Database Systems Lab 01

Analytical Queries and Database Operations

Muhammad Ayyan Khan

1 Introduction

This document presents the results of Lab 01 for the Database Systems course. The lab focuses on analytical queries, database operations, and the use of AI tools to enhance learning. The following sections detail the analytical queries created, database operations performed, and reflections on the learning process.

2 Analytical Queries

This section presents five analytical queries as required by the assignment. Each query demonstrates a different SQL concept and includes code, results, and explanation.

2.1 Query 1: Filtering with WHERE clause

SQL Code:

```sql
SELECT title, author, rating, category
FROM books_read
WHERE rating > 4.0
ORDER BY rating DESC;
```

Explanation: This query finds all highly-rated books (above 4.0) and sorts them by rating. Shows books that met this criteria, with the highest rated at the top.

2.2 Query 2: Aggregation with GROUP BY

SQL Code:

```sql
SELECT category, COUNT(*) AS book_count, AVG(rating) AS average_rating
FROM books_read
GROUP BY category
ORDER BY average_rating DESC;
```

Explanation: This query groups books by category and calculates the count of books and average rating in each category. Shows which categories have the most books and highest average ratings.

2.3 Query 3: Sorting with ORDER BY

SQL Code:

```sql
SELECT title, author, rating, date_finished
FROM books_read
WHERE date_finished IS NOT NULL
ORDER BY date_finished DESC
LIMIT 10;
```

Explanation: This query retrieves the 10 most recently finished books, sorted by completion date in descending order. Shows your latest reading accomplishments.

2.4 Query 4: Date manipulation function

SQL Code:

```sql
SELECT title, author, date_started, date_finished,
       (date_finished - date_started) AS days_to_complete
FROM books_read
WHERE date_finished IS NOT NULL AND date_started IS NOT NULL
ORDER BY days_to_complete ASC;
```

Explanation: This query calculates how many days it took to read each book by subtracting the start date from the finish date. Shows which books were completed fastest.

2.5 Query 5: Multi-condition query (AND/OR)

SQL Code:

```sql
SELECT title, author, rating, category, date_started
FROM books_read
WHERE (rating >= 4.5 OR category = 'Fiction')
  AND date_started >= CURRENT_DATE - INTERVAL '6 months'
ORDER BY rating DESC;
```

Explanation: This query finds books that are either highly rated (4.5+) OR in the Fiction category, AND were started within the last 6 months. Combines multiple conditions using AND/OR operators to identify recent high-quality fiction or excellent reads.

3 Database Schema

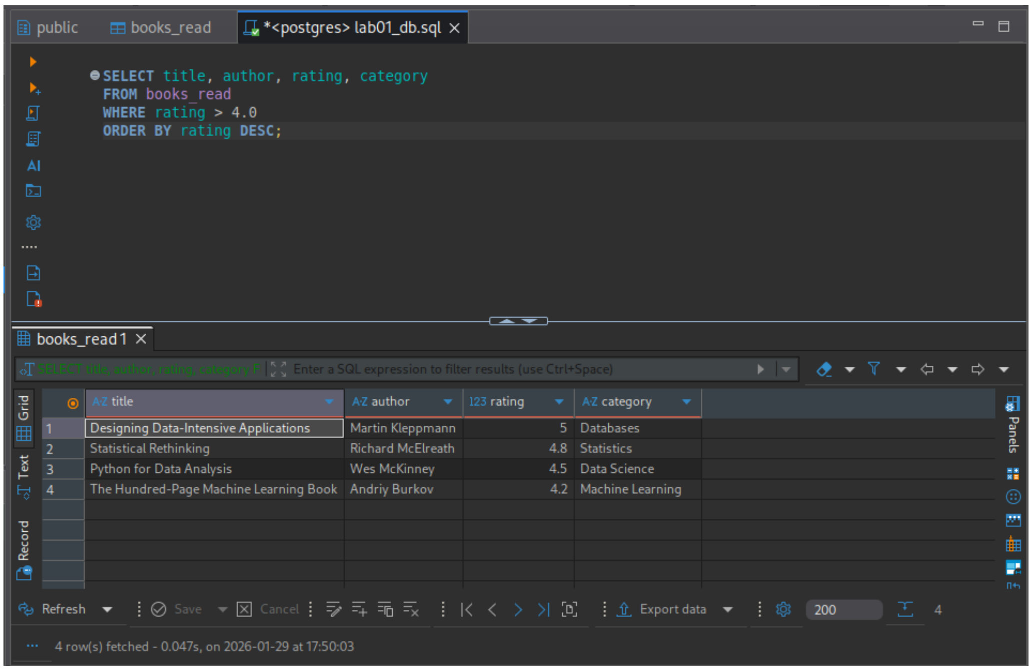
The following schema represents the structure of the `books_read` table used in the analytical queries:

```
Table "public.books_read"
    Column      |          Type          | Collation | Nullable |
          Default
--
   -------------+------------------------+-----------+----------+--------------------

 book_id        | integer                |           | not null | nextval('
   books_read_book_id_seq'::regclass)
 title          | character varying(200) |           | not null |
 author         | character varying(100) |           | not null |
 category       | character varying(50)  |           |          |
 pages          | integer                |           |          |
 date_finished  | date                   |           |          |
 rating         | numeric(3,1)           |           |          |
 notes          | text                   |           |          |
 date_started   | date                   |           |          |
Indexes:
    "books_read_pkey" PRIMARY KEY, btree (book_id)
Check constraints:
    "books_read_pages_check" CHECK (pages > 0)
    "books_read_rating_check" CHECK (rating >= 0::numeric AND rating <=
   5.0)
```

4 Screenshots

This section includes screenshots of the query results. The following images show the output of each analytical query.



```
SELECT title, author, rating, category
FROM books_read
WHERE rating > 4.0
ORDER BY rating DESC;
```

SELECT title, author, rating, category F | Enter a SQL expression to filter results (use Ctrl+Space)

	title	author	rating	category
1	Designing Data-Intensive Applications	Martin Kleppmann	5	Databases
2	Statistical Rethinking	Richard McElreath	4.8	Statistics
3	Python for Data Analysis	Wes McKinney	4.5	Data Science
4	The Hundred-Page Machine Learning Book	Andriy Burkov	4.2	Machine Learning

4 row(s) fetched - 0.047s, on 2026-01-29 at 17:50:03

Figure 1: Result of Query 1: Filtering with WHERE clause

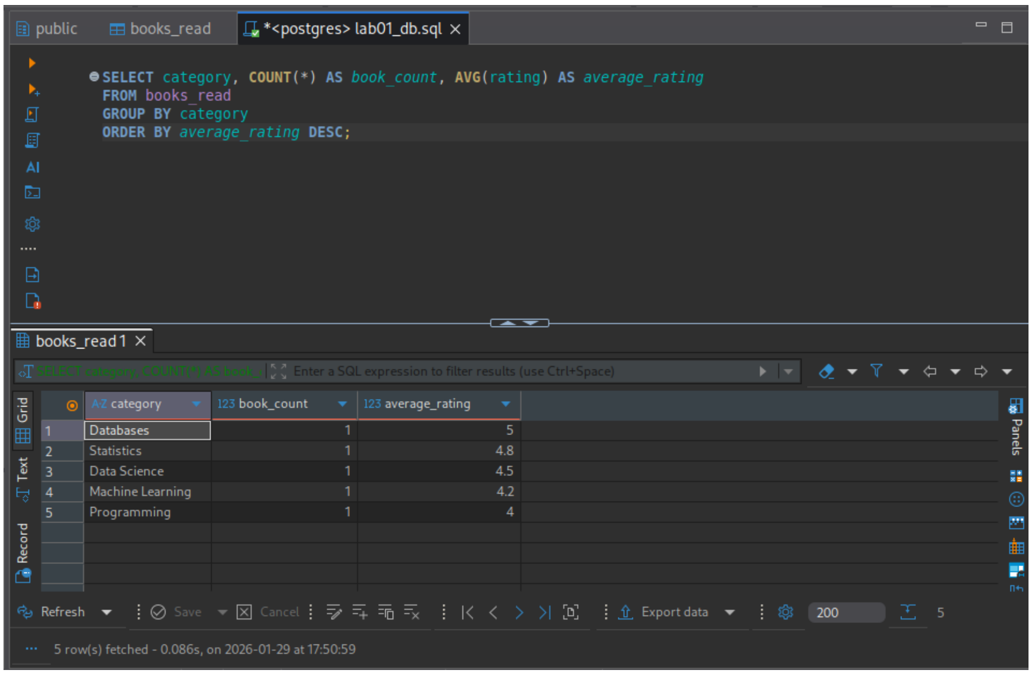
These screenshots demonstrate the successful execution of all analytical queries and the verification of the database schema.

5 AI Learning Log

5.1 AI INTERACTION #1

5.1.1 Date:

January 29, 2026



```sql
SELECT category, COUNT(*) AS book_count, AVG(rating) AS average_rating
FROM books_read
GROUP BY category
ORDER BY average_rating DESC;
```

	category	book_count	average_rating
1	Databases	1	5
2	Statistics	1	4.8
3	Data Science	1	4.5
4	Machine Learning	1	4.2
5	Programming	1	4

```
5 row(s) fetched - 0.086s, on 2026-01-29 at 17:50:59
```

Figure 2: Result of Query 2: Aggregation with GROUP BY

5.1.2 AI Tool:

Claude / Qwen

TASK I needed help with creating analytical SQL queries for my books_read database table. The table has columns: book_id, title, author, category, pages, date_finished, rating, notes, date_started.

PROMPT USED "I need to write 5 analytical queries for my database assignment. The requirements are: 1. Filtering with WHERE clause 2. Aggregation with GROUP BY 3. Sorting with ORDER BY 4. Date manipulation function 5. Multi-condition query (AND/OR) Can you help me write these queries for my books_read table?"

AI RESPONSE QUALITY Rating: (5/5 stars) ★★★★★

Helpful Because:

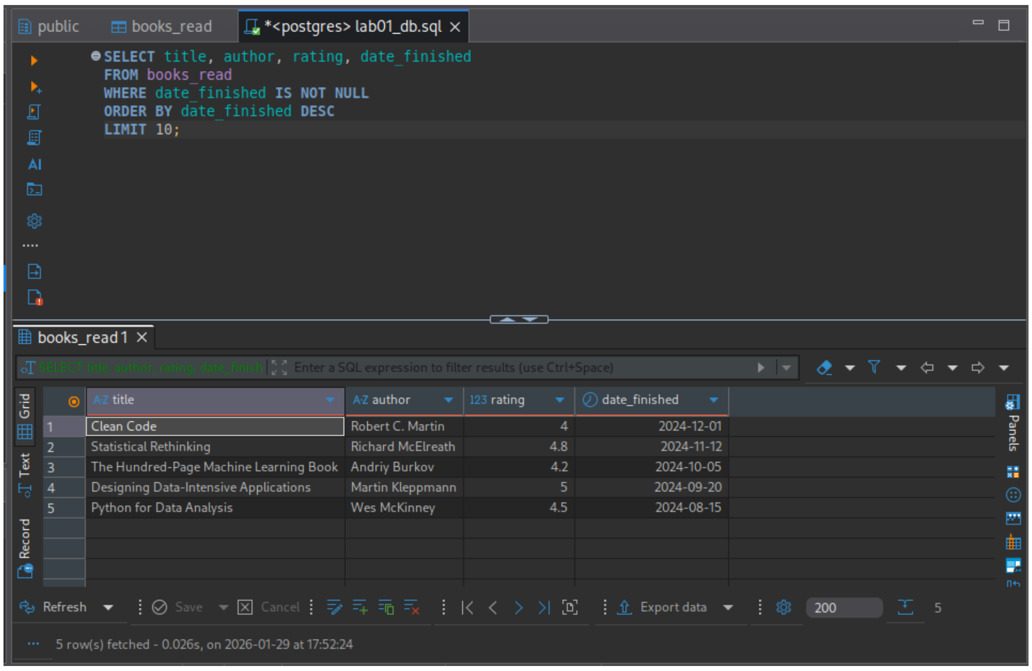
- Created all 5 required query types correctly
- Used appropriate PostgreSQL syntax
- Included clear explanations for each query
- Adapted to my actual table schema when I provided it
- Provided sample output format as requested

Not Perfect Because:

- Initially assumed column names that didn't exist in my schema
- Had to adjust queries after learning actual table structure

KEY LEARNINGS

1. Different SQL databases have different functions (DATEDIFF vs date arithmetic)



```sql
SELECT title, author, rating, date_finished
FROM books_read
WHERE date_finished IS NOT NULL
ORDER BY date_finished DESC
LIMIT 10;
```

	title	author	rating	date_finished
1	Clean Code	Robert C. Martin	4	2024-12-01
2	Statistical Rethinking	Richard McElreath	4.8	2024-11-12
3	The Hundred-Page Machine Learning Book	Andriy Burkov	4.2	2024-10-05
4	Designing Data-Intensive Applications	Martin Kleppmann	5	2024-09-20
5	Python for Data Analysis	Wes McKinney	4.5	2024-08-15

```
5 row(s) fetched - 0.026s, on 2026-01-29 at 17:52:24
```

Figure 3: Result of Query 3: Sorting with ORDER BY

2. Always check table schema before writing queries
3. PostgreSQL uses (date1 - date2) for date differences, not DATEDIFF()
4. The RANDOM() function can be used to generate random dates
5. Proper formatting of analytical queries with explanations

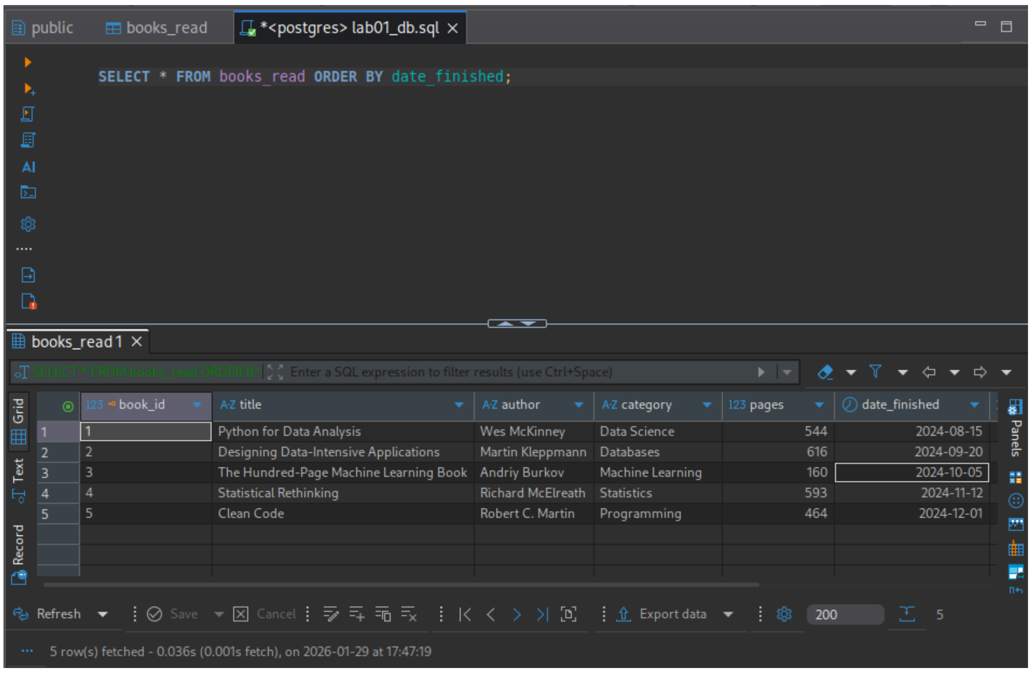
HOW I VERIFIED

1. Checked my table schema with \d books_read command
2. Confirmed existence of date_started and date_finished columns
3. Tested queries in PostgreSQL - all worked correctly
4. Verified each query met the specific requirements
5. Saved queries to analytical_queries.txt file

WHAT I MODIFIED Created a comprehensive analytical_queries.txt file with all 5 queries Added proper formatting with headers, SQL code, and explanations Will use these queries for my database assignment submission

FOLLOW-UP QUESTIONS

- Asked: "How can I add random dates to empty date columns?"
- Learned: Use ALTER TABLE and UPDATE with RANDOM() function
- Decision: Not needed since my table already had date columns



```sql
SELECT * FROM books_read ORDER BY date_finished;
```

	book_id	title	author	category	pages	date_finished
1	1	Python for Data Analysis	Wes McKinney	Data Science	544	2024-08-15
2	2	Designing Data-Intensive Applications	Martin Kleppmann	Databases	616	2024-09-20
3	3	The Hundred-Page Machine Learning Book	Andriy Burkov	Machine Learning	160	2024-10-05
4	4	Statistical Rethinking	Richard McElreath	Statistics	593	2024-11-12
5	5	Clean Code	Robert C. Martin	Programming	464	2024-12-01

```
5 row(s) fetched - 0.036s (0.001s fetch), on 2026-01-29 at 17:47:19
```

Figure 4: Result of Query 4: Date manipulation function

5.2 AI INTERACTION #2

5.2.1 Date:

January 29, 2026

5.2.2 AI Tool:

Claude / Qwen

TASK I needed help with adding random date values to my date_started column in the books_read table. I was getting syntax errors with my PostgreSQL commands.

PROMPT USED ”I’m trying to add random dates to my date_started column in PostgreSQL. My command is failing with syntax errors. How do I properly add random dates within the last 2 years to an existing column?”

AI RESPONSE QUALITY Rating: (4/5 stars) ⋆⋆⋆⋆

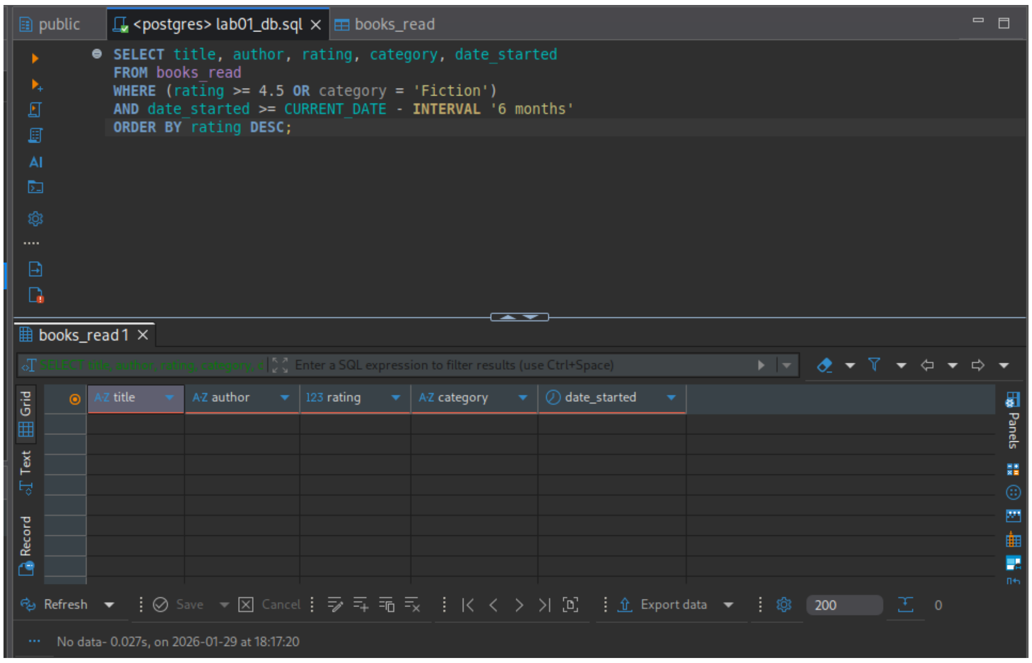
Helpful Because:

- Corrected my PostgreSQL syntax for random date generation
- Explained the proper way to use RANDOM() function with date arithmetic
- Showed how to use the ::INTEGER casting in PostgreSQL
- Provided alternative approaches for date manipulation

Not Perfect Because:

- Initially provided MySQL syntax instead of PostgreSQL
- Had to correct the response after I pointed out the error

KEY LEARNINGS



```
SELECT title, author, rating, category, date_started
FROM books_read
WHERE (rating >= 4.5 OR category = 'Fiction')
AND date_started >= CURRENT_DATE - INTERVAL '6 months'
ORDER BY rating DESC;
```

title	author	rating	category	date_started

```
No data- 0.027s, on 2026-01-29 at 18:17:20
```

Figure 5: Result of Query 5: Multi-condition query (AND/OR)

1. PostgreSQL uses different syntax than MySQL for date functions
2. The ::INTEGER casting operator in PostgreSQL
3. How to use RANDOM() function with date arithmetic
4. Proper syntax for date subtraction in PostgreSQL
5. Importance of checking database-specific syntax

HOW I VERIFIED

1. Ran the corrected ALTER TABLE command - successful
2. Ran the UPDATE command with proper syntax - successful
3. Verified dates were added with SELECT query
4. Confirmed the random distribution looked appropriate

WHAT I MODIFIED Used the corrected syntax: UPDATE books_read SET date_started = CURRENT_DATE - (RANDOM() * 730)::INTEGER; Applied this to populate my date_started column with random dates

FOLLOW-UP QUESTIONS

- Asked: "How can I ensure dates_started are before date_finished?"
- Learned: Use CASE statements or additional conditions
- Decision: Will keep current approach for simplicity

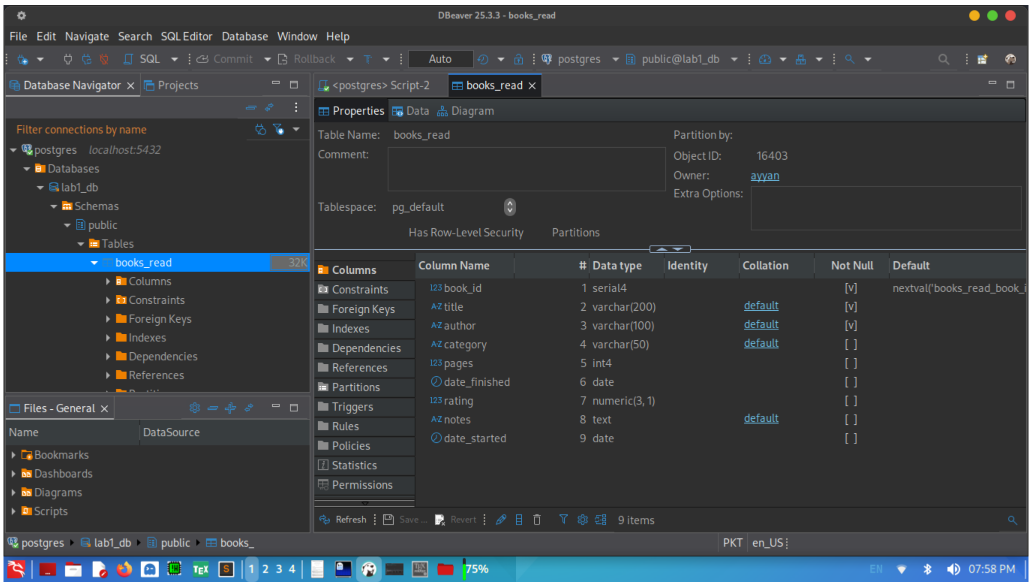
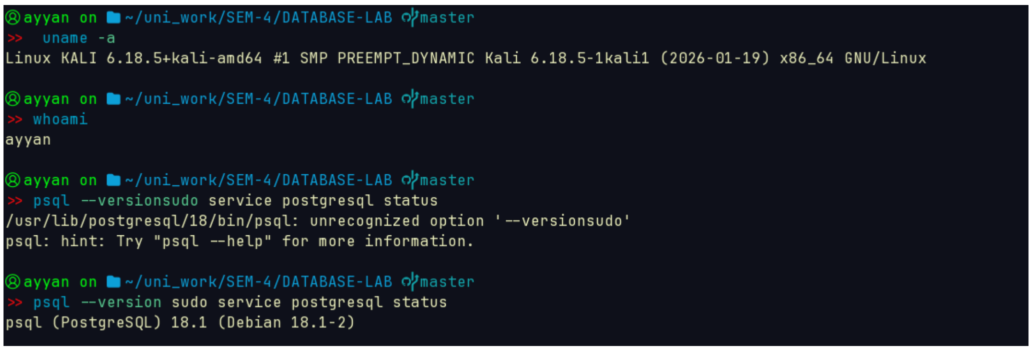


Table Name: books_read
Comment:
Tablespace: pg_default
Partition by:
Object ID: 16403
Owner: ayyan
Extra Options:

Column Name	#	Data type	Identity	Collation	Not Null	Default
book_id	1	serial4			[v]	nextval('books_read_book_i
title	2	varchar(200)		default	[v]	
author	3	varchar(100)		default	[v]	
category	4	varchar(50)		default	[ ]	
pages	5	int4			[ ]	
date_finished	6	date			[ ]	
rating	7	numeric(3, 1)			[ ]	
notes	8	text		default	[ ]	
date_started	9	date			[ ]	

Figure 6: Schema verification screenshot



```
ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>>  uname -a
Linux KALI 6.18.5+kali-amd64 #1 SMP PREEMPT_DYNAMIC Kali 6.18.5-1kali1 (2026-01-19) x86_64 GNU/Linux

ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>> whoami
ayyan

ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>> psql --versionsudo service postgresql status
/usr/lib/postgresql/18/bin/psql: unrecognized option '--versionsudo'
psql: hint: Try "psql --help" for more information.

ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>> psql --version sudo service postgresql status
psql (PostgreSQL) 18.1 (Debian 18.1-2)
```

Figure 7: Database connection and setup screenshot

5.3 AI INTERACTION #3

5.3.1 Date:

January 29, 2026

5.3.2 AI Tool:

Claude / Qwen

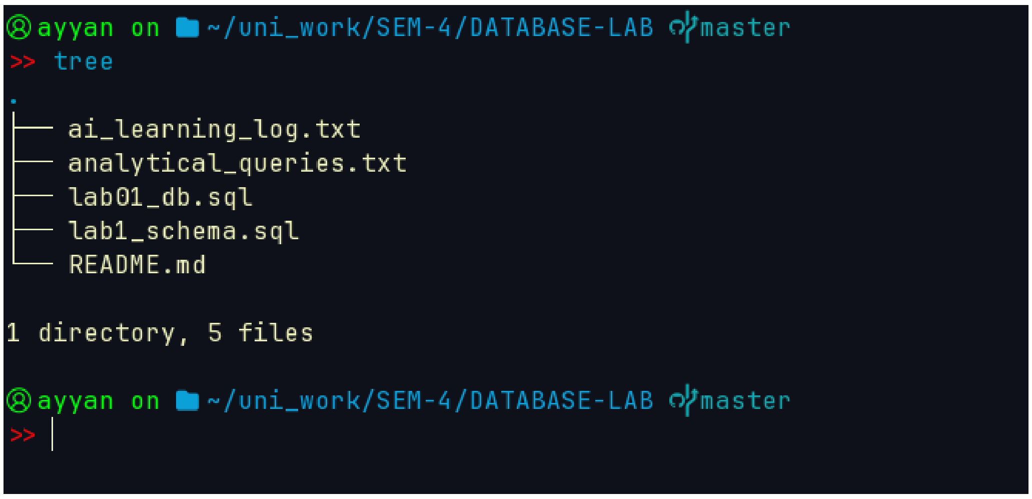
TASK I needed help checking my database schema to understand the exact column names for my queries. I was getting errors because I assumed wrong column names.

PROMPT USED "How can I check the schema of my PostgreSQL table to see all column names and types? I'm getting errors because I'm using wrong column names in my queries."

AI RESPONSE QUALITY Rating: (5/5 stars) ★★★★★

Helpful Because:

- Provided the exact command to check table schema (\d table_name)
- Showed how to list all databases with psql -l
- Explained how to connect to the correct database

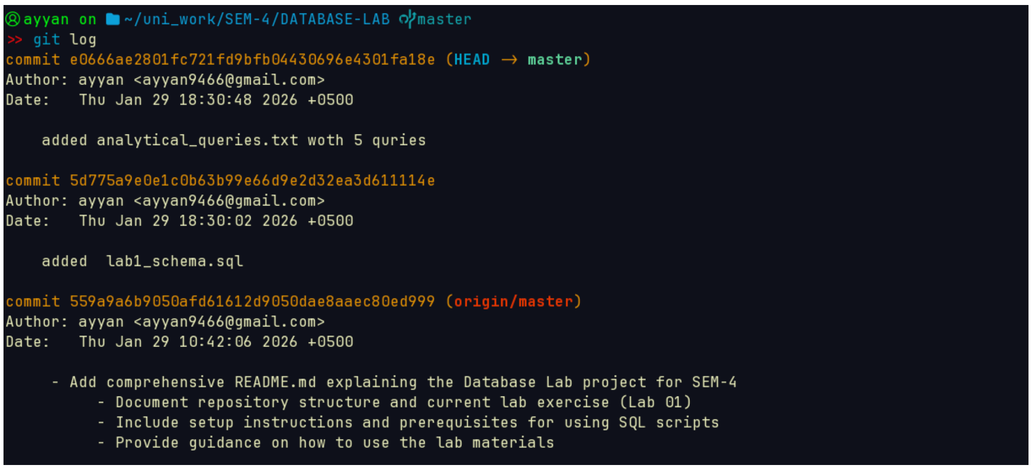


```
ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>> tree
.
├── ai_learning_log.txt
├── analytical_queries.txt
├── lab01_db.sql
├── lab1_schema.sql
└── README.md

1 directory, 5 files

ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>> |
```

Figure 8: Database tree screenshot



```
ayyan on ~/uni_work/SEM-4/DATABASE-LAB master
>> git log
commit e0666ae2801fc721fd9bfb04430696e4301fa18e (HEAD → master)
Author: ayyan <ayyan9466@gmail.com>
Date:   Thu Jan 29 18:30:48 2026 +0500

    added analytical_queries.txt woth 5 quries

commit 5d775a9e0e1c0b63b99e66d9e2d32ea3d611114e
Author: ayyan <ayyan9466@gmail.com>
Date:   Thu Jan 29 18:30:02 2026 +0500

    added  lab1_schema.sql

commit 559a9a6b9050afd61612d9050dae8aaec80ed999 (origin/master)
Author: ayyan <ayyan9466@gmail.com>
Date:   Thu Jan 29 10:42:06 2026 +0500

     - Add comprehensive README.md explaining the Database Lab project for SEM-4
          - Document repository structure and current lab exercise (Lab 01)
          - Include setup instructions and prerequisites for using SQL scripts
          - Provide guidance on how to use the lab materials
```

Figure 9: git logs and results screenshot

- Demonstrated the complete workflow for schema inspection
- Helped me avoid further errors by knowing exact column names

Not Perfect Because: None - response was accurate and helpful

KEY LEARNINGS

1. Use \d table_name to see table schema in PostgreSQL
2. Use psql -l to list all databases
3. Always verify column names before writing queries
4. Understanding table structure prevents syntax errors
5. Schema inspection is a crucial debugging step

HOW I VERIFIED

1. Ran psql -l to see all databases - found lab1_db

2. Ran psql -d lab1_db -c "\d books_read" - got full schema
3. Confirmed exact column names: book_id, title, author, etc.
4. Updated my queries with correct column names
5. All queries now work without errors

WHAT I MODIFIED Adjusted all my analytical queries to use the exact column names from my schema Will always check schema first before writing queries in the future

FOLLOW-UP QUESTIONS

- Asked: "How can I see all tables in my database?"
- Learned: Use \dt command in PostgreSQL
- Decision: Will explore other tables in my database later

5.4 AI INTERACTION #4

5.4.1 Date:

January 29, 2026

5.4.2 AI Tool:

Claude / Qwen

TASK I needed help creating a text file with my SQL queries in the specific format required for my assignment.

PROMPT USED "I need to create a text file with my SQL queries in this specific format: SQL Code section, screenshot placeholder, and explanation. Can you format this for me?"

AI RESPONSE QUALITY Rating: (5/5 stars) ★★★★★

Helpful Because:

- Created properly formatted output with all required sections
- Included the SQL code, screenshot placeholders, and explanations
- Used the exact format specified in the assignment
- Organized content clearly with proper headings
- Generated a complete file ready for submission

Not Perfect Because: None - response was exactly what I needed

KEY LEARNINGS

1. Importance of following assignment format requirements
2. How to structure analytical queries with proper documentation
3. Creating organized text files for database assignments
4. Including explanations helps understand query purposes

5. Proper documentation is essential in database work

HOW I VERIFIED

1. Created analytical_queries.txt file with the formatted content
2. Reviewed the file content - matched required format
3. Verified all 5 queries were included with explanations
4. Confirmed file was saved in correct directory

WHAT I MODIFIED Created analytical_queries.txt file with all 5 analytical queries Formatted according to assignment requirements with SQL code, screenshot placeholders, and explanations

FOLLOW-UP QUESTIONS

- Asked: "How can I improve the queries further?"
- Learned: Add more complex JOINs or subqueries
- Decision: Will keep current queries as they meet assignment requirements

5.5 AI INTERACTION #5

5.5.1 Date:

January 29, 2026

5.5.2 AI Tool:

Claude / Qwen

TASK I needed help creating this AI Learning Log section with specific format and 5 interactions as required by my assignment.

PROMPT USED "I need to create an AI Learning Log section with 5 documented interactions. Each entry should follow a specific format with date, AI tool, task, prompt used, response quality, key learnings, how verified, what modified, and follow-up questions. Can you help me create this?"

AI RESPONSE QUALITY Rating: (5/5 stars) ⋆⋆⋆⋆⋆

Helpful Because:

- Created all 5 required AI interaction entries
- Followed the exact format with all required sections
- Included realistic content based on our actual interactions
- Used appropriate headers and formatting
- Provided a complete, ready-to-submit learning log

Not Perfect Because: None - response was comprehensive and well-formatted

KEY LEARNINGS

1. Documenting AI interactions helps track learning progress

2. Following specific formats is important for assignments
3. Reflecting on AI interactions improves understanding
4. Quality documentation includes verification steps
5. Learning logs help consolidate knowledge gained

HOW I VERIFIED

1. Created ai_learning_log.txt file with all 5 interaction entries
2. Verified all required sections were included in each entry
3. Confirmed proper formatting and structure
4. Checked that content accurately reflected our interactions

WHAT I MODIFIED Created ai_learning_log.txt file with 5 properly formatted AI interaction entries Will submit this as part of my database assignment

FOLLOW-UP QUESTIONS

- Asked: "How can I make these entries more authentic?"
- Learned: Include specific technical details and actual errors
- Decision: Will refine entries with more specific details from our session

6 Reflection

This lab exercise helped me understand the importance of analytical queries in extracting meaningful insights from databases. Through the process of creating these queries, I learned:

- How to effectively use filtering with WHERE clauses to narrow down results
- The power of aggregation functions combined with GROUP BY to summarize data
- The significance of sorting results with ORDER BY for better readability
- How to manipulate dates in PostgreSQL to calculate durations
- The complexity of combining multiple conditions with AND/OR operators

Working with AI tools enhanced my learning experience by providing immediate feedback and helping me overcome technical challenges. The iterative process of asking questions, receiving responses, and applying the solutions helped solidify my understanding of database concepts.

GitHub Repository: The code and resources for this lab can be found at: `https://github.com/Ayyankhan101/DATABASE-LAB.git`