CIS 556 – Final Project Report
(Fall 2024)

I. Team composition and responsibilities

1. Sagnik Roy:

   Responsible for data preparation, ER diagram design, writing DDL statements and Documentation

2. Shubhi Gupta:

   Responsible for the DML Statements, SQL queries and indexing, performance profiling and Documentation

II. Project Goal

The goal of this project is to design an efficient relational schema and database for library inventory optimization using the Goodreads Books dataset. This involves importing and structuring the data into PostgreSQL, implementing efficient querying mechanisms, and analyzing performance improvements through indexing and query optimization techniques. The project aims to provide insights into optimizing library operations, including categorization, book recommendations, and usage tracking.

This project will provide an opportunity to apply and test the following concepts covered in class:

- Designing a conceptual model (ER diagram) for a real-world dataset.
- Translating a conceptual model into a SQL schema.
- Writing effective SQL queries.
- Implementing and deploying database indexes.
- Evaluating the performance of SQL queries and indexes.

III. Attached Files

1. Raw dataset (csv format): *books.csv.zip*
2. Data transformation/cleaning: *Data_Cleaning.ipynb*
3. Transformed dataset (csv format): *final_cleaned_data.csv*
4. DDL and DML statements: *ddl_indexes.sql , ddl_schema.sql*
5. SQL queries + code for experiments: *test_queries.sql*

6. Analyze Tables: *analyze_tables.sql*
7. Visualization files : *visualize_benchmark_planning_execution.py, visualize_benchmark.py*
8. Shell Script files: *run_benchmark.sh , run_project.sh*
9. Final Project Report - *CIS 556 Final Project Report - Shubhi Gupta, Sagnik Roy.pdf*

IV. Dataset

The dataset for this project is sourced from Goodreads, a popular online platform for book reviews and recommendations.

It consists of book information including titles, authors, publication details, and genres for categorization and retrieval; user ratings and reviews for assessing book popularity and recommendations; and additional features such as ISBN numbers, language, and year of publication, which are crucial for inventory management and identifying unique copies.

Dataset link - https://www.kaggle.com/datasets/jealousleopard/goodreadsbooks

Below is the snapshot of the dataset-

A	B	C	D	E	F	G	H	I	J	K	L
bookID	title	authors	average_ratii	isbn	isbn13	language_code	num_pages	ratings_coun	text_reviews	publication_date	publisher
1	Harry Potter and the Half-Blood Prince (Harry Potter #6)	J.K. Rowling/Mary GrandPrâ€©	4.57	439785960	9.7804E+12	eng	652	2095690	27591	9/16/2006	Scholastic Inc.
2	Harry Potter and the Order of the Phoenix (Harry Potter #5)	J.K. Rowling/Mary GrandPrâ€©	4.49	439358078	9.7804E+12	eng	870	2153167	29221	09/01/04	Scholastic Inc.
4	Harry Potter and the Chamber of Secrets (Harry Potter #2)	J.K. Rowling	4.42	439554896	9.7804E+12	eng	352	6333	244	11/01/03	Scholastic
5	Harry Potter and the Prisoner of Azkaban (Harry Potter #3)	J.K. Rowling/Mary GrandPrâ€©	4.56	043965548X	9.7804E+12	eng	435	2339585	36325	05/01/04	Scholastic Inc.
8	Harry Potter Boxed Set Books 1-5 (Harry Potter #1-5)	J.K. Rowling/Mary GrandPrâ€©	4.78	439682584	9.7804E+12	eng	2690	41428	164	9/13/2004	Scholastic
9	Unauthorized Harry Potter Book Seven News: "Half-Blood Prince	W. Frederick Zimmerman	3.74	976540606	9.781E+12	en-US	152	19	1	4/26/2005	Nimble Books
10	Harry Potter Collection (Harry Potter #1-6)	J.K. Rowling	4.73	439827604	9.7804E+12	eng	3342	28242	808	09/12/05	Scholastic
12	The Ultimate Hitchhiker's Guide: Five Complete Novels and One	Douglas Adams	4.38	517226952	9.7805E+12	eng	815	3628	254	11/01/05	Gramercy Books
13	The Ultimate Hitchhiker's Guide to the Galaxy (Hitchhiker's Guid	Douglas Adams	4.38	345453743	9.7803E+12	eng	815	249558	4080	4/30/2002	Del Rey Books
14	The Hitchhiker's Guide to the Galaxy (Hitchhiker's Guide to the G	Douglas Adams	4.22	1400052920	9.7814E+12	eng	215	4930	460	08/03/04	Crown
16	The Hitchhiker's Guide to the Galaxy (Hitchhiker's Guide to the G	Douglas Adams/Stephen Fry	4.22	739322206	9.7807E+12	eng	6	1266	253	3/23/2005	Random House Audio
18	The Ultimate Hitchhiker's Guide (Hitchhiker's Guide to the Galax	Douglas Adams	4.38	517149257	9.7805E+12	eng	815	2877	195	1/17/1996	Wings Books

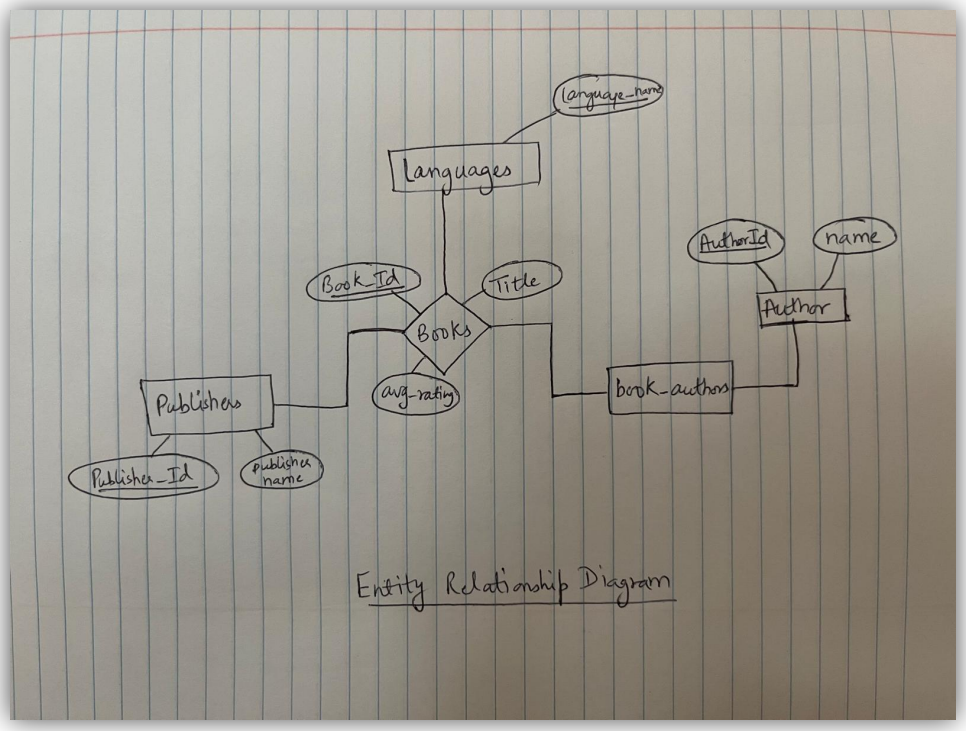
V. Data Transformation

The following cleaning and transformation steps were applied to the dataset to ensure data consistency, integrity, and compatibility with PostgreSQL:

1. **Dropped Unnamed Column:** A column without a header and containing only a few values was removed.
2. **Title Cleanup:** Leading and trailing spaces were removed from the "title" column.
3. **Authors Cleanup:** Non-alphabetic characters were removed, and extra spaces were trimmed from the "authors" column.
4. **Authors Splitting:** Multiple authors listed together were split using the '/' delimiter.
5. **ISBN Standardization:** The "isbn" and "isbn13" columns were converted to strings. Unwanted characters were removed while retaining the 'X' if it appeared as the last character, indicating the 10th digit.

6. **Rating Conversion:** The "average_rating" column was converted to a numeric data type, and rows with missing values were removed.
7. **Reviews Count Conversion:** The "text_reviews_count" column was converted to a numeric data type.
8. **Language Code Cleanup:** Digits were removed from the "language_code" column.
9. **Language Mapping:** "language_code" values were mapped to their corresponding full language names in a new "language_name" column, making the old column redundant and allowing its removal.
10. **Publisher Name:** Non-alphabetic characters in the "publisher" column were kept to preserve accuracy for books published in languages other than English.
11. **Publication Date Formatting:** The "publication_date" column was converted to a datetime format (YYYY-MM-DD) compatible with PostgreSQL. Rows with missing or invalid publication dates were removed.
12. **Unique Identifiers:** A unique "publisher_id" was generated for each publisher, ensuring data normalization.
13. **Author IDs:** A unique "author_id" was generated for each author to maintain relational consistency.
14. **Table Creation:** Separate CSV files were created for authors, book_authors, publishers, and languages to support a well-structured database schema.
15. **Final Dataset Export:** The cleaned dataset was saved as "final_cleaned_data.csv," ensuring compatibility with PostgreSQL.
16. **Dataset Shape:** After the cleaning process, the dataset consisted of **11,117 rows** and **14 columns**.

VI. ER Diagram



VII. DDL (Data Definition Language) Statements

The ER diagram was translated into DDL statements tailored for the PostgreSQL environment-

```
-- Drop existing tables if they exist to start fresh
DROP TABLE IF EXISTS book_authors CASCADE;
DROP TABLE IF EXISTS books CASCADE;
DROP TABLE IF EXISTS authors CASCADE;
DROP TABLE IF EXISTS publishers CASCADE;
DROP TABLE IF EXISTS languages CASCADE;

-- Create authors table
CREATE TABLE authors (
    authorID VARCHAR(255) PRIMARY KEY,
    name VARCHAR(1000) NOT NULL
);

-- Create publishers table
CREATE TABLE publishers (
    publisherID INT PRIMARY KEY,
    publisher_name VARCHAR(100) NOT NULL
);

-- Create languages table
CREATE TABLE languages (
```

```
    language_name VARCHAR(50) PRIMARY KEY
);

-- Create books table
CREATE TABLE books (
    bookID INT PRIMARY KEY,
    title VARCHAR(1000) NOT NULL,
    authors VARCHAR(1000),
    average_rating DECIMAL(3, 2),
    isbn TEXT CHECK (isbn ~ '^[0-9X]+$'),
    isbn13 TEXT CHECK (isbn13 ~ '^[0-9X]+$'),
    num_pages INT,
    ratings_count INT,
    text_reviews_count INT,
    publication_date DATE,
    publisher VARCHAR(200) NOT NULL,
    language_name VARCHAR(50),
    publisherID INT,
    authorID VARCHAR(255),
    FOREIGN KEY (publisherid) REFERENCES publishers(publisherid),
    FOREIGN KEY (language_name) REFERENCES languages(language_name)
);

-- Create book_authors table
CREATE TABLE book_authors (
    bookID INT REFERENCES books(bookID),
    authorID VARCHAR(255) REFERENCES authors(authorID),
    PRIMARY KEY (bookID, authorID)
);
```

VIII. DML Statements

We populated our schema using the following DML statements -

```
-- Load data into the authors table
\COPY authors(name,authorID) FROM '../data/authors.csv' WITH CSV HEADER ENCODING
'UTF8';

-- Load data into the publishers table
\COPY publishers(publisherid,name) FROM '../data/publishers.csv' WITH CSV HEADER
ENCODING 'UTF8';

-- Load data into the languages table
\COPY languages(language_name) FROM '../data/languages.csv' WITH CSV HEADER
ENCODING 'UTF8';
```

-- Load data into the books table
\COPY
books(bookID,title,authors,average_rating,isbn,isbn13,num_pages,ratings_count,text_reviews_count,publication_date,publisher, language_name, publisherID,authorID) FROM '../data/final_cleaned_data_new.csv' WITH CSV HEADER ENCODING 'UTF8';

-- Load data into the book_authors table
\COPY book_authors(bookID, authorid) FROM '../data/book_authors.csv' WITH CSV HEADER ENCODING 'UTF8';

IX. Data Verification in PostgreSQL

To ensure the successful insertion of data into PostgreSQL, we executed the following queries to inspect sample records from each table:

- SELECT * FROM authors LIMIT 5;
- SELECT * FROM books LIMIT 5;
- SELECT * FROM publishers LIMIT 5;
- SELECT * FROM languages LIMIT 5;
- SELECT * FROM book_authors LIMIT 5;

X. SQL Queries with Explanation and Results

We created nine distinct SQL queries based on the available data. Below, each query is described along with its explanation and the observed results.

Common Steps for Query Execution:

Step 1:

To evaluate query performance, we used the EXPLAIN ANALYZE <query> command.

Queries were executed under three different conditions:

1. **Without Statistics or Indexes** – Initial execution without optimization features.
2. **With Statistics but No Indexes** – After collecting statistics using the ANALYZE command.
3. **With Both Statistics and Indexes** – After applying relevant indexes.

Step 2:

To gather table statistics, we ran the following commands:

```
ANALYZE VERBOSE books;
ANALYZE VERBOSE publishers;
ANALYZE VERBOSE book_authors;
ANALYZE VERBOSE authors;
ANALYZE VERBOSE languages;
```

Step 3:

We created indexes to optimize query performance using the following commands:

```
CREATE INDEX IF NOT EXISTS idx_books_avg_rating134 ON books (average_rating);

CREATE INDEX IF NOT EXISTS idx_books_text_reviews134 ON books (text_reviews_count);

CREATE INDEX IF NOT EXISTS idx_books_publication_date134 ON books (publication_date);

CREATE INDEX IF NOT EXISTS idx_publishers_name134 ON publishers (name);

CREATE INDEX IF NOT EXISTS idx_book_authors_bookid134 ON book_authors (bookID);

CREATE INDEX IF NOT EXISTS idx_book_authors_authorid134 ON book_authors (authorID);

CREATE INDEX IF NOT EXISTS idx_authors_name134 ON authors (name);

CREATE INDEX IF NOT EXISTS idx_book_authors_author_book134 ON book_authors
(authorID, bookID);

CREATE INDEX IF NOT EXISTS idx_books_publisher_date134 ON books (publisherID,
publication_date);

CREATE INDEX IF NOT EXISTS idx_books_author_rating134 ON books (average_rating,
language_name);

CREATE INDEX IF NOT EXISTS idx_books_no_reviews134 ON books (text_reviews_count);

CREATE INDEX IF NOT EXISTS idx_books_high_ratings_reviews134 ON books (average_rating
DESC, text_reviews_count DESC);

CREATE INDEX IF NOT EXISTS idx_books_publisher_name134 ON publishers (name);

CREATE INDEX IF NOT EXISTS idx_books_ratings_reviews134 ON books (ratings_count,
text_reviews_count);
```

```
CREATE INDEX IF NOT EXISTS idx_books_language134 ON books (language_name,
average_rating);

CREATE INDEX IF NOT EXISTS idx_books_author_languages134 ON books (language_name);

CREATE INDEX IF NOT EXISTS idx_books_rare_languages134 ON books (language_name);
```

Comparison Table: Execution and Planning Times (ms)

Query	Scenario	Execution Time (ms)	Planning Time (ms)
Query 1: Top 10 Authors by Books	Without Stats or Indexes	56.833	0.385
	With Stats but No Indexes	56.727	0.380
	With Both Stats and Indexes	56.694	0.472
Query 2: Most Published Publishers	Without Stats or Indexes	1.895	0.307
	With Stats but No Indexes	1.893	0.275
	With Both Stats and Indexes	1.978 (▼0.11%)	0.282
Query 3: Average Rating for Each Author	Without Stats or Indexes	37.487	0.402
	With Stats but No Indexes	37.348	0.369
	With Both Stats and Indexes	36.092 (▼3.72%)	0.377
Query 4: Authors with No Reviews	Without Stats or Indexes	12.187	0.376
	With Stats but No Indexes	12.307	0.363
	With Both Stats and Indexes	12.068	0.373
Query 5: Books with High Ratings and Reviews Published by Active Publishers	Without Stats or Indexes	6.727	0.543
	With Stats but No Indexes	6.703	0.542
	With Both Stats and Indexes	6.963	0.553

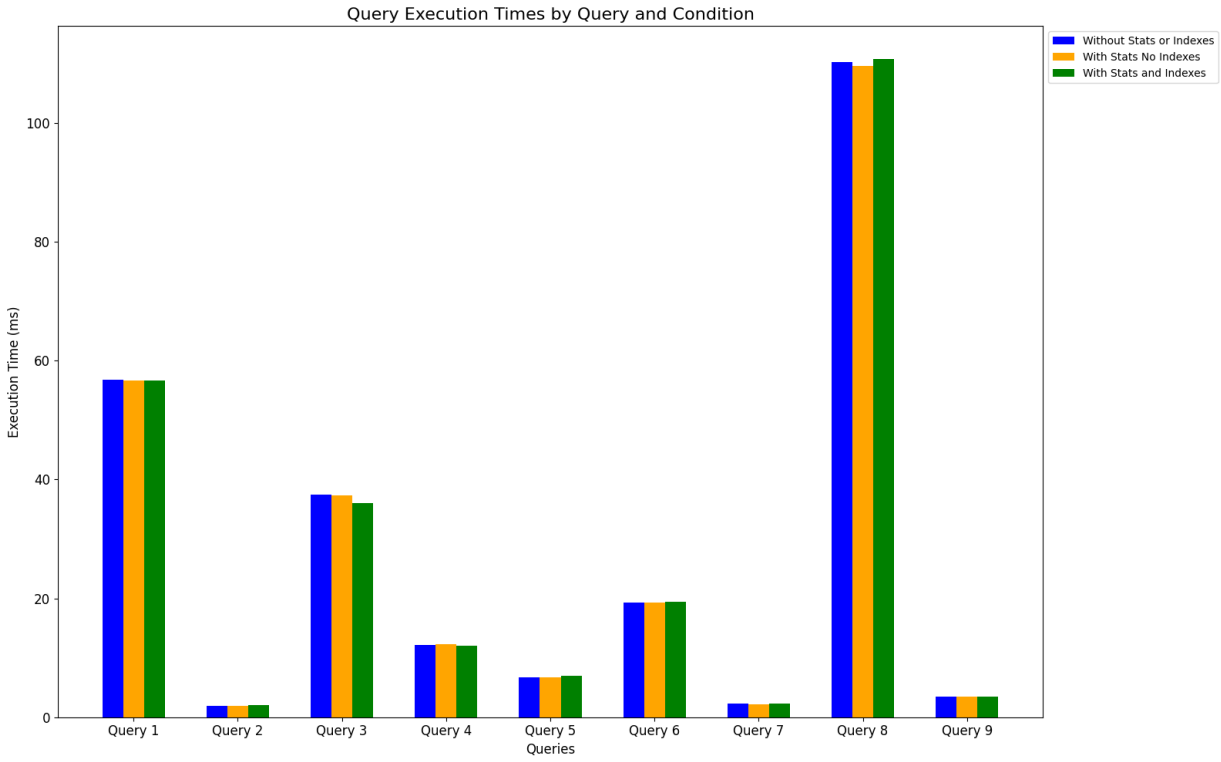
Query 6: Authors with the Highest Number of Ratings and Reviews	Without Stats or Indexes	19.256	0.396
	With Stats but No Indexes	19.273	0.400
	With Both Stats and Indexes	19.397	0.393
Query 7: Average Ratings by Language	Without Stats or Indexes	2.262	0.169
	With Stats but No Indexes	2.234	0.161
	With Both Stats and Indexes	2.309	0.173
Query 8: Authors Writing in Multiple Languages	Without Stats or Indexes	110.244	0.374
	With Stats but No Indexes	109.639	0.384
	With Both Stats and Indexes	110.788	0.370
Query 9: Books in Rare Languages	Without Stats or Indexes	3.426	0.224
	With Stats but No Indexes	3.428	0.218
	With Both Stats and Indexes	3.433	0.226

NOTE: *The execution and planning times were measured on macOS using PostgreSQL version 13.3. The results may vary on other operating systems such as Windows, due to potential differences in system configurations and optimizations.*

XI. Benchmarks

This section presents the observed performance metrics for each query execution, along with the corresponding query plans generated by the PostgreSQL optimizer. Each query was executed N times, and the average execution and planning times were recorded to ensure accurate benchmarking results.

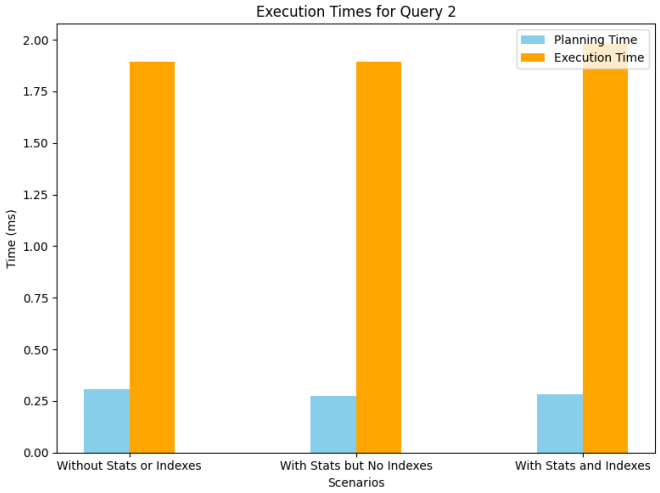
Below is the combined plot comparing the **execution times** of all nine queries across the above three scenarios.

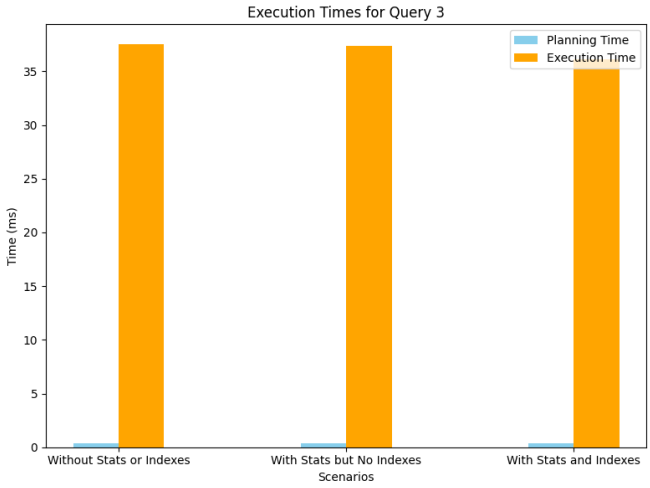


To provide deeper insights, we selected a few representative queries and examined both planning and execution times:

- **Query 2: Most Published Publishers** -
  Execution time improved from **1.895 ms** (without statistics or indexes) to **1.893 ms** (with statistics only), showing a **0.11% improvement.** This demonstrates how gathering statistics can slightly optimize query execution even without indexes.

- **Query 3: Average Rating for Each Author** -
  Execution time improved from **37.487 ms** (without statistics or indexes) to **36.092 ms** (with both statistics and indexes), showing a **3.72% improvement.** This highlights how gathering statistics and applying relevant indexes can enhance the performance of aggregation queries.





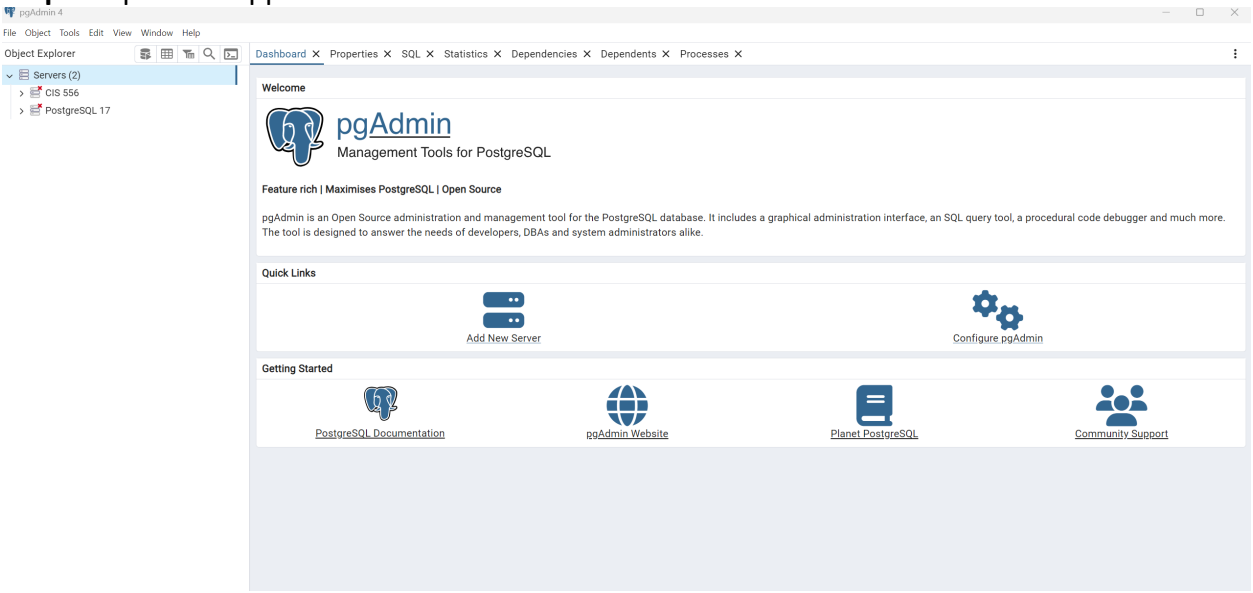
XII. Instructions for reproducing the experiments

Folder Structure of the zip file

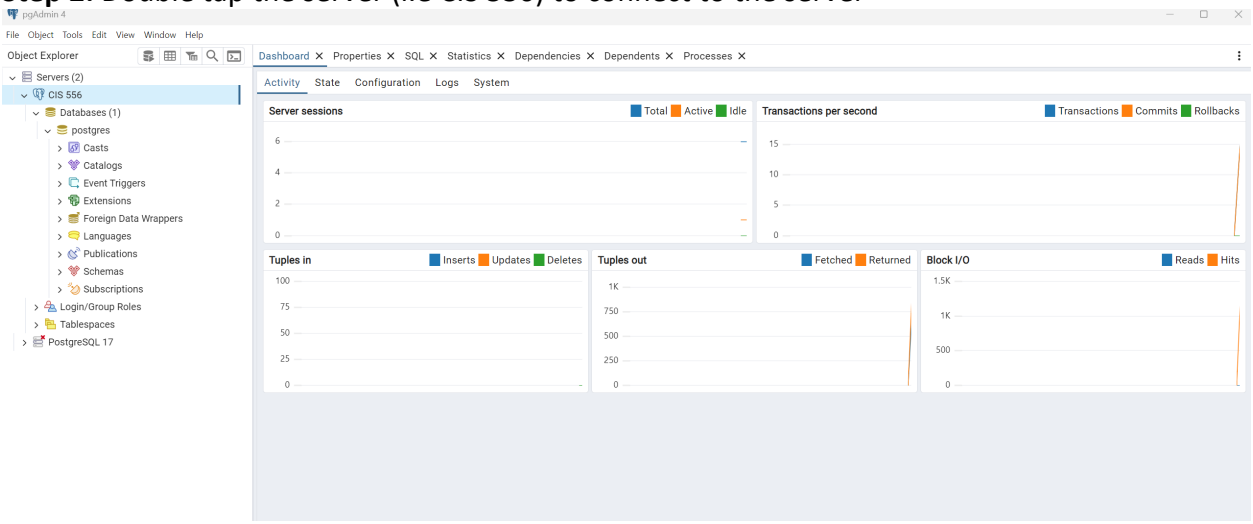
CIS 556 Final Project/
|— Data/
| |— Contains cleaned, raw, and generated files from executing scripts in the PostgreSQL environment.
|— scripts/
| |— Contains all SQL, shell (`.sh`), and Python files.
| |— output/
| |— Stores output files generated from script executions.

For Windows (using pgAdmin App)-

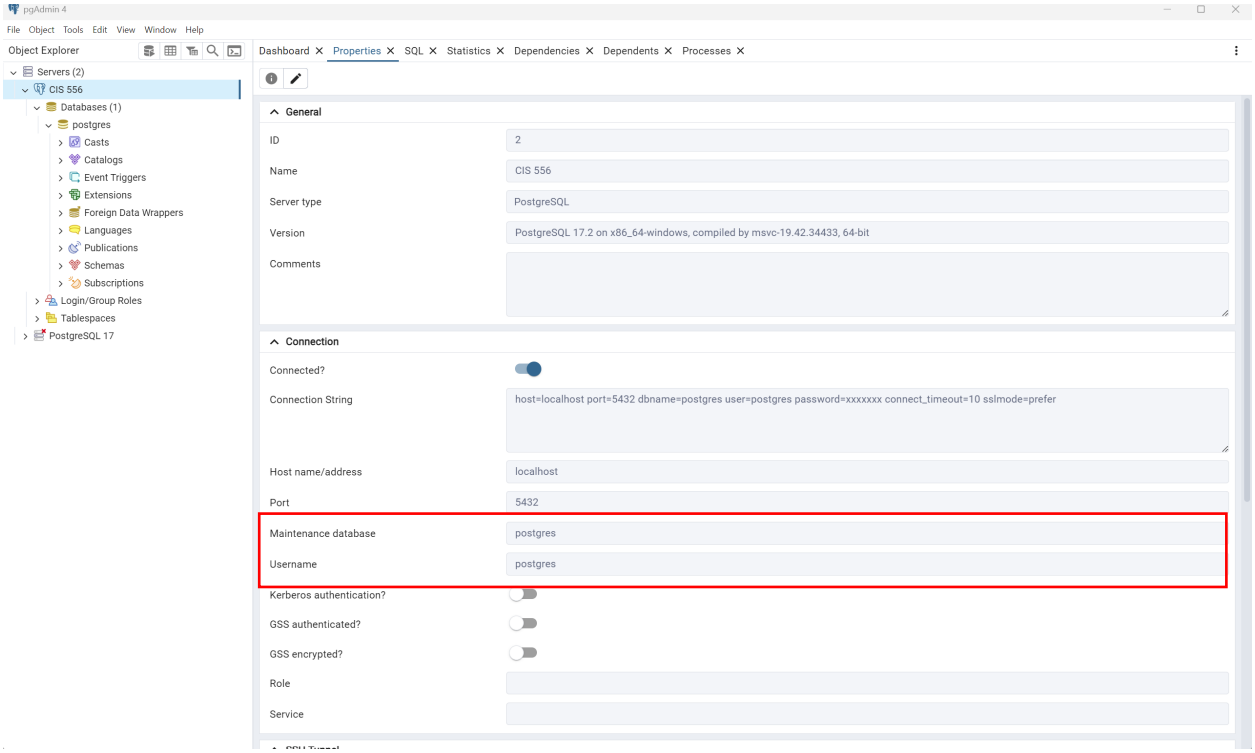
Step 1: Open the Application.



Step 2: Double tap the server (i.e CIS 556) to connect to the server



Step 3: To get username and database name, select PostgreSQL Server name (i.e CIS 556) and click “Properties” on top right to get the details



For macOS (Using PostgreSQL App)

1. **Launch PostgreSQL App:** Open the elephant PostgreSQL app.
2. **Start the Server:** Ensure the server is running. If not, start it.
3. **Open Terminal:** Launch the terminal.
4. **Enter PostgreSQL Environment:** Type **psql** to access the PostgreSQL environment.
5. **Check Current Database and Username:** Use the following commands to identify the default username and database:

```
SELECT current_user;
SELECT current_database();
```

   - Note: The default username is your Mac system username.

6. **Provide Password if Prompted:** Use the password set during the PostgreSQL app setup.

For macOS (Without PostgreSQL App)

Prerequisite: PostgreSQL must already be installed.

1. **Open Terminal:** Launch the terminal.
2. **Connect to PostgreSQL:** Use your Mac system username to connect.

```
psql -U $USER
```

3. **Check Current Database and Username:** Run the following SQL commands:

```
SELECT current_user;
SELECT current_database();
```

Conclusion:
You will need the PostgreSQL username and database name to proceed with the project. Follow either of the above methods to retrieve these details.

For Windows Execution

1. **Navigate to Scripts Folder:** Locate the folder containing the run_project and run_benchmark files. For example, the "scripts" folder.
2. **Open Git Bash:** Right-click the folder and select **Open Git Bash Here**.
3. **Set Environment Variables:** Run the following commands:

```
export PGPASSWORD="your_password"
export PGUSER=" your_username "
export PGDATABASE="your_database"
```

   - Note: Above details are to be filled by the user.

4. **Execute the Scripts:** Run the following command in the bash environment:

```
bash ./run_project.sh && ./run_benchmark.sh
```

For macOS Execution

1. **Open Terminal:** Launch the terminal.
2. **Enter PostgreSQL Environment:** Type **psql** to access the PostgreSQL environment.
3. **Run Scripts:** Use the following command to navigate to the scripts directory and execute the files:

```
\! cd ~/Downloads/CIS\ 556\ Final\ Project/scripts && ./run_project.sh &&
./run_benchmark.sh
```

   - Note: The \! command allows running shell commands from the PostgreSQL environment.
   - Replace ~/Downloads/CIS\ 556\ Final\ Project/scripts with the actual path where your .sh files are located.

4. **Provide Credentials:** Enter the username and database name when prompted (only once).
5. **Permission Denied -** Use this command chmod +x run_project.sh run_benchmark.sh. Then again run the command in Step 3.

XIII. Final Conclusion

This project focused on building a library inventory management system using the Goodreads Books dataset and PostgreSQL. We designed a clear ER model and created a SQL database to manage data about books, authors, publishers, and ratings. The goal was to organize the data efficiently and run queries to gain useful insights, such as identifying top-rated books, popular authors, and publishers with the most books.

We developed a variety of SQL queries to support key aspects of library inventory optimization, focusing on authors, publishers, books, and languages. These queries included identifying popular authors based on the total number of books published, publishers with the most publications over time, and books with high ratings and reviews. Additionally, we calculated average book ratings per author and identified authors writing in multiple languages to support multilingual cataloging.

We applied key database concepts such as data modelling, indexing, and query optimization. By running performance tests, we improved query execution times using indexes and analyzing query plans. This process helped us understand how database design impacts performance. Overall, the project provided hands-on experience in creating and managing a database system while solving real-world data challenges.