Census Data Management System

Aditya Srivatsav. 50559985, Tarun Kumar Reddy Nallagari. 50560230

1. INTRODUCTION

A. Description

The proposed system, a central database to store demographic and income information collected from the people, which should be manipulated and analyzed for various purposes. Because we are using a relational database, structured and organized data will be obtained that facilitates easy access and management with it. We can, therefore, guarantee greater integrity and coherence of data, hence reduced errors that generally occur with less organized systems. The core focus of this system is building a strong model that will clearly predict-accurately or otherwise-whether people make more than $50,000 annually, using a wide set of predictor variables that include a person's age, education attainment, work class, marital status, and several socio-economic indicators that provide an integrated view of each citizen's economic state.

This data analysis gives critical information that shall form the basis of the income inequality problem and a trend that policymakers and organizations can use to develop appropriate initiatives for economic progress. From the relationship that the predictor variables hold with the income levels, the system can identify disparities in those areas and propose interventions. As shown, the realization of how education influences income could naturally lead to various initiatives that make higher education more accessible to underrepresented groups. Longitudinally, the database will be able to support research on change over time, thus adding even greater insight into the dynamics influencing the distribution of income. In the final analysis, the system is one not only of individual benefit-in analysis-but also in contributing to general society's goals of economic growth and equity.

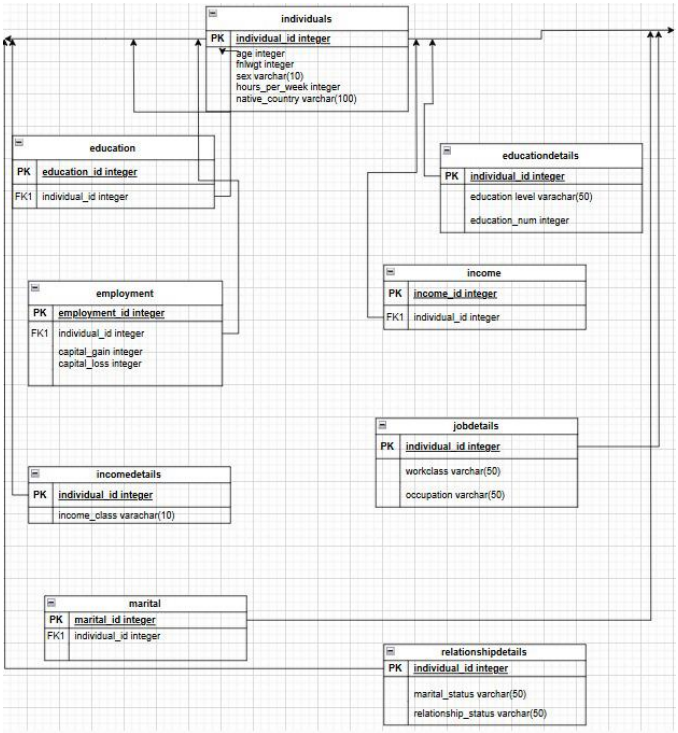
WHY DO YOU NEED A DATABASE INSTEAD OF AN EXCEL FILE:

The advantages of choosing a database system over using traditional Excel files lie in their: -scalability, where databases easily manage larger data sets with no degradation of performance, data integrity constraints enforce accuracy and consistency throughout their data life cycle, and a number of users can access a database at the same time without any risk of conflicts and data losses. Advanced querying through SQL enables complex data retrieval and manipulation beyond Excel’s capabilities, while user roles and permissions safeguard sensitive information, ensuring compliance with privacy standards.

Besides, databases are designed to handle complex relationships of data and indexing for swift, efficient retrieval. Unlike Excel, which works in a flat format, relational databases establish complicated relationships between tables, thus permitting the analysis of related data points. This is- especially useful for multi-dimensional datasets, supporting sophisticated operations like joins and aggregations. Databases also feature automated backups and recovery mechanisms, protecting against data loss due to system failures. Overall, a database system enhances data management and lays a solid foundation for future growth and analytical needs.

Target User:
The proposed database for demographic and income data will serve a variety of stakeholders,
including researchers, policymakers, social scientists, and non-profit organizations.
Users: The target users of the database are researchers and data analysts who may carry out extensive studies of earnings inequality, trends in demography, and socio-economic factors that affect income. Policy analysts might use data from the database to develop policies to reduce disparities in income and educational and job opportunity. Social scientists could also gain access to the data to inquire about the relationship existing among several socio-economic indicators and income levels. Non-
profit organizations focused on economic development could also access this data to design targeted programs and initiatives that would support underprivileged communities.
Administrators: A full-time DBA shall be appointed, who shall be responsible for the efficiency and security of the database. He shall grant permissions to users for access, back up data regularly, and optimize performance of the databases for all users. This DBA is contracted by a
research or governmental agency to ensure the reliability of the database as a source from which the stakeholders analyze and interpret the demographic and income data. After all, it will support informed decision-making and researching ways to meet the economic challenges at hand yet be managed by a qualified administrator.

1. ER DIAGRAM



2. DATABASE SCHEMAS

DROP THE TABLES

```
DROP TABLE IF EXISTS IncomeDetails;
DROP TABLE IF EXISTS RelationshipDetails;
DROP TABLE IF EXISTS Marital;
DROP TABLE IF EXISTS EducationDetails;
DROP TABLE IF EXISTS JobDetails;
DROP TABLE IF EXISTS Employment;
DROP TABLE IF EXISTS Individuals;
```

```
## CREATE INDIVIDUALS
CREATE TABLE Individuals (
    individual_id INT PRIMARY KEY,
    age INT NOT NULL,
    fnlwgt INT NOT NULL,
    sex VARCHAR(10) NOT NULL,
    hours_per_week INT NOT NULL,
    native_country VARCHAR(100)
);


## CREATE EMPLOYMENT
CREATE TABLE Employment (
    employment_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    capital_gain INT DEFAULT 0,
    capital_loss INT DEFAULT 0,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);

## CREATE JOB DETAILS
CREATE TABLE JobDetails (
    individual_id INT PRIMARY KEY,
    workclass VARCHAR(50) NOT NULL,
    occupation VARCHAR(50) NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);

## CREATE EDUCATION
CREATE TABLE Education (
    education_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);


## CREATE EDUCATION DETAILS
CREATE TABLE EducationDetails (
    individual_id INT PRIMARY KEY,
    education_level VARCHAR(50) NOT NULL,
    education_num INT NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);

## CREATE TABLE MARITAL STATUS

CREATE TABLE Marital (
    marital_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);

## CREATE TABLE RELATIONSHIP

CREATE TABLE RelationshipDetails (
    individual_id INT PRIMARY KEY,
    marital_status VARCHAR(50) NOT NULL,
    relationship VARCHAR(50) NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);


## CREATE TABLE INCOME
CREATE TABLE Income (
```

```
    income_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);

# CREATE TABLE INCOME
CREATE TABLE IncomeDetails (
    individual_id INT PRIMARY KEY,
    income_class VARCHAR(10) NOT NULL,
    FOREIGN KEY (individual_id)
REFERENCES Individuals(individual_id)
);
```

3. RELATIONS AND ATTRIBUTES

1 **Individuals Table**:

Purpose: Stores personal details of individuals.
Attributes:

- `individual_id` (Primary Key): Unique identifier for each individual.
- `age`: Age of the individual.
- `fnlwgt`: Final weight, representing the number of people the individual is representing.
- `sex`: Gender of the individual.
- `hours_per_week`: Number of hours worked per week.
- `native_country`: Country of origin for the individual.

2 **Employment Table**:

Purpose: Captures financial attributes related to employment.
Attributes:

- `employment_id` (Primary Key): Unique identifier for each employment record.
- `individual_id` (Foreign Key): Links to the `Individuals` table.
- `capital_gain`: Income from capital gains.
- `capital_loss`: Losses from capital transactions.

3 **Job Details Table**:

Purpose: Contains details about the job of the individual.
Attributes:

- `individual_id` (Primary Key and Foreign Key): Links to the `Individuals` table.
- `workclass`: Type of work sector (e.g., private, public).
- `occupation`: Job role or position.

4 **Education Table**:

Purpose: Tracks education details of individuals
Attributes:

- `education_id` (Primary Key): Unique identifier for education records.
- `individual_id` (Foreign Key): Links to the `Individuals` table.

5 **Education Details Table**:

Purpose: Stores detailed education information.
Attributes:

- `individual_id` (Primary Key and Foreign Key): Links to the `Individuals` table.
- `education_level`: Highest level of education attained (e.g., Bachelor's, Master's).

- `education_num`: Numeric representation of education (e.g., years of schooling)

6 **Marital Table**:

Purpose: A basic structure to track marital information.
Attributes:

- `marital_id` (Primary Key): Unique identifier for marital records.
- `individual_id` (Foreign Key): Links to the `Individuals` table.

6 **Relationship Details Table**:

Purpose: Tracks the relationship and marital status of individuals.
Attributes:

- `individual_id` (Primary Key and Foreign Key): Links to the `Individuals` table.
- `marital_status`: Marital status of the individual (e.g., married, single).
- `relationship`: Family relationship (e.g., spouse, own child).

7 **Income Table**:

Purpose: Captures the basic income information of individuals.
Attributes:
`income_id` (Primary Key): Unique identifier for income records.
`individual_id` (Foreign Key): Links to the `Individuals` table.

8 **Income Details Table**:

Purpose: Tracks income class details.
Attributes:

- `individual_id` (Primary Key and Foreign Key): Links to the `Individuals` table.
- `income_class`: Income classification (e.g., `>50K`, `<=50K`).

Functional Dependencies

From the initial schema, the following functional dependencies can be inferred:

1 **Individuals**:

`individual_id → age, fnlwgt, sex, hours_per_week, native_country`

2 **Employment**:

`employment_id → individual_id, capital_gain, capital_loss`

`individual_id → capital_gain, capital_loss`

3 **JobDetails**:

`individual_id → workclass, occupation`

4 **EducationDetails**:

`individual_id → education_level, education_num`

5 **Relationship Details**:

`individual_id → marital_status, relationship`

6 **Income Details**:

`individual_id → income_class`

BCNF Decomposition

To ensure the schema is in **BCNF (Boyce-Codd Normal Form)**:

- A relation is in BCNF if, for every non-trivial functional dependency X → Y X \to Y X→Y, X X X is a super key.

Step-by-Step Decomposition:

1 **Individuals**:

Already in BCNF since `individual_id` is the primary key and determines all other attributes.

2 **Employment**:

Dependency: `individual_id → capital_gain, capital_loss`

Decomposition:
Table 1: `Employment (employment_id, individual_id)`

Table 2:
`CapitalDetails(individual_id, capital_gain, capital_loss)`

3 **Job Details**:

Already in BCNF as `individual_id` is the key and determines all other attributes.

4 **Education Details**:

Already in BCNF as `individual_id` is the key and determines all other attributes.

5 **Relationship Details**:

Already in BCNF as `individual_id` is the key and determines all other attributes.

6 **Income Details**:

Already in BCNF as `individual_id` is the key and determines all other attributes.

Analysis

Individuals Table:

- This table stores personal information about each individual, including their `age`, `fnlwgt` (final weight), `sex`, `hours_per_week`, and `native_country`. The primary key is `individual_id`, which uniquely identifies each individual.

Employment Table:

- This table connects each individual's employment details to a unique `employment_id`. It includes `capital_gain` and `capital_loss` values to represent financial gains or losses from capital transactions. The `individual_id` acts as a foreign key linking back to the `Individuals` table.

Employment Table:

- This table stores details about each individual's capital transactions (`capital_gain` and `capital_loss`). The primary key is

`individual_id`, ensuring that each individual's capital data is uniquely recorded.

JobDetails Table:

- This table contains the work-related details of each individual, including their `workclass` (type of work sector) and `occupation` (job role). The primary key is `individual_id`, which links back to the `Individuals` table.
-

education Details Table:

- This table provides detailed information about each individual's education level, including `education_level` (e.g., Bachelor's, Master's) and `education_num` (number of years of education). The `individual_id` serves as the primary key and ensures that the data is uniquely associated with each individual.

relationship Details Table:

This table stores information about an individual's marital status (`marital_status`, e.g., married, single) and their relationship within a family (`relationship`, e.g., spouse, own child). The primary key is `individual_id`, linking it to the `Individuals` table.

Income Details Table:

- This table records income-related information, specifically the `income_class` of each Individual (`>50K` or `<=50K`). The primary key is `individual_id`, ensuring that income data is uniquely linked to each individual.

Advantages of BCNF Decomposition

- The Redundancy is going to be Eliminated from the database
- BCNF minimizes the anomalies from the Update, Insertion, Deletion which makes the CRUD operations effective.
- BCNF is going to Ensure that the Dependency Anomalies are going to be preserved

DATABASE IMPLEMENTATION

A. Creating Tables:

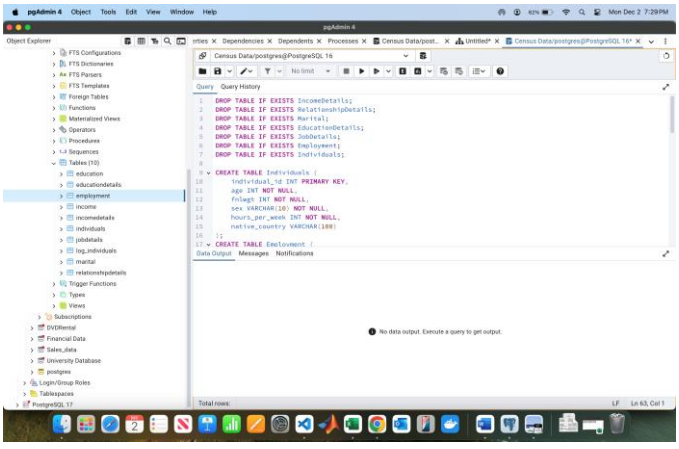


Fig. 1. Creation of Individuals table

```sql
CREATE TABLE Employment (
    employment_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    capital_gain INT DEFAULT 0,
    capital_loss INT DEFAULT 0,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

Fig. 2. Creation of Employment table

```sql
CREATE TABLE JobDetails (
    individual_id INT PRIMARY KEY,
    workclass VARCHAR(50) NOT NULL,
    occupation VARCHAR(50) NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

Fig. 3. Creation of Job Details table

```sql
CREATE TABLE Education (
    education_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

Fig. 4. Creation of Education table

```sql
CREATE TABLE EducationDetails (
    individual_id INT PRIMARY KEY,
    education_level VARCHAR(50) NOT NULL,
    education_num INT NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

Fig. 5. Creation of Education Details table

```sql
CREATE TABLE Marital (
    marital_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

Fig. 6. Creation of Marital table

```sql
CREATE TABLE RelationshipDetails (
    individual_id INT PRIMARY KEY,
    marital_status VARCHAR(50) NOT NULL,
    relationship VARCHAR(50) NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

Fig. 7. Creation of _relationship Details table

```sql
CREATE TABLE Income (
    income_id INT PRIMARY KEY,
    individual_id INT NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

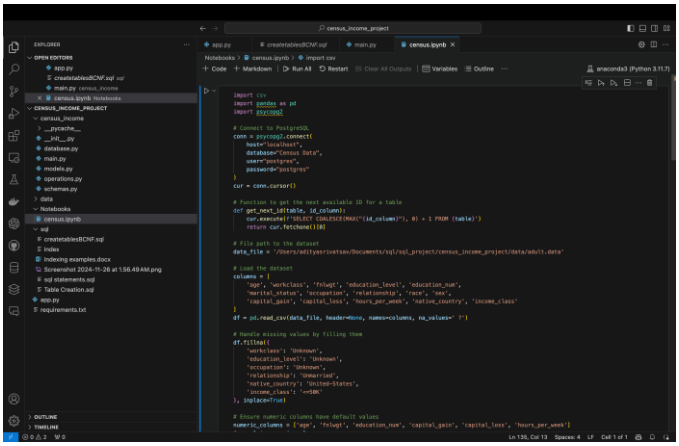
Fig. 8. Creation of Income table

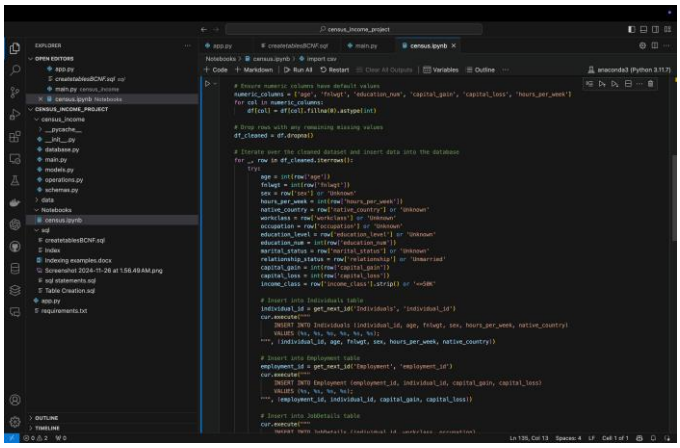
```sql
CREATE TABLE IncomeDetails (
    individual_id INT PRIMARY KEY,
    income_class VARCHAR(10) NOT NULL,
    FOREIGN KEY (individual_id) REFERENCES Individuals(individual_id)
);
```

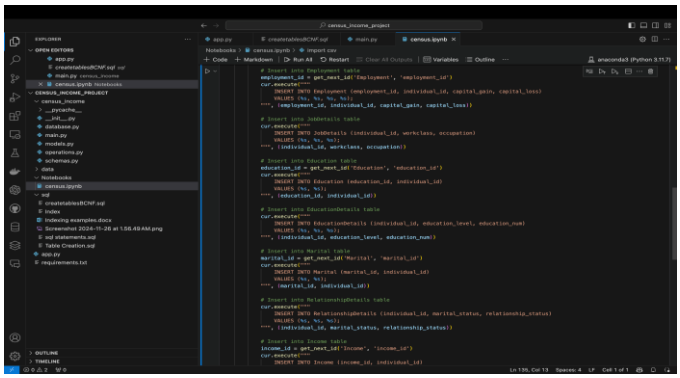
Fig 9. Creation of IncomeDetails

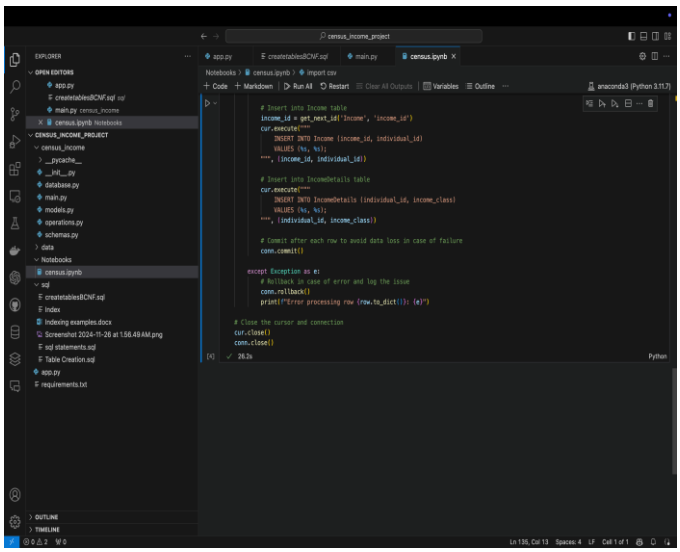
B) Insertion of Data:

- Inserted the data with the help of Python Implementation process as there are bulk data of 215K rows.
- Preprocessed Data: The data is preprocessed by removing the null values
- Database Connection: The data base connection is written with the Postgres URL to the Census Data
- Insertion: The data is inserted by normalization procedure into BCNF for the better output and to perform the CRUD operations in the API.



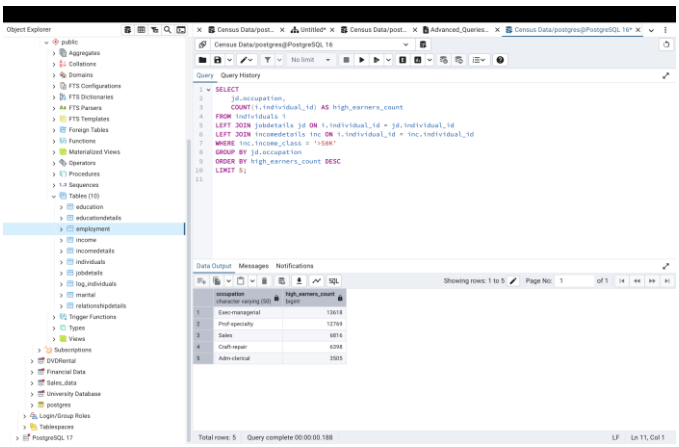




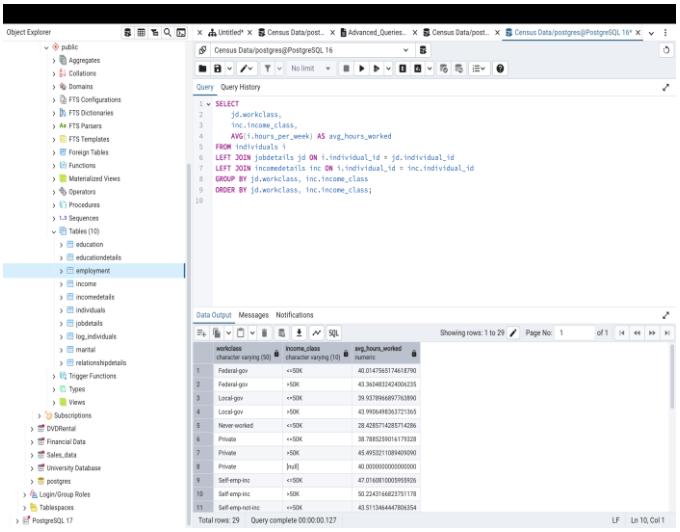


ADVANCED QUERIES

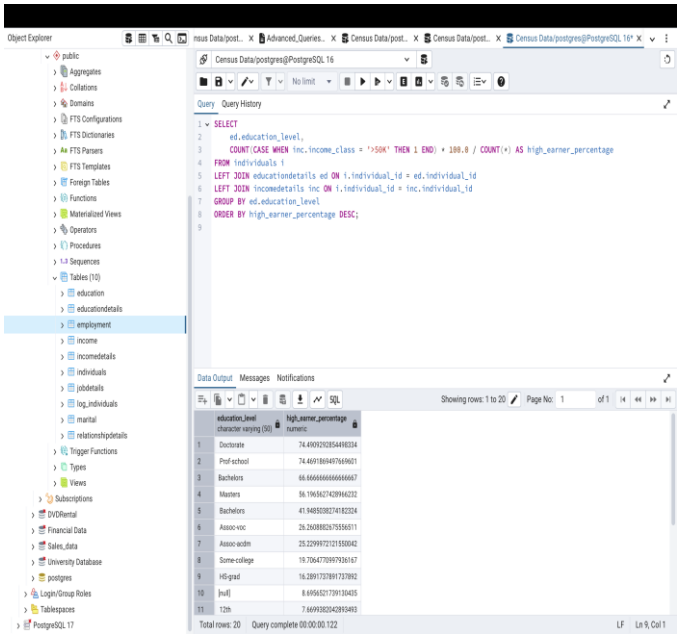
1. Top 5 OCCUPATIONS BY HIGH PAY EARNERS



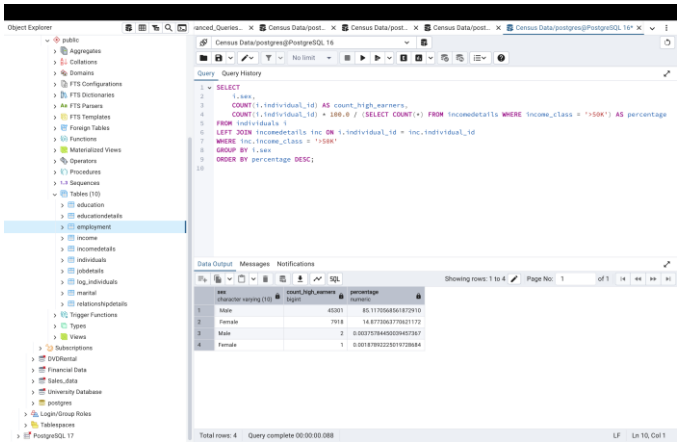
2. Average Hours Worked Per week by Work Class and IncomeClass



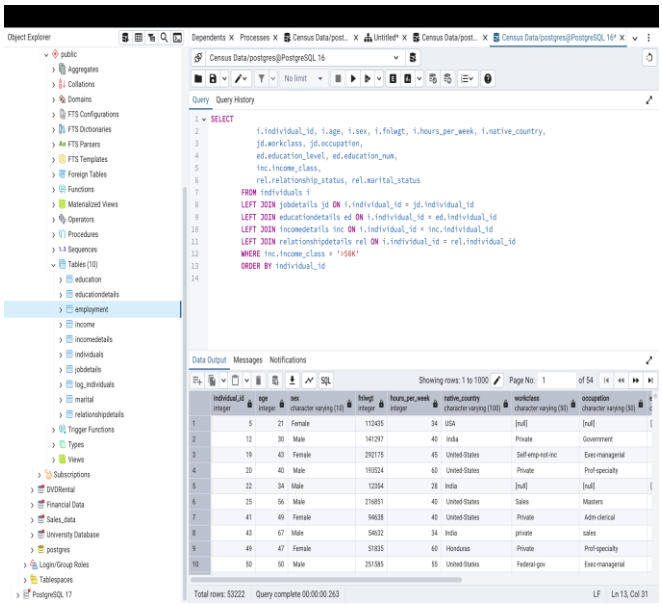
3. EDUCATION LEVELS WITH HIGHEST PROPORTION OF HIGH EARNERS



4. GENDER BREAKDOWN AMONG HIGH EARNERS



5. QUERY TO GET ALL THE TABLES



1. QUERY EXECUTION AND ANALYSIS

A) INDEXING:

Indexing in a database is a mechanism of optimizing the performance of a database using data structures to provide rapid retrieval of rows, thereby avoiding full table scans. Common types are primary, secondary, clustered, and non-clustered indexes.

This is done to enhance query performance for operations that involve SELECT, WHERE, JOIN, GROUP BY, and ORDER BY clauses by providing a direct path to the data instead of having to scan the entire table.

Advantages:

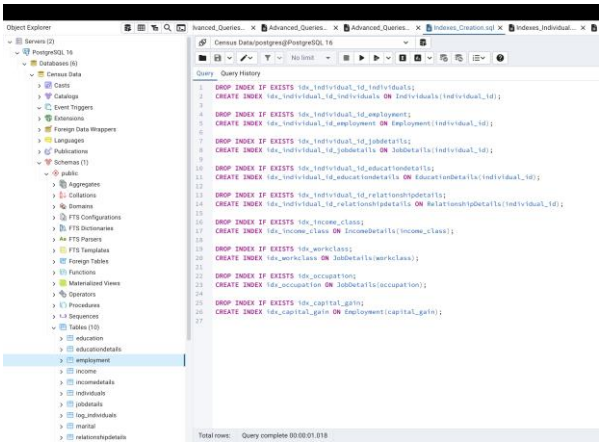
Speeds up data retrieval for big datasets.

Speeds up sorting, filtering, and aggregate functions. Reduces I/O operations, enhancing overall query efficiency.

Disadvantages:

Consumes additional storage space. Slows down write operations (e.g., INSERT, UPDATE, DELETE) as indexes need to be updated.
Requires periodic maintenance to prevent fragmentation.
refine query performance with appropriate indexes.

Creation of Indexes:



Index Creation for Primary Joins: The creation of indexes for the column individual_id in each of the related tables (Individuals, Employment, JobDetails, EducationDetails, RelationshipDetails, and IncomeDetails) for the purpose of optimizing JOIN operations.

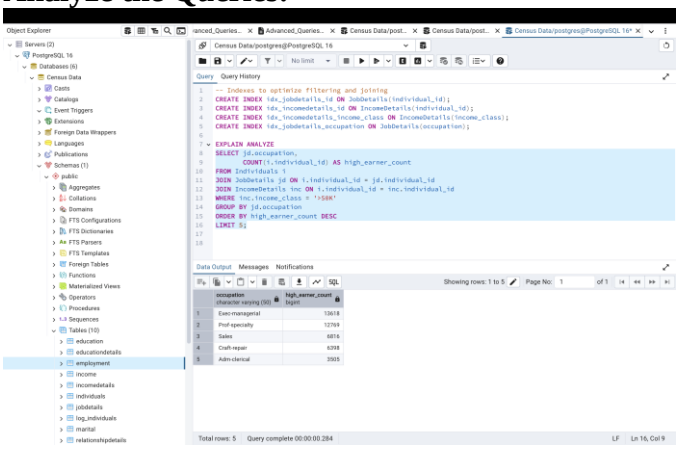
Optimization for Filtering and Grouping: Adding indexes to important columns such as income_class in IncomeDetails and occupation in JobDetails improves performance for WHERE, GROUP BY, and filtering clauses.

Sorting Efficiency: Indexes on the frequently sorted columns like 'capital_gain' of 'Employment' improve such operations as ORDER BY.

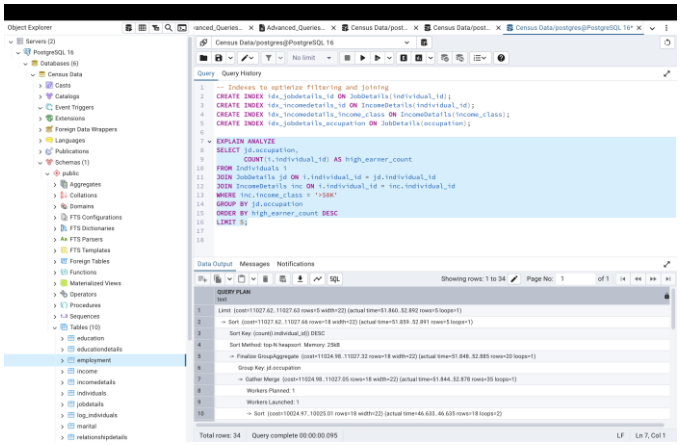
IF NOT EXISTS in Conditional Indexing: DROP INDEX IF EXISTS to clear the existing index, allowing for the creation without raising conflicts or duplication.

Broad Inclusive Index Approach: Most of the tables have significant columns indexed to help improve performance on queries executed to access data.

Analyze the Queries:



- This is the Image before Indexing and the time taken for the index is about the time of 284ms to execute the query.



Analysis after indexes:

- Optimized Query Execution: The query identifies the top 5 occupations with the highest count of high earners, using efficient filtering (WHERE), grouping (GROUP BY), and sorting (ORDER BY) with a LIMIT clause for minimal output.
- Effective Indexing: Individual_id, income_class, and occupation indexing at the individual level greatly improve the performance in joins, filtering, and grouping operations.

  Efficient Sorting and Memory Usage: The top-N heapsort method ensures efficient sorting, with minimal memory usage (25KB), indicating optimized handling of the dataset.

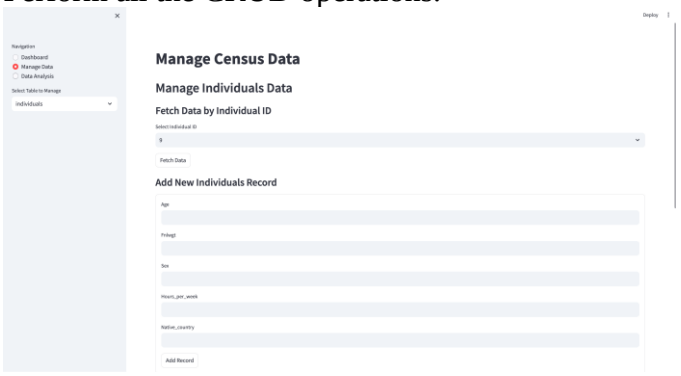
  Further Optimization: Potential improvements include increasing parallel workers for scalability, analyzing table statistics for better query planning, or using materialized views for frequent queries.

2. WEBPAGE AND API CALLS:

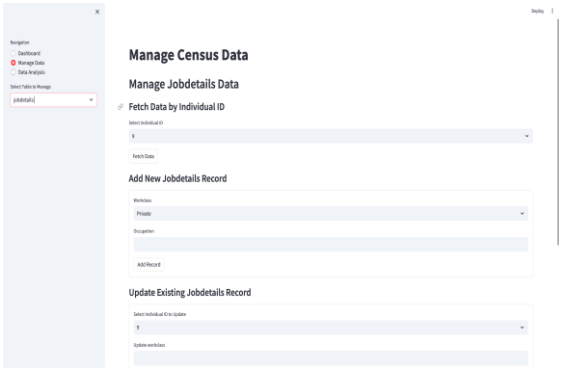
There are 3 different Navigations in the UI for the Project

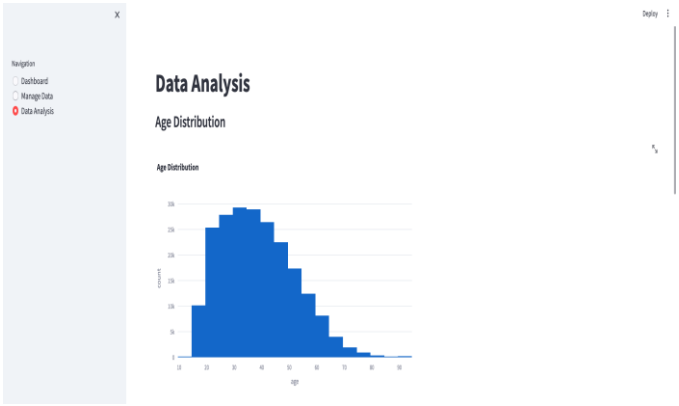
1. Dash Board: On the Dashboard we can see the Details of the Income data where we can fetch the data with Filter data with minimum of 5000 rows till maximum of 50000

2. The data is going to be obtained and can be converted to excel for the further studies
3. Manage Data: The Manage data is used to Perform all the CRUD operations.

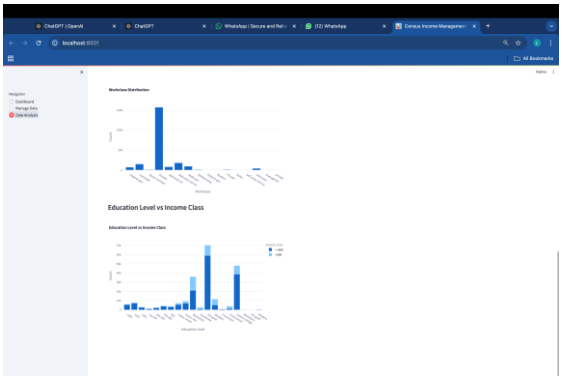


4. Design and Architecture: The Census_data has schemas, models, operations, main file where we have written all the Protocols to get the responses





5. In the UI we have Add/ Update options for all the tables. The individual table has no delete option as that is the main table. If we perform any deletion from the table, the other tables will be affected.
6. On the Dashboard you can fetch the metrics in the where we can get the salary/income whose >50k.
7. In Analytics tool we Implemented the Analysis of the Bar Graph with different Parameters as well.



Conclusion: The Project Census Data Management System is used to get the information regarding the Income and performed the end-to-end development to Create, Read, update and Deletion methods in the UI.

References:

https://archive.ics.uci.edu/dataset/20/census+income