CS411 Project Report

By: Nicholas Chan, John Driscoll, and Tony Liu

1. Changes in the direction of the project

a. Many aspects of the website were expanded from using just a zip code to using a zip code, FIPS, and county. We found that it was almost impossible to visually identify a zip code on a map of the United States, which reduced the usefulness of the interactive map. However, dividing the map into counties provides a much better user experience due to their larger geographic area. Similarly, sorting by the top 5 zip codes in the country/state often returns only 5 zip codes in the same county because much of the data is recorded by county. Making the switch from solely zip to zip, county, and FIPS allows the user to receive more useful information and interact better with the website.

b. We've removed the Top five worst ZIP codes feature from our original proposal. This did not provide useful or meaningful information for someone considering moving, as they would likely never consider a zip code with such low rankings. We decided to focus on positive rankings and relevant data to help users make informed decisions.

c. The ability to search by zip code attributes was removed from our original proposal. Due to the large scope of our database, many of the zip codes had datasets that were too incomplete to be utilized properly. This severely hindered the functionality of our process, as many zip codes could never be identified through it, so we decided to remove the process from our website.

2. What our application achieved/failed to achieve usefulness

a. Our application was able to return data from many different databases regarding GDP, education, population, housing prices, and other socioeconomic factors for each ZIP Code/FIPS region.

b. Both guest users and signed-in users can access data. However, signing into an account will allow for additional functions, such as the top 5 counties per state for a specific attribute the user chooses, and allow users to favorite the ZIP codes they want to look at later.

c. If users forget their password, they are given the option to create a new password as long as they remember their username.

d. Some ZIP codes did not have complete datasets because we took data from many different databases, which were not created by the same people. This caused some ZIP codes not to have complete datasets for every attribute.

3. Schema changes

a. More relevant data files were added to our database, as well as a FIPS_to_Counties table. While working with our data files, we discovered that some data files were incomplete, stopping before all the states were covered. One of the most important occurrences of this was that our zip_to_counties table stopped at Montana instead of going through all states. This originally posed a large problem until we were able to find a FIPS_to_County data table. This allowed the connection of zip to counties, along with all important county-exclusive data, through our preexisting FIPS_to_ZIP table.

b. Many files were updated to include a state column whenever a county column was present. Data files came in 3 main forms: 1)just counties, 2)counties and state names in one column, and 3)state and county names in separate columns. This difference in structure caused problems not only in linking files together(as there were problems with primary and secondary keys) but also in attempting to code the project. The best option we found was to convert all files, when applicable, to a structure containing both county and state names. This normalization helped prevent errors where 2 states shared a county with the same name, as well as improving the general usability of our database.

   c. Many of the data files needed to be changed due to poor original formatting. Numerous files included a strange formatting convention where, instead of using columns, rows were used with suffixes. This organization convention made accessing data from some data files almost impossible, as variable names were extremely lengthy and rows had to be skipped to collect data of the same type. Reorganizing these files allowed the data within to be properly used.

4. ER diagram changes
   a. Our ER Diagram is very different from our actual databases in the application.

   b. Data is no longer one entity; multiple different attributes are its own entity now. For Example:
      i. Housing Affordability
      ii. Job Opportunities
      iii. Population Growth
      iv. GDP Growth
      v. Education Metric
      These entities have their own primary keys and attributes.

   c. We removed the Region database from our application. Instead, each of our data entities has a ZIP/FIPS code as its primary key.

5. What functionalities were added/removed
   a. We decided to add user groups. This is so that families can create groups and access their favorite zip codes together to see their options. A transaction was used so that when the last member of a group leaves, the group is automatically deleted from the user_groups table. This is because we do not want any hanging groups or data when the last member leaves.

   b. We attempted to add a function where users could input their preferred housing types, and it would return a group of counties that had more of what they wanted. This was scrapped as there were too many errors and not enough time to resolve them.

6. Explain how you think your advanced database programs complement your application.
    a. Transaction: The transaction ensures that user_groups stays consistent with no lingering data. When the last member of a group leaves, the group is deleted, and all of its favorites are also deleted. We made this a transaction so that in the case of a crash or network error, it would either go through fully or not at all. This ensures our user_groups database has no lingering groups or favorites.

    b. Constraint: Numerous implementations throughout the project base. For example, to conserve storage capacity, we limited the number of favorites per user to 10 by imposing constraints.

    c. Trigger: Login attempts restrictions to ensure security. Three consecutive failures would lead to a forced password reset. We have a trigger that checks when we update the user database. This way, when we set up a new password. The counter for consecutive failures resets to 0, so users who reset their password do not constantly get a warning asking them to reset the password.

    d. Stored Procedures: Our top 5 countries application uses a stored procedure that performs all the necessary tasks, depending on the selected drop-down menu option. This way, we can just call this procedure to get all of the information instead of doing all of the work every time.

7. Technical Challenges Encountered
    a. Nicholas Chan: I used Google Cloud Platform to host my SQL database; however, I ran out of free credits a few days before the final demo and was not able to access my databases. This set our team back by days of work. A bit of helpful advice for future teams: back up your work and make sure to shut it off so you don’t waste the free credits. Another way of getting around it is to have a different Google account take control of that project and then remove the old Google account along with its billing account. This method allows you to continue working on the project until you reach the final demo.

b. John Driscoll: I attempted to code the search for the zipcode by user preference, but it would not function properly. The idea was to accept the user's preference based on what button they clicked, which would fill a variable user_preference VARCHAR(255) with the column name associated with their selection. The goal was to then return all zipcodes where the user selection held the highest value among all other similar columns, for example, if the user selected 2-bedroom houses, the program would return the zipcodes with the most 2-bedroom houses compared to all other housing types. This process would then repeat for all other selections the user made. The goal was to then count the number of times each zipcode occurred, which was its score, and return only the zipcodes with the highest score. Many problems arose with this implementation,

c. Tony Liu: The development of the map is very troublesome because of the limited storage of the virtual machine. We initially tried using Leaflet with JSON data of all the county boundaries to map them out on the front end, but failed. Then I worked with Nick to resolve it through other methods of indirectly fetching data without occupying storage.

8. Feedback from Stage 2

a. Feedback: Appropriate connectors are not used in the diagram.
    i. Before the change, we only partially specified the ER relationships with numbers, yet without explaining our connector arrows.
    ii. Changes: The diagram was updated to include appropriate connectors and connection labels in our new design.

b. Feedback: SearchResult and SearchPreferences should probably not be entities
    i. Changes: SearchResult and SearchPreferences were removed as tables.

c. Feedback: You haven’t shown primary keys in some diagrams.

    i. Changes: Primary keys are now properly constructed within all the relations, as showcased in stage 3.

d. Feedback: Email should determine the username
    i. Changes: Email and Username are the same attribute now. As there was no need for a username since email would also serve as the primary key. This is showcased during our stage 4 demo.

e. Feedback: The Normalization process is not shown
    i. Changes: Removed all shared columns primary keys to get rid of all

```
CREATE TABLE householdinfo_by_county(
        countyandstateName VARCHAR(255)
        totalHouseholds INT,
        households_childrenunder18 INT,
        households_childrenunder17 INT,
        households_children6to17 INT,
        households_adultsover60 INT,
        households_adultsover65 INT
        PRIMARY KEY(countyandstateName)
);
CREATE TABLE householdinfo_by_county(
        countyName VARCHAR(255),
        stateName VARCHAR(255),
        totalHouseholds INT,
        households_childrenunder18 INT,
        households_childrenunder17 INT,
        households_children6to17 INT,
        households_adultsover60 INT,
        households_adultsover65 INT,
        PRIMARY KEY(countyName, stateName)
);
```

Table name	Primary key	Dependency	Form
fips_to_counties	FIPS_Code	FIPS_Code->StateName, CountyName	BCNF

labor_force_data	StateName, CountyName	StateName, CountyName-> unemp2020,unemp2021,unemp2022	BCNF
population_estimate	StateName, CountyName	StateName, CountyName-> p1_5,p6_10,p11_18,p19_25,etc	BCNF
gdp_by_county	StateName, CountyName	StateName, CountyName-> gdp2020,gdp2021,gdp2022	BCNF
educationtest	StateName, CountyName	StateName, CountyName-> CollegeGrads	BCNF
fips_to_zipcode	zipCode	zipCode->FIPS_CODE,StateName,CountyName	BCNF
user_database	Username	Username->Passwords, failed_attempts	BCNF
Favorites	Username	Username->favorite1, .., favorite10	BCNF
group_memberships	membership_id	Membership_id -> Username, Group_id	BCNF

- Fips_to_counties: converts fips to county and state name. BCNF sets unique primary key FIPS_Code, all other values dependent on it
    - Fips:unique county identifier
    - CountyName: specific name of county

    - StateName:specific name of state
- Labor_force_data: returns the rate of employment for multiple years in each county . BCNF uses unique primary key pair countyName/stateName to determine rows, all other values dependent on it
    - CountyName/StateName: unique identification pair used to identify a specific county in a certain state
    - Unemp2020:unemployment rate for 2020 in specific county
    - Unemp2021:unemployment rate for 2021 in specific county
    - unemp2022:unemployment rate for 2022 in specific county
- population_estimate: returns the number of people in each county . BCNF uses unique primary key pair countyName/stateName to determine rows, all other values dependent on it
    - CountyName/StateName: unique identification pair used to identify a specific county in a certain state
    - P1_5:number of people ages 1-5 in specific county
    - P6_10::number of people ages 6-10 in specific county
    - P11_18:number of people ages 11-18 in specific county
    - P19_25:number of people ages 19-25 in specific county
    - Etc:rest of data up to 75 years old in county
- gdp_by_county: returns the rate of gdp change for multiple years in each county . BCNF uses unique primary key pair countyName/stateName to determine rows, all other values dependent on it
    - CountyName/StateName: unique identification pair used to identify a specific county in a certain state
    - gdp2020:gdp rate for 2020 in specific county
    - gdp2021:gdp rate for 2021 in specific county
    - gdp2022:gdp rate for 2022 in specific county
- fips_to_zipcode: converts zipcode to fips code, county and state name. BCNF sets unique primary key zipCode, all other values depend on it
    - Zipcode:unique county identifier
    - Fips:code associated with specific county
    - CountyName: specific name of county
    - StateName:specific name of state
- user_database: converts Username to password and the number of failed attempts . BCNF sets unique primary key Username, all other values depend on it
    - username:unique user identifier
    - Password: specific password for user
    - failed_attempts: number of attempts failed at signing in
- favorites: converts Username to variables storing favorites 1 through 10 . BCNF sets unique primary key Username, all other values depend on it
    - username:unique user identifier
    - favorite1: specific favorite 1 value for user
    - favorite2: specific favorite 2 value for user
    - favorite3: specific favorite 3 value for user

- favorite4: specific favorite 4 value for user
    - favorite5: specific favorite 5 value for user
    - favorite6: specific favorite 6 value for user
    - favorite7: specific favorite 7 value for user
    - favorite8: specific favorite 8 value for user
    - favorite9: specific favorite 9 value for user
    - favorite10: specific favorite 10 value for user
- group_memberships: converts membership_id to username and group name. BCNF sets unique primary key membership_id, all other values depend on it
    - membership_id:unique user identifier
    - username:specific user identifier
    - Group_id: specific group identifier

f. Feedback: Relationships and their cardinalities are not clearly specified. They have been mentioned in the entity description, but not exhaustively, and also have issues with cardinality (e.g., if something is 0 to 1 or exactly one, 0 to many or 1 to many etc.)

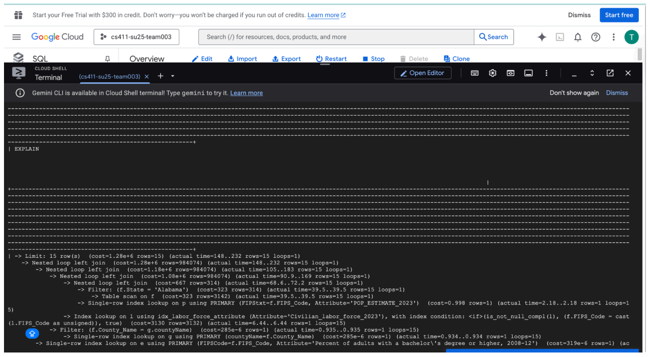
   i. Changes:

9. Feedback from Stage 3

a. Feedback: Screenshots of explain, analyze commands for each indexing strategy are missing.

   i. Changes:

      1. Query One



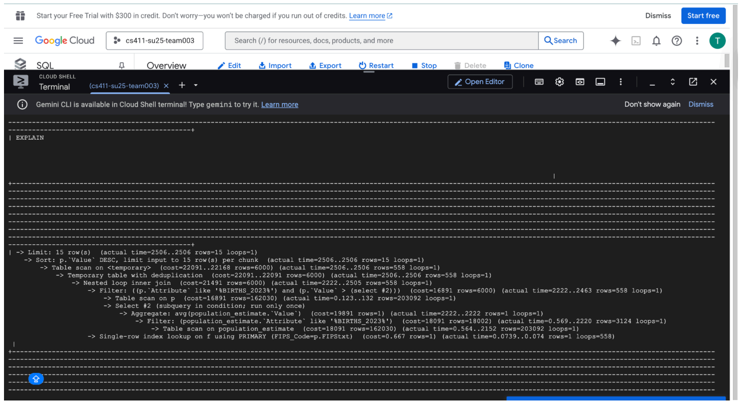
Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more

Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more

```
| EXPLAIN

| -> Limit: 15 row(s)  (cost=1.28e+6 rows=15) (actual time=148..232 rows=15 loops=1)
    -> Nested loop left join  (cost=1.28e+6 rows=984074) (actual time=148..232 rows=15 loops=1)
        -> Nested loop left join  (cost=1.18e+6 rows=984074) (actual time=105..183 rows=15 loops=1)
            -> Nested loop left join  (cost=1.08e+6 rows=984074) (actual time=90.9..169 rows=15 loops=1)
                -> Nested loop left join  (cost=667 rows=314) (actual time=68.6..72.2 rows=15 loops=1)
                    -> Filter: (f.State = 'Alabama')  (cost=323 rows=314) (actual time=39.5..39.5 rows=15 loops=1)
                        -> Table scan on f  (cost=323 rows=3142) (actual time=39.5..39.5 rows=15 loops=1)
                    -> Single-row index lookup on p using PRIMARY (FIPStxt=f.FIPS_Code, Attribute='POP_ESTIMATE_2023')  (cost=0.998 rows=1) (actual time=2.18..2.18 rows=1 loops=1
5)
                -> Index lookup on l using idx_labor_force_attribute (Attribute='Civilian_labor_force_2023'), with index condition: <if>(is_not_null_compl(l), (f.FIPS_Code = cast
(l.FIPS_Code as unsigned)), true)  (cost=3130 rows=3132) (actual time=6.44..6.44 rows=1 loops=15)
            -> Filter: (f.County_Name = g.countyName)  (cost=285e-6 rows=1) (actual time=0.935..0.935 rows=1 loops=15)
                -> Single-row index lookup on g using PRIMARY (countyName=f.County_Name)  (cost=285e-6 rows=1) (actual time=0.934..0.934 rows=1 loops=15)
        -> Single-row index lookup on e using PRIMARY (FIPSCode=f.FIPS_Code, Attribute='Percent of adults with a bachelor\'s degree or higher, 2008-12')  (cost=319e-6 rows=1) (ac
```

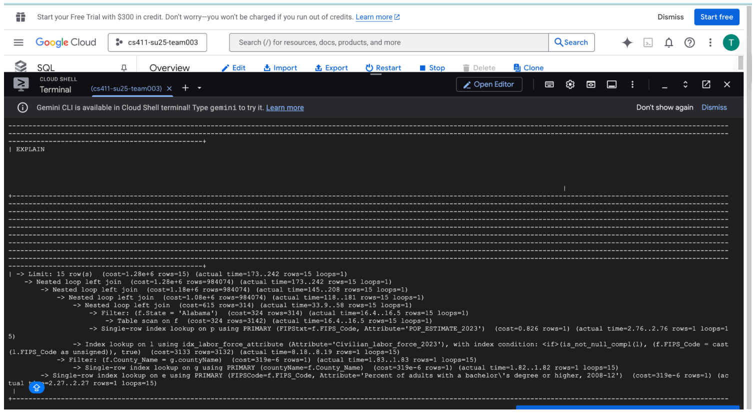
2. Query Two


Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
Dismiss
Start free
Google Cloud
cs411-su25-team003
Search (/) for resources, docs, products, and more
Search
SQL
Overview
Edit
Import
Export
Restart
Stop
Delete
Clone
CLOUD SHELL
Terminal
(cs411-su25-team003)
Open Editor
Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more
Don't show again
Dismiss

```
| EXPLAIN

+-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
-----------------------------------------+
| -> Limit: 15 row(s)  (actual time=2506..2506 rows=15 loops=1)
    -> Sort: p.`Value` DESC, limit input to 15 row(s) per chunk  (actual time=2506..2506 rows=15 loops=1)
        -> Table scan on <temporary>  (cost=22091..22168 rows=6000) (actual time=2506..2506 rows=558 loops=1)
            -> Temporary table with deduplication  (cost=22091..22091 rows=6000) (actual time=2506..2506 rows=558 loops=1)
                -> Nested loop inner join  (cost=21491 rows=6000) (actual time=2222..2505 rows=558 loops=1)
                    -> Filter: ((p.`Attribute` like '%BIRTHS_2023%') and (p.`Value` > (select #2)))  (cost=16891 rows=6000) (actual time=2222..2463 rows=558 loops=1)
                        -> Table scan on p  (cost=16891 rows=162030) (actual time=0.123..132 rows=203092 loops=1)
                        -> Select #2 (subquery in condition; run only once)
                            -> Aggregate: avg(population_estimate.`Value`)  (cost=19891 rows=1) (actual time=2222..2222 rows=1 loops=1)
                                -> Filter: (population_estimate.`Attribute` like '%BIRTHS_2023%')  (cost=18091 rows=18002) (actual time=0.569..2220 rows=3124 loops=1)
                                    -> Table scan on population_estimate  (cost=18091 rows=162030) (actual time=0.564..2152 rows=203092 loops=1)
                    -> Single-row index lookup on f using PRIMARY (FIPS_Code=p.FIPStxt)  (cost=0.667 rows=1) (actual time=0.0739..0.074 rows=1 loops=558)
 |
+-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
```

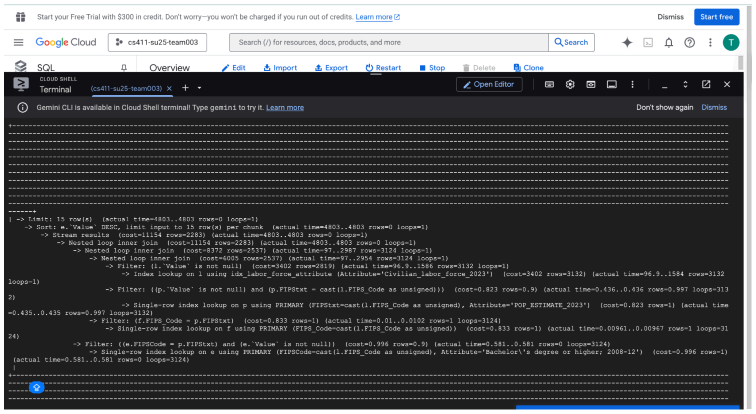
3. Query Three


Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
Dismiss
Start free
Google Cloud
cs411-su25-team003
Search (/) for resources, docs, products, and more
Search
SQL
Overview
Edit
Import
Export
Restart
Stop
Delete
Clone
CLOUD SHELL
Terminal
(cs411-su25-team003)
Open Editor
Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more
Don't show again
Dismiss

```
| EXPLAIN

+-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
-----------------------------------------+
| -> Limit: 15 row(s)  (cost=1.28e+6 rows=15) (actual time=173..242 rows=15 loops=1)
    -> Nested loop left join  (cost=1.28e+6 rows=984074) (actual time=173..242 rows=15 loops=1)
        -> Nested loop left join  (cost=1.18e+6 rows=984074) (actual time=145..208 rows=15 loops=1)
            -> Nested loop left join  (cost=1.08e+6 rows=984074) (actual time=118..181 rows=15 loops=1)
                -> Nested loop left join  (cost=615 rows=314) (actual time=33.9..58 rows=15 loops=1)
                    -> Filter: (f.State = 'Alabama')  (cost=324 rows=314) (actual time=16.4..16.5 rows=15 loops=1)
                        -> Table scan on f  (cost=324 rows=3142) (actual time=16.4..16.5 rows=15 loops=1)
                    -> Single-row index lookup on p using PRIMARY (FIPStxt=f.FIPS_Code, Attribute='POP_ESTIMATE_2023')  (cost=0.826 rows=1) (actual time=2.76..2.76 rows=1 loops=1
5)
                -> Index lookup on l using idx_labor_force_attribute (Attribute='Civilian_labor_force_2023'), with index condition: <if>(is_not_null_compl(l), (f.FIPS_Code = cast
(l.FIPS_Code as unsigned)), true)  (cost=3133 rows=3132) (actual time=8.18..8.19 rows=1 loops=15)
            -> Filter: (f.County_Name = g.countyName)  (cost=319e-6 rows=1) (actual time=1.83..1.83 rows=1 loops=15)
                -> Single-row index lookup on g using PRIMARY (countyName=f.County_Name)  (cost=319e-6 rows=1) (actual time=1.82..1.82 rows=1 loops=15)
        -> Single-row index lookup on e using PRIMARY (FIPSCode=f.FIPS_Code, Attribute='Percent of adults with a bachelor\'s degree or higher, 2008-12')  (cost=319e-6 rows=1) (ac
tual time=2.27..2.27 rows=1 loops=15)
 |
+-----------------------------------------------------------------------------------------------------
```

4. Query Four


Start your Free Trial with $300 in credit. Don't worry—you won't be charged if you run out of credits. Learn more
Dismiss
Start free
Google Cloud
cs411-su25-team003
Search (/) for resources, docs, products, and more
Search
SQL
Overview
Edit
Import
Export
Restart
Stop
Delete
Clone
CLOUD SHELL
Terminal
(cs411-su25-team003)
Open Editor
Gemini CLI is available in Cloud Shell terminal! Type gemini to try it. Learn more
Don't show again
Dismiss

```
+-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
------+
| -> Limit: 15 row(s)  (actual time=4803..4803 rows=0 loops=1)
    -> Sort: e.`Value` DESC, limit input to 15 row(s) per chunk  (actual time=4803..4803 rows=0 loops=1)
        -> Stream results  (cost=11154 rows=2283) (actual time=4803..4803 rows=0 loops=1)
            -> Nested loop inner join  (cost=11154 rows=2283) (actual time=4803..4803 rows=0 loops=1)
                -> Nested loop inner join  (cost=8372 rows=2537) (actual time=97..2987 rows=3124 loops=1)
                    -> Nested loop inner join  (cost=6005 rows=2537) (actual time=97..2954 rows=3124 loops=1)
                        -> Filter: (l.`Value` is not null)  (cost=3402 rows=2819) (actual time=96.9..1586 rows=3132 loops=1)
                            -> Index lookup on l using idx_labor_force_attribute (Attribute='Civilian_labor_force_2023')  (cost=3402 rows=3132) (actual time=96.9..1584 rows=3132
loops=1)
                        -> Filter: ((p.`Value` is not null) and (p.FIPStxt = cast(l.FIPS_Code as unsigned)))  (cost=0.823 rows=0.9) (actual time=0.436..0.436 rows=0.997 loops=313
2)
                            -> Single-row index lookup on p using PRIMARY (FIPStxt=cast(l.FIPS_Code as unsigned), Attribute='POP_ESTIMATE_2023')  (cost=0.823 rows=1) (actual time
=0.435..0.435 rows=0.997 loops=3132)
                    -> Filter: (f.FIPS_Code = p.FIPStxt)  (cost=0.833 rows=1) (actual time=0.01..0.0102 rows=1 loops=3124)
                        -> Single-row index lookup on f using PRIMARY (FIPS_Code=cast(l.FIPS_Code as unsigned))  (cost=0.833 rows=1) (actual time=0.00961..0.00967 rows=1 loops=31
24)
                -> Filter: ((e.FIPSCode = p.FIPStxt) and (e.`Value` is not null))  (cost=0.996 rows=0.9) (actual time=0.581..0.581 rows=0 loops=3124)
                    -> Single-row index lookup on e using PRIMARY (FIPSCode=cast(l.FIPS_Code as unsigned), Attribute='Bachelor\'s degree or higher; 2008-12')  (cost=0.996 rows=1)
 (actual time=0.581..0.581 rows=0 loops=3124)
 |
+-----------------------------------------------------------------------------------------------------
------------------------------------------------------------------------------------------------------
```

10. What can be improved in the application?
    a. We could update the datasets to include more recent data and incorporate additional factors, providing users with more options.

    b. We had planned to add housing information so that families could look at the statistics of what kind of house they wanted to live in and how many of those types of houses were in certain zip codes. Sadly, this ended up not working, so we scrapped the idea. However, with more time, this idea could become a big deal in our application.

c. We planned to add a function that allows a user to rate how important certain factors are to their “perfect” zipcode, but this idea was scrapped as it didn’t work as intended. We believe this application improves the website as it provides additional search options for the user to search for a ZIP code.

11. Final Division of Labor and Teamwork

a. Nicholas Chan: Worked on Front-end/Back-end, implementing the main menu and the ZIP code search. Also implemented the Top 5 Counties Stored Procedure and code. Created the UML Diagram in stage 2. Helped implement interactive map and favorites code.

b. John Driscoll: Worked on Front-end/Back-end; implementing user favorited ZIP codes and zipcode_search_by_preference(discarded). Gathered data files on various factors, including Education, GDP, Labor, and other socioeconomic data factors. This was especially the case in stage 1. Organized all data into a ZIP file so that other team members could access it easily and edited all Excel files that were unnormalized and formatted incorrectly. Worked on creating CSS code so that the website would look visually appealing. Found the original interactive map JavaScript code

c. Tony Liu: Worked on Front-end/Back-end, implementing interactive map and different SQL Advanced Programs. Developed a backup website with full-stack development under Node.js and have uploaded it to the branch.