table
95		<code>select TaxRankGroup,</code>	"business_question_one". This table takes the
96		<code>round(avg(Lowest_10),0) as lowest_10,</code>	average of each of the household income
97		<code>round(avg(Median_50),0) as middle_50,</code>	brackets (lowest 10%, middle 50%, top 10%)
98		<code>round(avg(Top_10),0) as top_10</code>	by the Tax Rank. This is a much more reader
99		<code>from one_four</code>	friendly way to answer the business question.
100		<code>group by TaxRankGroup;</code>	The table can be seen in section 5.2 where
			the conclusion will be drawn.

5.2. Show connection between queries and use case

TaxRankGroup	lowest_10	middle_50	top_10
1	43340	56735	82861
2	47507	63244	89184
3	50275	63274	88399

Figure 4: Business Question 1

Figure 4 shows the key to the main business question. The data presents median household incomes for three distinct groups of counties categorized by their relative income: lowest_10%, middle_50%, and top_10%. Each group is further divided into three levels based on their corporate tax rates: Level 1 (most tax-free), Level 2 "medium tax free", and Level 3 (least tax-free). Analysis of this chart allows the business conclusion found in Section 8: Conclusions & Lessons Learned

Lowest 10% of Counties:

- Moving from Level 1 to Level 2 gives a change of \$4,167 (\$47,507 - \$43,340).
- Moving from Level 2 to Level 3 gives a change of \$2,768 (\$50,275 - \$47,507).

Middle 50% of Counties:

- Moving from Level 1 to Level 2 gives a change of \$6,509 (\$63,244 - \$56,735).
- Moving from Level 2 to Level 3 gives a change of \$30 (\$63,274 - \$63,244).

Top 10% of Counties:

Moving from Level 1 to Level 2 gives a change of \$6,323 (\$89,184 - \$82,861).
Moving from Level 2 to Level 3 gives a change of **-\$785** (\$88,399 - \$89,184).

Results

The results indicate that in all but one cases, moving to a higher corporate tax group would benefit the median household income of counties regardless of the "bracket" which the county is in. The sole exception is moving from "medium" amount of corporate taxes to the highest amount of corporate taxes shows a decrease of \$785 income for the wealthiest 10% of counties.

6. Query Optimization

To optimize SQL queries, we use various techniques such as analyzing the execution plan, query statistics, creating indexes, using materialized views, and ensuring queries are sargable.

6.1. Sargable Queries: Sargable queries are those that can take advantage of indexes, improving performance. Avoid using functions on columns in the WHERE clause as it makes them non-sargable.

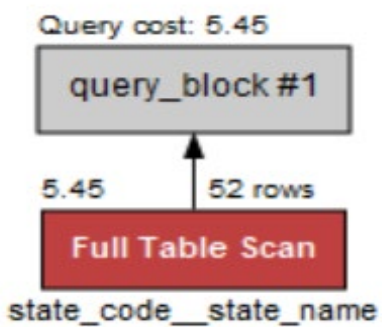
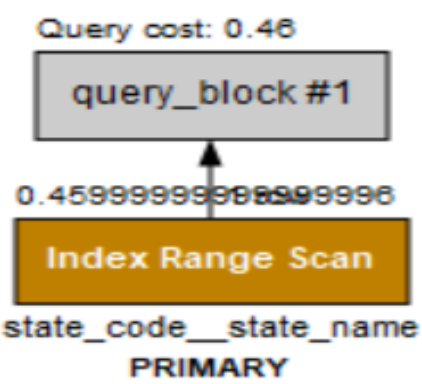
Here we compare two SQL queries to illustrate the difference between sargable and non-sargable queries. We will analyze the execution plan, performance, and effectiveness of each query using a Visual explain, a Tabular explain, and Query statistics.

NON Sargable Query

- `SELECT * FROM state_code_state_name WHERE LEFT(State_Code, 2) = 'AL';`
 - This query uses the LEFT() function on the State_Code_state_name column.
 - The database engine cannot directly use an index on State_Code_state_nam because it needs to apply the LEFT() function to each row before filtering.
 - This results in a full table scan, where every row in the table is checked as shown below in Fig.A.

Optimized Sargable Query

- `SELECT * FROM state_code_state_name WHERE State_Code LIKE 'AL%';`
 - The query uses the LIKE operator with a wildcard at the end ('AL%').
 - The database engine can directly use an index on State_Code to find rows that match the pattern.
 - This results in an Index Range scan, which is much faster than a full table scan as shown below in Fig.B.

1. Visual Explain	
Fig.A	Fig.B
	

B. Tabular explain

ular Explain

select_type	table	type	key...	rows	filtered	Extra
SIMPLE	state_co...	ALL		52	100.00	Using where

abular Explain

select_type	table	type	key...	rows	filtered	Extra
SIMPLE	state_co...	range	8	1	100.00	Using where

The above two tables show that with non sargable query it scans 52 rows whereas for sargable only 1 row is scanned.

C. Query Statistics

ry Statistics

Timing (as measured at client side):

Execution time: 0:00:0.00000000

Timing (as measured by the server):

Execution time: 0:00:0.00158540

Table lock wait time: 0:00:0.00000500

Errors:

Fatal Errors: NO

Warnings: 0

Rows Processed:

Rows affected: 0

Rows sent to client: 1000

Rows examined: 1004

Temporary Tables:

Temporary disk tables created: 0

Temporary tables created: 0

Joins per Type:

Full table scans (Select_scan): 1

Joins using table scans (Select_full_join): 0

Joins using range search (Select_full_range_join): 0

Joins with range checks (Select_range_check): 0

Joins using range (Select_range): 0

Sorting:

Sorted rows (Sort_rows): 0

Sort merge passes (Sort_merge_passes): 0

Sorts with ranges (Sort_range): 0

Sorts with table scans (Sort_scan): 0

Index Usage:

No Index used

Other Info:

Event Id: 172

Thread Id: 114

Query Statistics

Timing (as measured at client side):

Execution time: 0:00:0.00000000

Timing (as measured by the server):

Execution time: 0:00:0.00051950

Table lock wait time: 0:00:0.00000500

Joins per Type:

Full table scans (Select_scan): 0

Joins using table scans (Select_full_join): 0

Joins using range search (Select_full_range_join): 0

Joins with range checks (Select_range_check): 0

Joins using range (Select_range): 1

Errors:

Had Errors: NO

Warnings: 0

Rows Processed:

Rows affected: 0

Rows sent to client: 1

Rows examined: 1

Sorting:

Sorted rows (Sort_rows): 0

Sort merge passes (Sort_merge_passes): 0

Sorts with ranges (Sort_range): 0

Sorts with table scans (Sort_scan): 0

Index Usage:

At least one Index was used

Temporary Tables:

Temporary disk tables created: 0

Temporary tables created: 0

Other Info:

Event Id: 175

Thread Id: 114

The above 2 figures show that the Execution time has changed from 1.58 milliseconds to 0.51 milliseconds. The reduction in execution time is approximately **67.72%**. In this way we have successfully the Runtime of the queries.

6.2. Execution Plan: The execution plan helps to understand how the SQL engine executes a query, showing the steps taken to retrieve the data. By analyzing the execution plan, we can identify inefficient operations and optimize them.

To view the execution plan in MySQL, we use the `EXPLAIN` statement before the query.

```
1 • EXPLAIN SELECT * FROM counties WHERE Median_HH_Income_2021 = 50000;
```

6.3. Indexes: Indexes speed up the retrieval of rows by using pointers. They are particularly effective for columns used in WHERE, JOIN, ORDER BY, and GROUP BY clauses.

We created indexes on frequently queried columns like FIPS_Code and State_Code as shown in query below.

```
1 • CREATE INDEX idx_median_income ON counties(Median_HH_Income_2021);
2 • CREATE INDEX idx_fips_code ON counties(FIPS_Code);
3 • CREATE INDEX idx_state_code ON state_freedom(State_Code);
```

7. Visualization & Decision Support

For our visualization of our data and results, we decided to use Metabase as the primary tool. Connecting it to the database was not a difficult task, and as soon as we had the connection, it had some visual aids ready for us. We quickly learned that, though it lacks some features, it is still a very powerful tool.

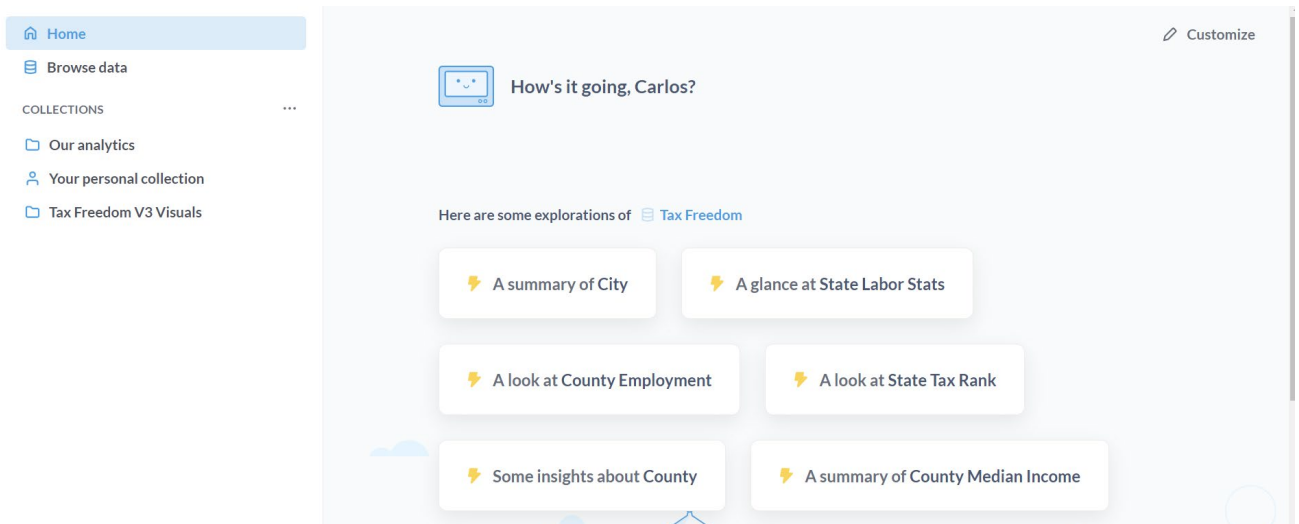


Figure 5: Metabase Interface

As we can see here, on the landing page it already has some suggestions for us, as well as the ability to create a dashboard, which we didn't use, and collections to keep our preferred visuals. Becoming more accustomed to it would take some more time, but in our opinion, would definitely be worth the effort.

Below, we will go through our visuals and lightly break them down to make it more digestible, so as to expand on our research and questions. These visual aids were all generated exclusively with the help of Metabase and no other visualization tools.



Figure 6: Scatterplot showing relationship between Corporate Tax Rankings of counties and their median household incomes, split into 3 categories based on the relative income of the county to the state they are in.

Figure 6 shows that the pattern discovered in the query is not clear in a scatterplot. Particularly for the yellow (lower income counties), there seems to be little changes as tax ranks increase (more taxes). However, for the purple (richest counties), towards the end of the diagram there does seem to be a drop off in income as the wealth increases. Unfortunately, Metabase does not allow line of best fits, even just horizontal ones, which is an often requested feature. This feature would have helped in discerning a clearer pattern. The **business conclusion** for this graphic is that the spread of incomes at different tax levels is fairly high, meaning that taxation ranking is likely not the primary determinant of county median household income. As such local governments should be open to exploring alternative means of increasing wealth.

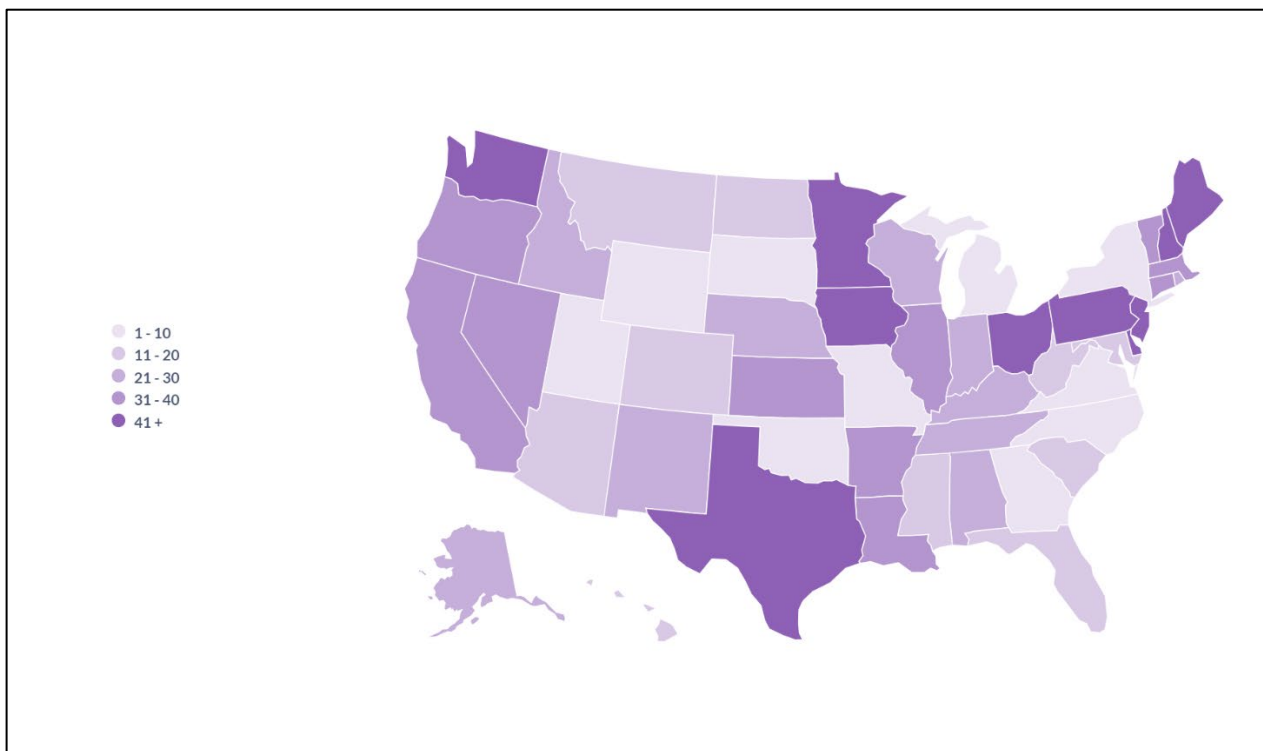


Figure 7: Average County-Level Corporate Tax Rank, grouped by State

In Figure 7 we can see how average county-level corporate tax rank behaves by state, with those values closer to 1 having the more preferable taxes for corporations. It's interesting to see recognizable states such as Texas and Washington being on the lower end of the tax rankings, and California and New York higher up. The **business value** is that states deciding to alter their tax code can see, at a glance how their state is on corporate taxation relative to other states. This will help them in assessing the political feasibility of increasing corporate taxes. If a state is already relatively high, the voters may not accept higher taxes, and companies may be increasingly likely to "flee" the high tax area.

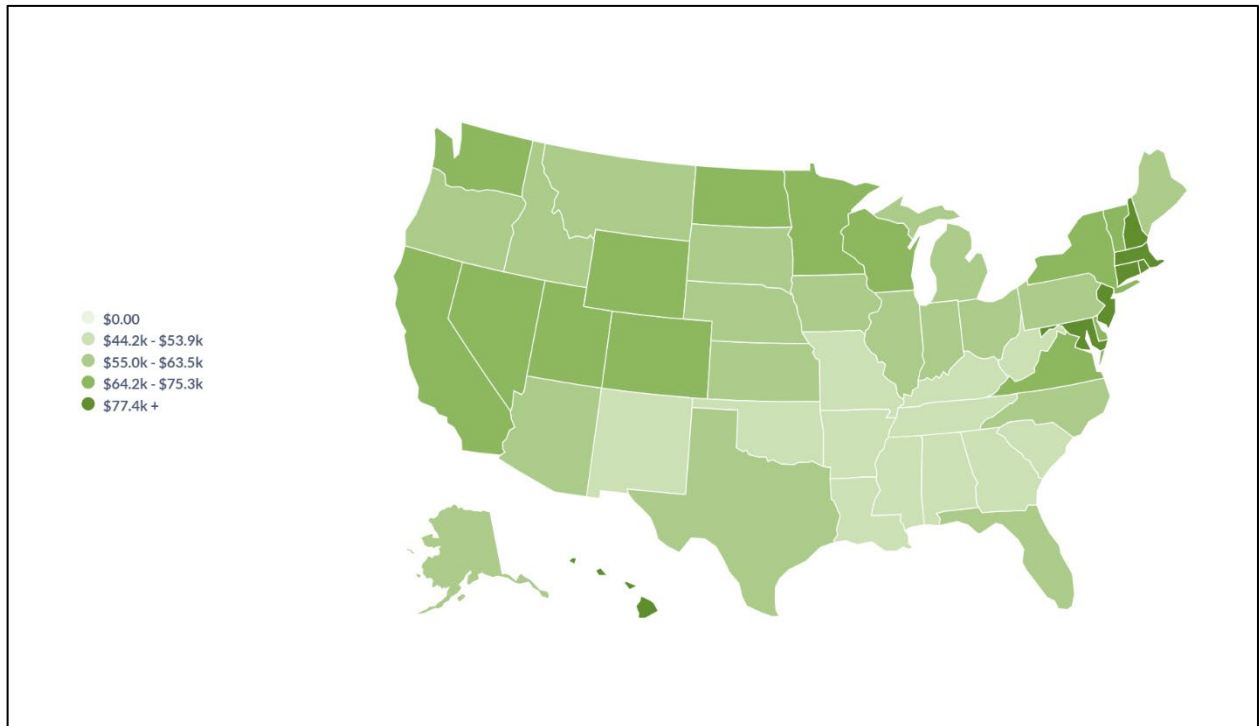


Figure 8: Average of County-Level Median Household Income, grouped by State

In Figure 8, we observe the average of median household income by county, by state. We can see that, in some cases, higher income states align with higher tax-ranked states. However, in general, we are able to appreciate that higher corporate taxes tend to lead to higher income for the population at a state level. The **business value** of this figure is that states can see, at a glance where they stand relative to other states in the average county Median Household Income criteria. This may help them in assessing the need for updating the tax code to increase household income, if a state is already wealthy, the voters may have other priorities than economics. Using the levers of corporate taxation to push economic growth (and thus income) oftentimes comes at the cost of other factors such as: environmental issues from increased economic activity. States can ask themselves, looking at those around us, do we really need this?

8. Conclusions & Lessons Learned

We have performed a descriptive analysis of different average household income levels at different levels of corporate taxations for counties in different wealth groups relative to their states.

8.1. Business Conclusion:

The results indicate that the money collected by corporate taxes can be efficiently used by the government of most counties to boost household income, with the exception of wealthy counties at already moderate levels of taxation. It is likely that in the top 10% wealthiest counties of each state, a lot of wealthy people live that more directly benefit from anything that enhances corporate profits, which is why in these areas we see lower household incomes with higher taxation levels. These results are only an indication however, and they do not test for causality.

Persona State Government: Our original client, the state government, needs to ask itself who they want to please economically? Given that most counties will not be in the wealthiest 10%, it is likely that overall increases corporate taxes will improve the household income for most counties, depending on the

county's wealth quite significantly even, with the largest change being an increase in median household income over \$6k. However, should the state for whatever reason decide to prioritize wealthier areas, then if the area is already a "moderate" corporate tax amount, they should not move the highest tax amount. And if the state is in the highest tax amount, then they should consider reducing the corporate tax burden.

Given the US is a democratic country it is likely that states should favour a higher tax burden for corporates this improves the economic situation of households according to these results. Of course, we do not expect states to change their tax code solely based on the result: voter sentiment and turnout rates across counties will play a significant role in the states's calculus in making this decision.

Our client decides to forward the conclusion to another **persona, a county level government**, which also have some leeway in setting taxes. They instruct counties to consider where they are on the economic ladder and make decisions according to the results. The guiding principle is in general taxes should be increased on corporate entities except if a county is wealthy and already has significant taxes on corporates, however in this scenario the benefits of a corporate tax decrease will likely be small.

8.2. Lessons Learned

This project was not well suited to using a DBMS, it is likely that an Excel or Statistical Programming language would have allowed us to obtain better results faster, using more advanced statistical tools than creating a table. We nevertheless learned about how to use SQL to extract information from various tables stored in a database, which is didactically valuable given their frequent use in data driven business.

Collaboration was the main challenge and the main lesson learned is to ensure proper communication. Possibly due to unfamiliarity with SQL or the SQL program itself, we found it very difficult to create a Minimally Viable Product to then iterate over and improve. The necessity to define things permanently early on made this process, which is usually followed in data science projects difficult. This reality would have necessitated an additional effort on communication.

9. Appendix

9.1. AI Disclosure

Table 8 describes how students used AI in this project.

Student	Use of AI
Vayloyan	<ul style="list-style-type: none">- SQL query: translating my request into a code. I did always make 2-3 “good faith” attempts to write the code from memory, by using non AI internet resources or course content. I found it useful not to describe what the query should be but how the result should look like from a starting table.- Spelling and reviewing.- For the tedious interpretation of 5.2, wrote one interpretation and asked engine to write the interpretation of all.- Questions around Chen’s modelling.
Leon	<ul style="list-style-type: none">- SQL: openai’s ChatGPT was tremendous help for creating tables when it came to visualization. Some tables were a bit too complex for the Metabase tool to accurately capture what we wanted to see, but thanks to this tool, breaking down tables into components and views was made easier.- General advice to create a better project.- Aid in making a more efficient ETL process.
Patil	<ul style="list-style-type: none">-Understand Potential Insights: Gain insights into the datasets to identify key trends and relationships.-Identify Efficient Methods: Learned about efficient data loading techniques, such as using LOAD DATA INFILE for bulk imports.-Troubleshoot Loading Errors: Received guidance on troubleshooting common data loading errors, such as format issues or constraint violations.-Obtained ideas for complex transformations, including joining tables and creating temporary tables for intermediate steps.-Debug Errors: Get different ways to debug errors encountered during executionTroubleshoot Execution Plan Issues: Find solutions for troubleshooting the unavailability of execution plans on the VM.Optimize SQL Queries: making queries sargable and using indexes.

Table 8: AI Disclosure

9.2. Login Information

Please use following details to log into the Virtual Machine to check the Database:

IP: 86.119.42.55

Username:

carlosleon

Password:

13%Tm2o8CZbd5rtBs\$qI

Metabase:

Because we had issues with the VM, we had to run Metabase locally, which is why it would not be possible for us to grant access.

9.3. Self-Assessment

Evaluation of your database project work in the DBM module

Title:	United States County Level Income Response to Corporate Taxation					
Team:	Federal SQL Consultants					
Students:	Name		Score	Name		Score
	Alec Vayloyan			Carlos Leon		
	Aishwarya Patil					
Professor:	Luis Terán					

	Maximum Possible points	Points achieved	Self-assessment	Initials of the responsible students
1 Project idea and use case Motivation, Decision Support, Data, DB Tech.	5		4.5	AV, AP, CL
2 Model and scheme ER, DDL, NF	5		5	AV, AP, CL
3 Loading and transforming Architecture, Load, Transformation	5		5	AV, AP, CL
4 Queries and analyses Query, Aggregation, Grouping, Join, Selection, Proj	5		5	AV, AP, CL
5 Performance optimization Execution Plan, EX, MV, SARG	5		5	AP, CL, AV
6 Visualization and interpretation BI Tool, Graph, Action, Personas	5		4.5	CL, AP, AV
Total	30			

We hereby certify that our team has independently developed this project and that each team member will receive points on exactly those criteria to which they have contributed.

**please consider as signed by all group members on 7.6.2024, Switzerland. Former team name "ACA"*

Why not full points?

Project use case: Loose half a point because these business questions could more easily be answered in a Python or even Excel, although if the plan is too add more information later then the DB System makes sense.

Visualizations: Loose a half point because the visualizations are quite basic and the insight, especially from the maps could be relatively easily read off from a ranking.