**Msc Computer science**

**Database Management and Analysis**

**Birmingham City University**

**Case Study: MSc Supervision Management System (MSMS)**

25105436

Table of Contents

[Critical Evaluation of Database Technologies for the HASS System 4](#_Toc197262612)

[Introduction 4](#_Toc197262613)

[Relational Databases 4](#_Toc197262614)

[Non-Relational (NoSQL) Databases 4](#_Toc197262615)

[Object-Oriented Databases 5](#_Toc197262616)

[Microsoft SQL Server 5](#_Toc197262617)

[Recommendation for HASS 5](#_Toc197262618)

[Database Design 7](#_Toc197262619)

[Relational Schema Mapping 8](#_Toc197262620)

[Assumptions Underpinning the EERD Design for HASS 8](#_Toc197262621)

[Database Development 10](#_Toc197262622)

[Tables 10](#_Toc197262623)

[Appointment Table 11](#_Toc197262624)

[Billing Table 11](#_Toc197262625)

[Diagnoses Table 12](#_Toc197262626)

[Doctor Table 12](#_Toc197262627)

[Feedback Table 13](#_Toc197262628)

[MedicalRecords Table 13](#_Toc197262629)

[Notification Table 14](#_Toc197262630)

[Patient Table 15](#_Toc197262631)

[Person Table 15](#_Toc197262632)

[Staff Table 17](#_Toc197262633)

[Test\_result Table 18](#_Toc197262634)

[Userrole Table 18](#_Toc197262635)

[Userrolenotification Table 19](#_Toc197262636)

[Interactive Queries 19](#_Toc197262637)

[Security Scenario 25](#_Toc197262638)

[Security Scenario 1 – Administrator Role 25](#_Toc197262639)

[Security Scenario 2 – Receptionist Role 25](#_Toc197262640)

[Security Scenario 3 - Doctor Role 26](#_Toc197262641)

[Security Scenario 4 – Pharmacist Role 26](#_Toc197262642)

[Security Scenario 5 – Billing Officer Role 27](#_Toc197262643)

[Security Scenario 6 – Lab Technician Role 27](#_Toc197262644)

[Conclusion 28](#_Toc197262645)

[References 28](#_Toc197262646)

# Critical Evaluation of Database Technologies for the HASS System

# Introduction

Choosing the right database technology plays a central role in the performance, reliability, and security of any digital system. For healthcare applications like the Healthcare Appointment Scheduling System (HASS), which manages confidential patient records, appointment data, and administrative transactions, the decision is particularly critical. This section examines a range of database models including relational, non-relational, and object-oriented systems before narrowing the evaluation to Microsoft SQL Server and ultimately recommending the most suitable choice for the HASS project.

# Relational Databases

Relational databases have long formed the foundation of many software systems due to their structured approach to data management. Built on the relational model first described by Codd in 1970, these systems store data in well-defined tables linked by keys and queried using SQL (Codd, 1970; Elmasri & Navathe, 2017). Common implementations include MySQL, PostgreSQL, Oracle Database, and Microsoft SQL Server.

MySQL is widely appreciated for being open-source, lightweight, and compatible with platforms like XAMPP and PHP (Widenius & Axmark, 2002). PostgreSQL provides advanced features such as user-defined types and full-text search, making it a strong contender for complex applications (Stonebraker & Rowe, 2010). Oracle Database, while enterprise-class in performance and security, often exceeds the resource and cost requirements of smaller systems (Oracle, 2020). Microsoft SQL Server integrates smoothly with the Windows ecosystem and is known for its powerful administrative and analytical tools (Coronel & Morris, 2015).

Relational systems offer notable benefits such as strong data integrity controls, comprehensive query capabilities, and a wide support community (Date, 2003). However, their strict schema requirements can hinder development agility, and they often fall short when handling semi-structured or rapidly evolving datasets (Moniruzzaman & Hossain, 2013). In the context of HASS, MySQL’s open-source model, simplicity, and proven reliability make it a fitting candidate.

# Non-Relational (NoSQL) Databases

Non-relational databases have gained popularity in the era of big data and flexible data models. Unlike their relational counterparts, NoSQL databases are schema-less, enabling them to store a wide variety of data types including documents, key-value pairs, wide-column data, and graphs (Han et al., 2011).

Popular NoSQL systems include MongoDB (document-based), Redis (key-value), Cassandra (column-family), and Neo4j (graph) (Chodorow, 2013; Lakshman & Malik, 2010; Robinson et al., 2015). These systems excel in horizontal scalability and fast data access, especially in distributed or high-throughput environments.

On the downside, many NoSQL databases trade strict consistency for performance and flexibility, adopting a BASE model over ACID guarantees (Pritchett, 2008). They often lack a standard querying language, and their security features can be inconsistent across platforms (Moniruzzaman & Hossain, 2013). Given that HASS deals primarily with structured, relational data in a moderate-scale environment, NoSQL databases may offer more complexity than benefit.

# Object-Oriented Databases

Object-oriented databases (OODBMS) aim to unify the programming and data management layers by storing objects as they exist in the application code. Systems such as ObjectDB and db4o are examples of databases that support features like inheritance and encapsulation within the data store (Atkinson et al., 1989; ObjectDB, 2021).

These databases are ideal for applications involving complex data relationships, such as engineering simulations or multimedia management (Mokrani, 2015). Their main appeal lies in the elimination of object-relational mapping and the close alignment between the application logic and the data model.

Despite these advantages, object-oriented databases are rarely used in mainstream enterprise systems. They suffer from poor tool support, limited standardization, and low compatibility with common reporting and analytics tools (Versant, 2008). For HASS, which requires relational processing and SQL integration, object databases are not a practical fit.

# Microsoft SQL Server

Microsoft SQL Server combines the robustness of relational models with advanced features for business analytics, indexing, and security. It is especially effective in organizations already using Microsoft tools and environments, offering seamless integration with the .NET framework and Microsoft Excel (Vacca, 2014).

SQL Server’s graphical user interface and built-in functions make administration more intuitive than command-line-based systems. It also provides fine-grained access controls and backup tools suited for sensitive environments such as healthcare (Coronel & Morris, 2015).

However, these features come at a cost. The licensing fees can be substantial for small projects, and the system requires more memory and processing resources compared to lighter alternatives like MySQL (TechRepublic, 2018). For a lightweight, open-source solution like HASS, SQL Server’s complexity may be unnecessary.



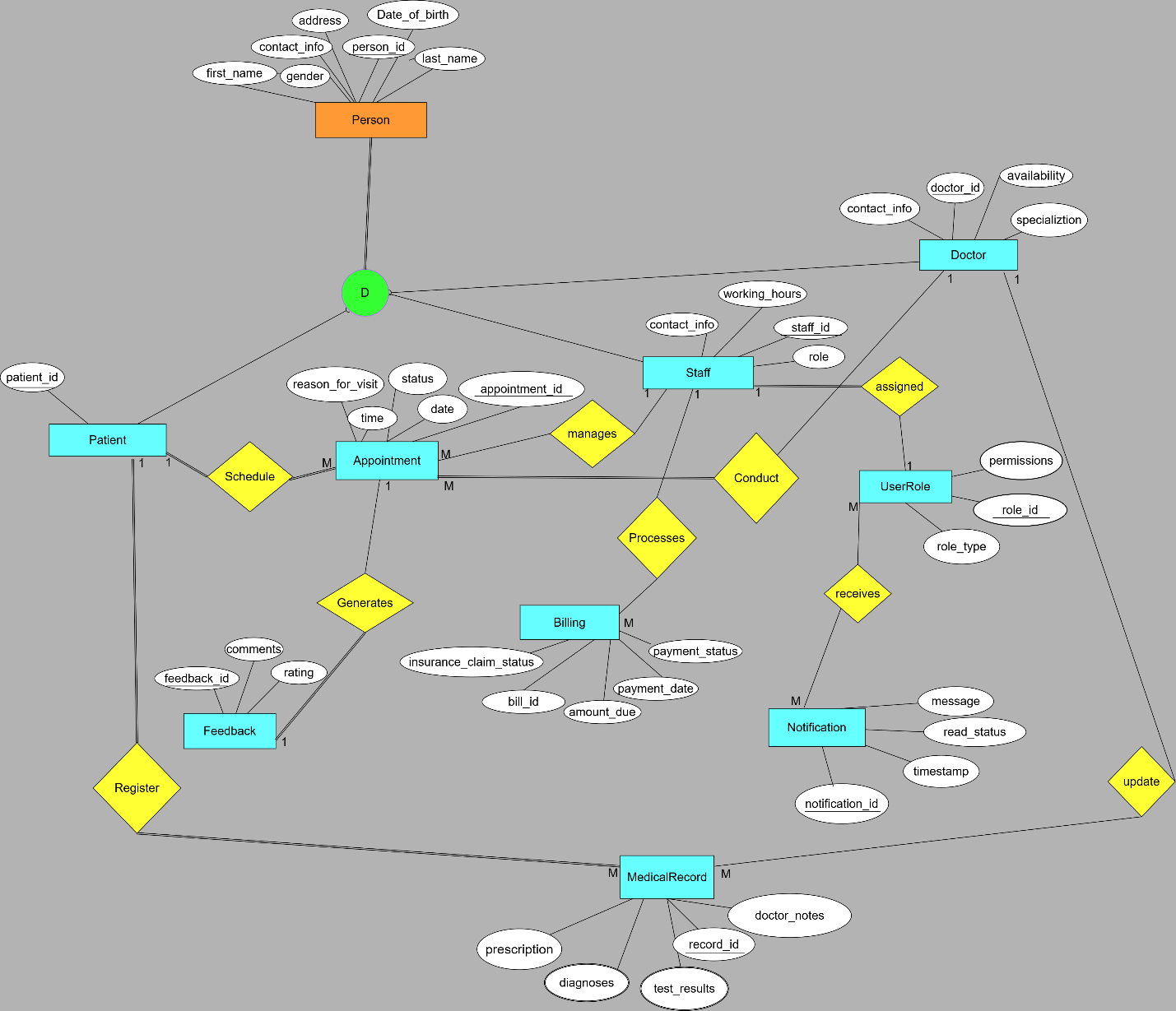
# Recommendation for HASS

Taking all factors into account, MySQL is the most practical and effective choice for the HASS system. It aligns well with the project’s use of XAMPP, supports relational logic, and requires minimal resources while still offering reliability and strong SQL functionality (Widenius & Axmark, 2002). NoSQL and object-oriented databases bring strengths in flexibility and modeling but exceed the needs of a moderately sized healthcare scheduler. Although SQL Server is feature-rich, its administrative overhead and licensing requirements reduce its suitability for this specific deployment.

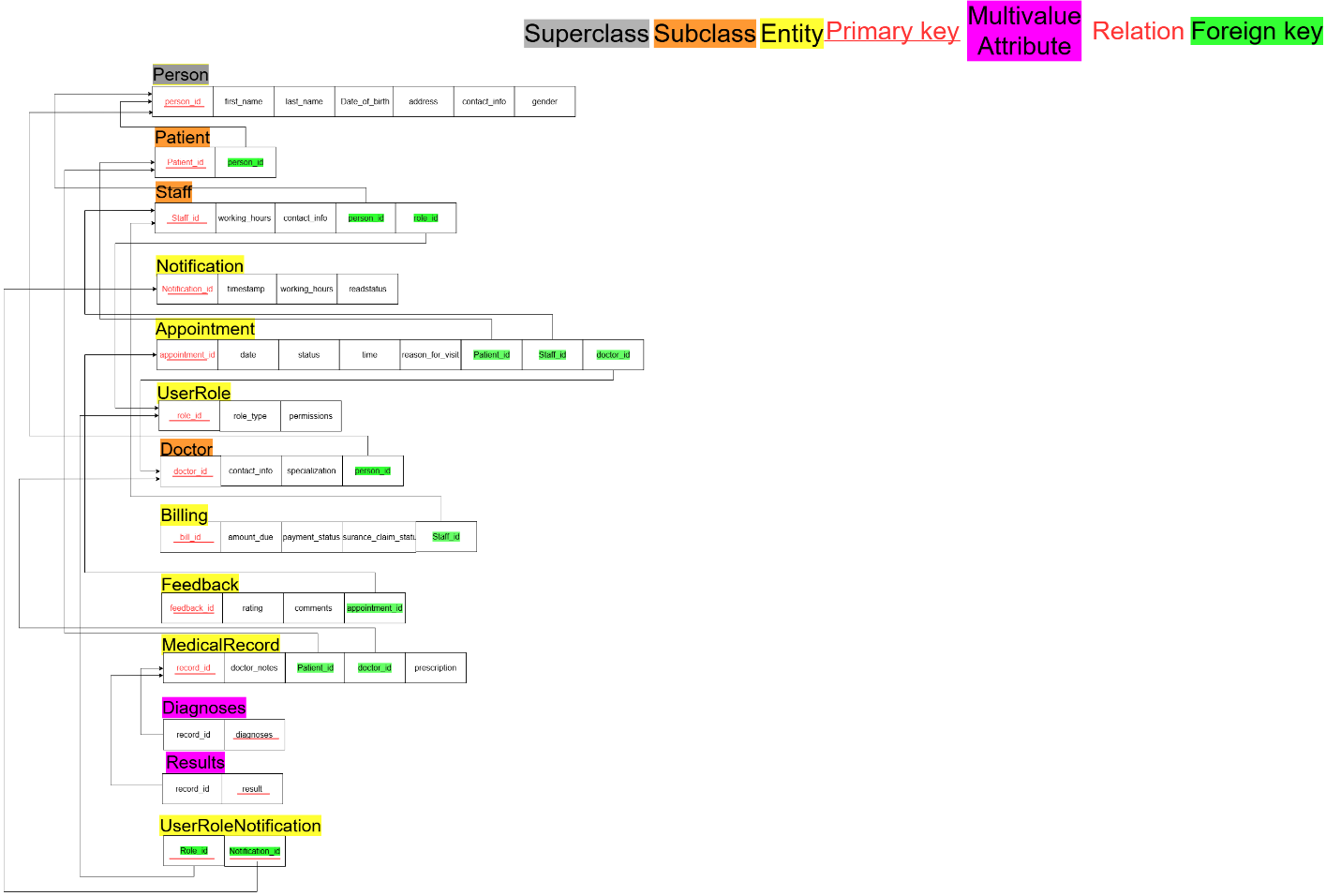
This tailored evaluation ensures that the selected DBMS not only meets technical requirements but also considers the practical realities of project scale, available resources, and expected growth.

# Database Design

Enhanced Entity Relational Diagram



# Relational Schema Mapping



# Assumptions Underpinning the EERD Design for HASS

In crafting the Enhanced Entity-Relationship Diagram (EERD) for the Healthcare Appointment Scheduling System (HASS), several foundational assumptions were adopted to translate the case study into a practical and logical data model. These assumptions serve to enhance clarity, reduce redundancy, and ensure the system’s architecture aligns with real-world healthcare workflows.

Firstly, a supertype entity named "Person" was introduced to encompass common characteristics shared by all individuals in the system, including patients and staff members. Attributes such as first name, last name, date of birth, contact information, gender, and residential address are stored within this central entity to promote normalization and avoid unnecessary duplication.

Secondly, the model treats "Doctor" as a subtype of "Staff," recognizing that doctors possess additional specialized attributes such as medical focus or scheduling availability which in turn inherits from Person, maintaining a logical hierarchy.

It is also presumed that each staff member holds only one active system role at a time—be it receptionist, nurse, or administrator. The case study did not suggest scenarios where multiple roles are simultaneously assigned to a single staff member.

The "Appointment" entity functions as the operational hub of the system, linking together key components like patients, staff, doctors, billing entries, and feedback. It represents the key transactional unit within the application.

In line with the system description, each appointment is expected to generate a single, unique billing record. This assumption is based on the statement that each patient visit results in a billing event.

Feedback from patients is assumed to follow a one-to-one relationship with appointments. That is, every appointment can receive exactly one feedback entry, aligning with the use case that feedback is tied directly to the visit.

The MedicalRecord entity accommodates multi-valued information such as diagnoses, prescriptions, and test results. These attributes are either normalized into associated entities or managed through supported structures based on the selected DBMS.

Notifications are considered user-specific and are assumed to be sent to individual users based on their involvement in events within the system. Whether the recipient is a doctor, patient, or support staff, the notification is tailored to the user's role and the action that triggered it.

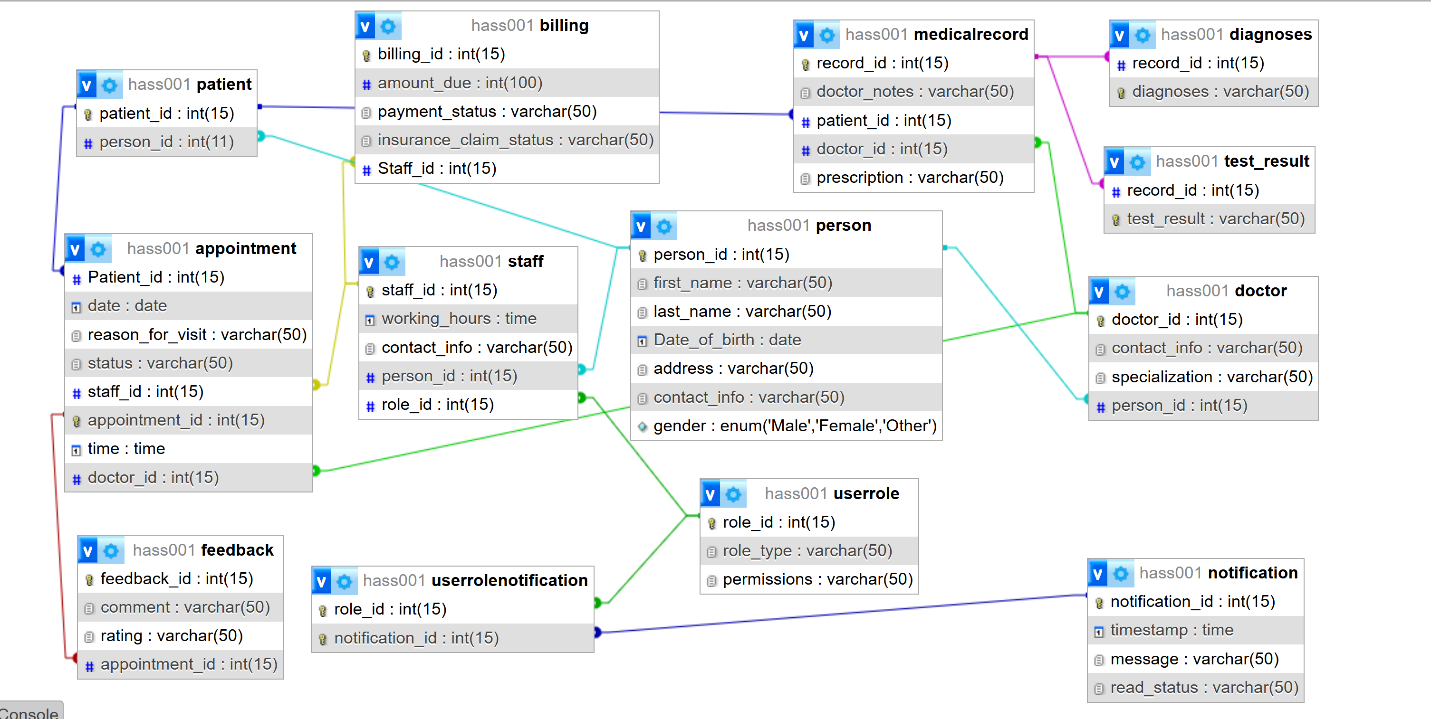
It is further assumed that various non-medical staff, such as receptionists, may contribute to appointment coordination, assist with patient records, or initiate billing processes. These activities may be logged for system auditing and operational transparency.

Lastly, timestamp fields are presumed to be present in entities like appointments, billing, and notifications. This allows the system to maintain accurate records of time-sensitive actions for reporting and accountability purposes.

These assumptions collectively ensure that the resulting EERD is grounded in practical, scalable, and maintainable database principles suitable for managing operations within a healthcare appointment management system.

# Database Development

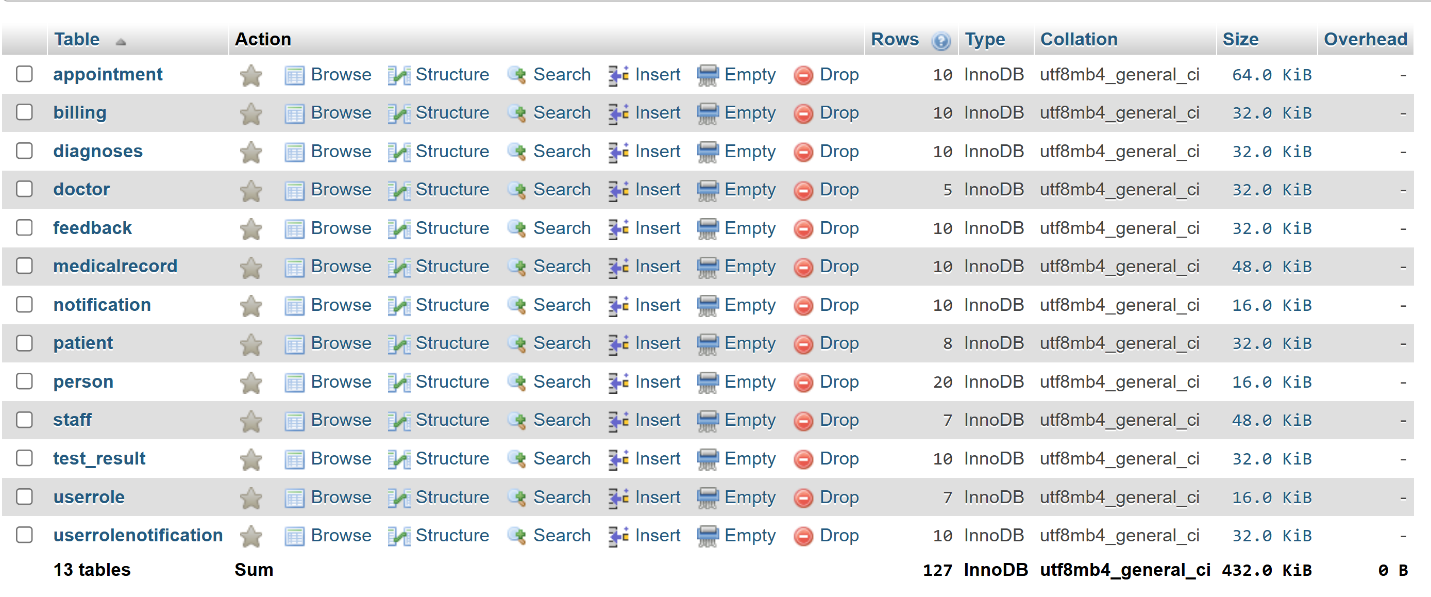
Database Designer Interface



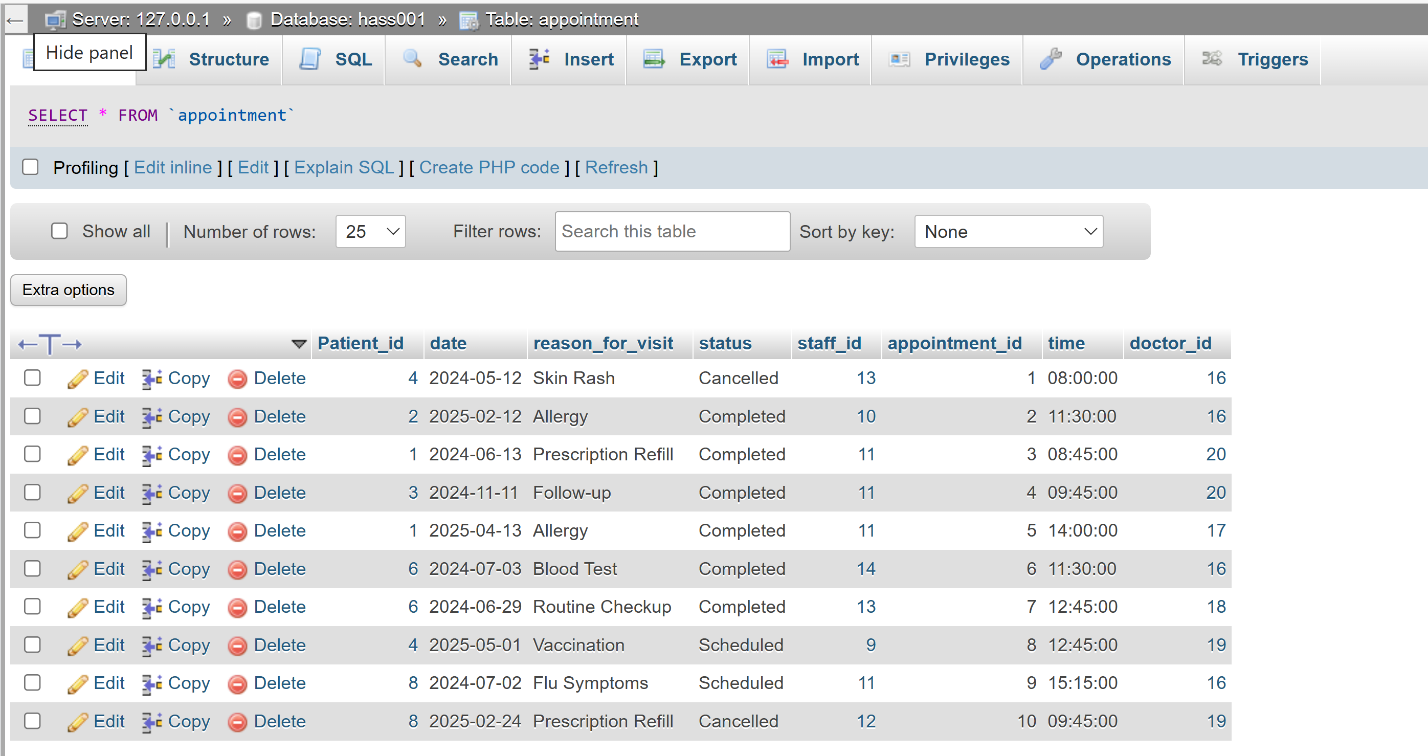
# Tables

For Hass database populated with at least 10 data

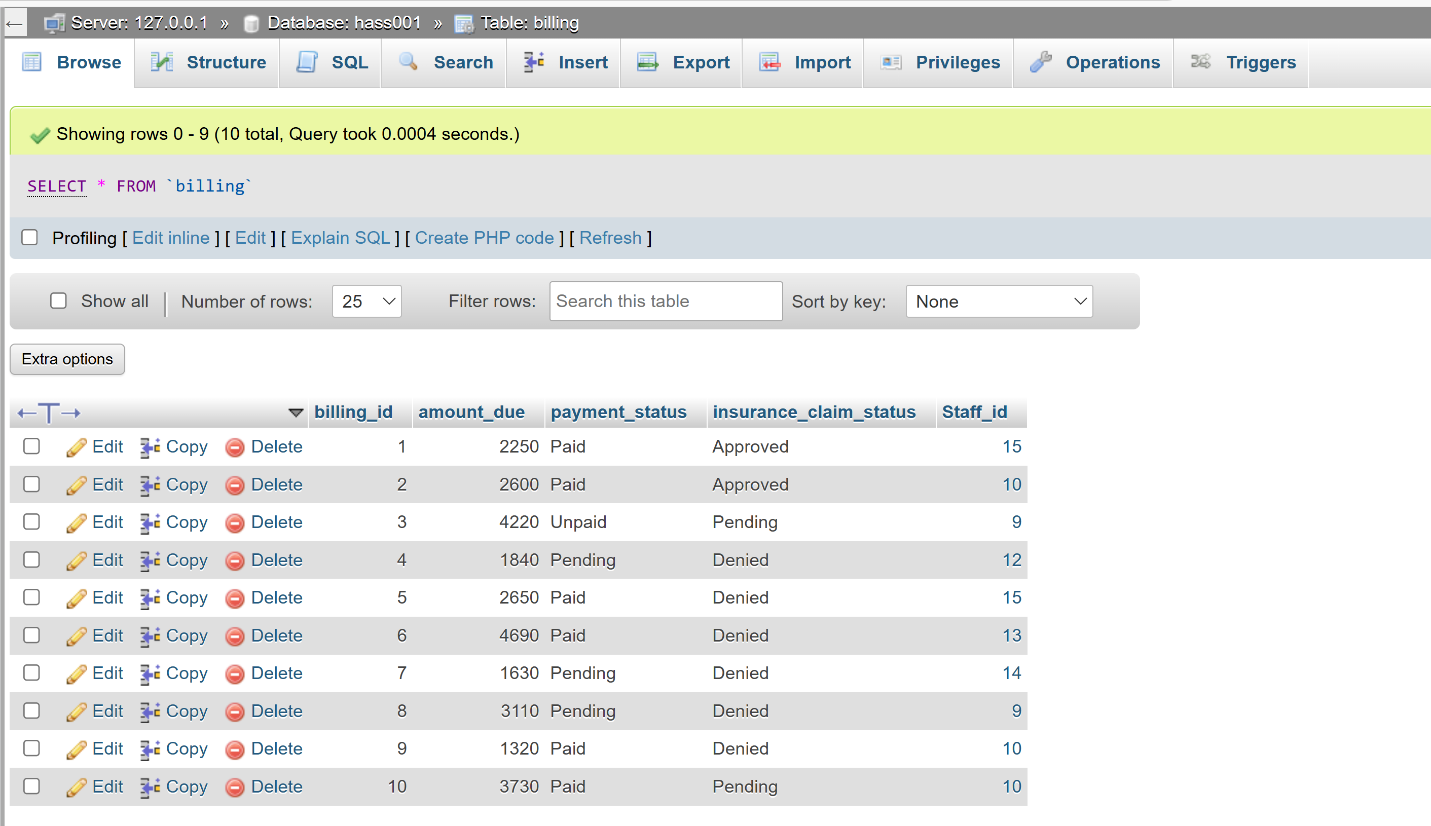
List of all tables of HASS databases



## Appointment Table



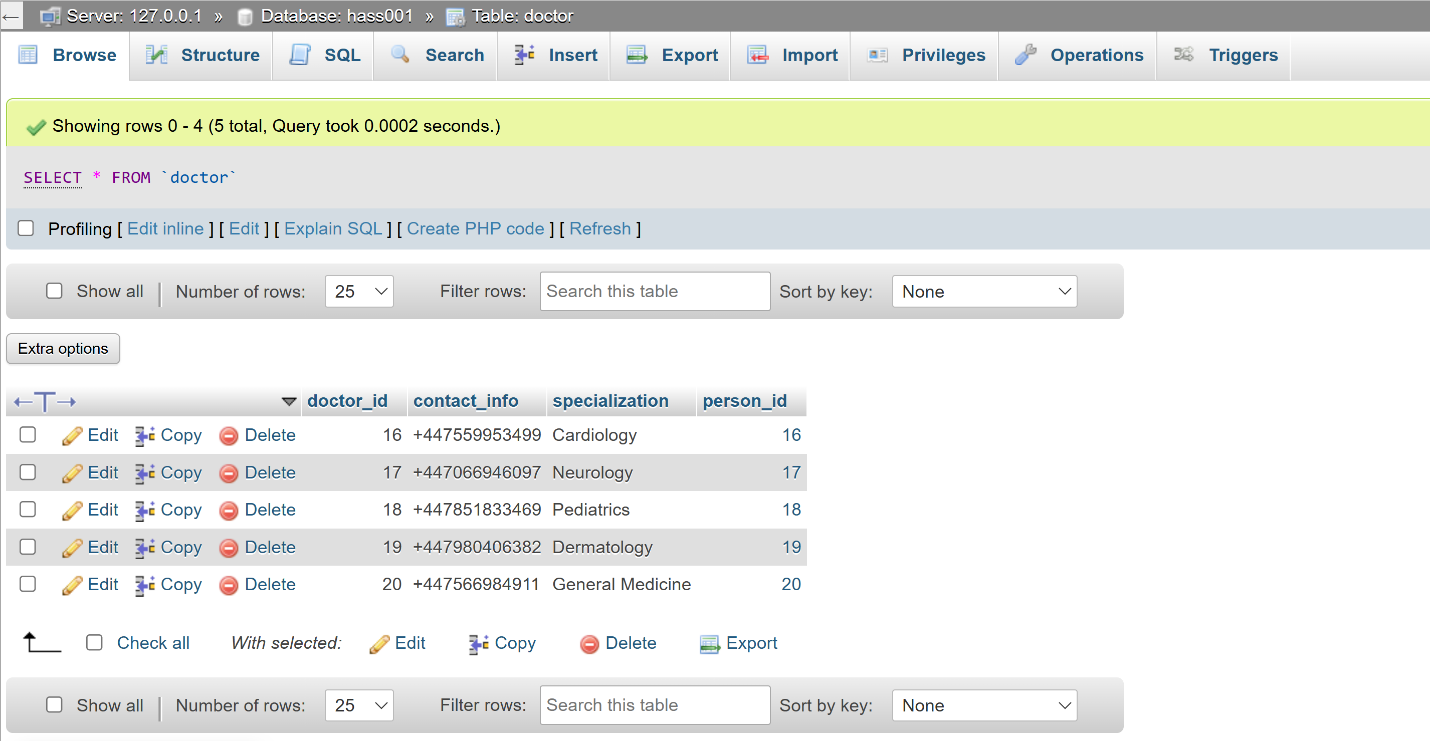
## Billing Table



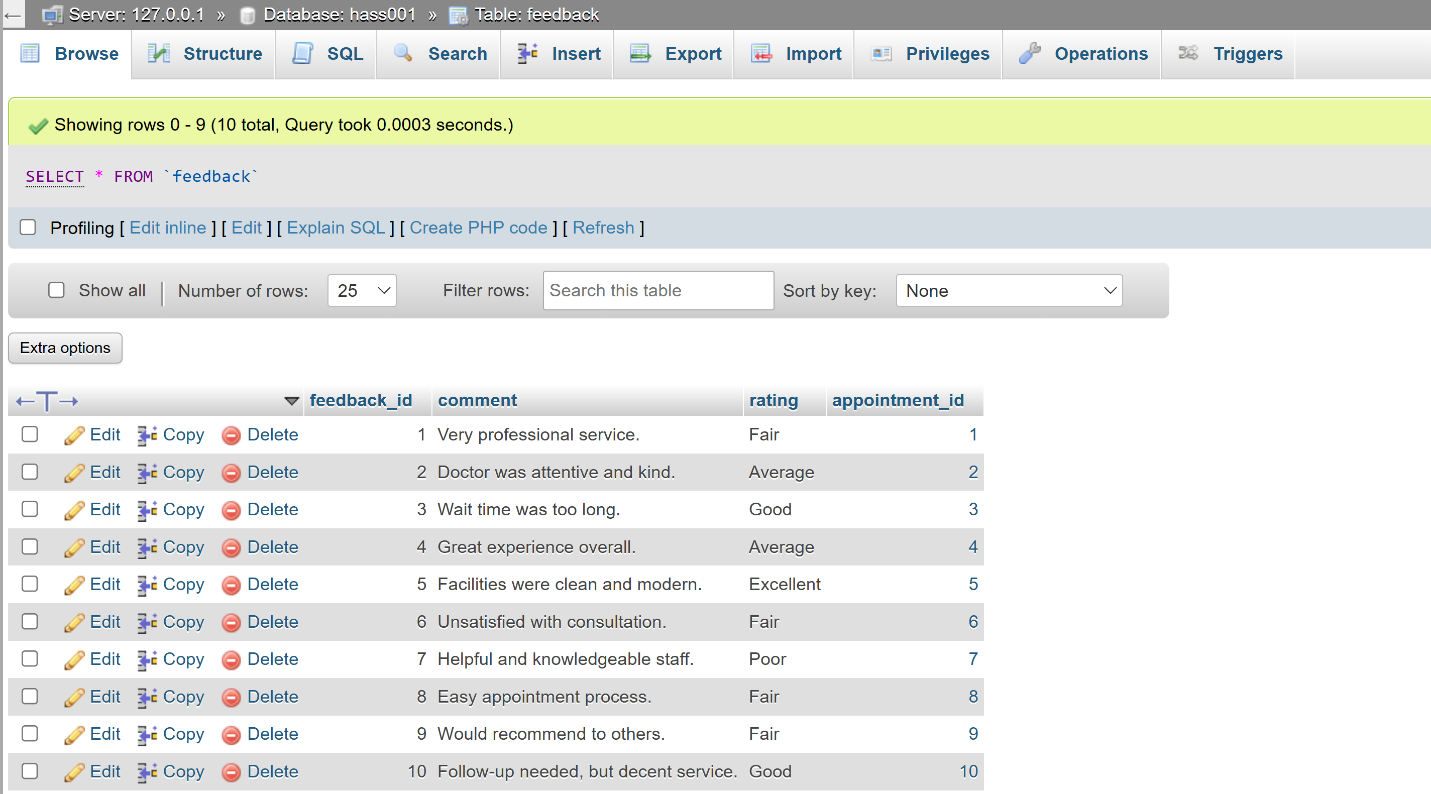
## Diagnoses Table



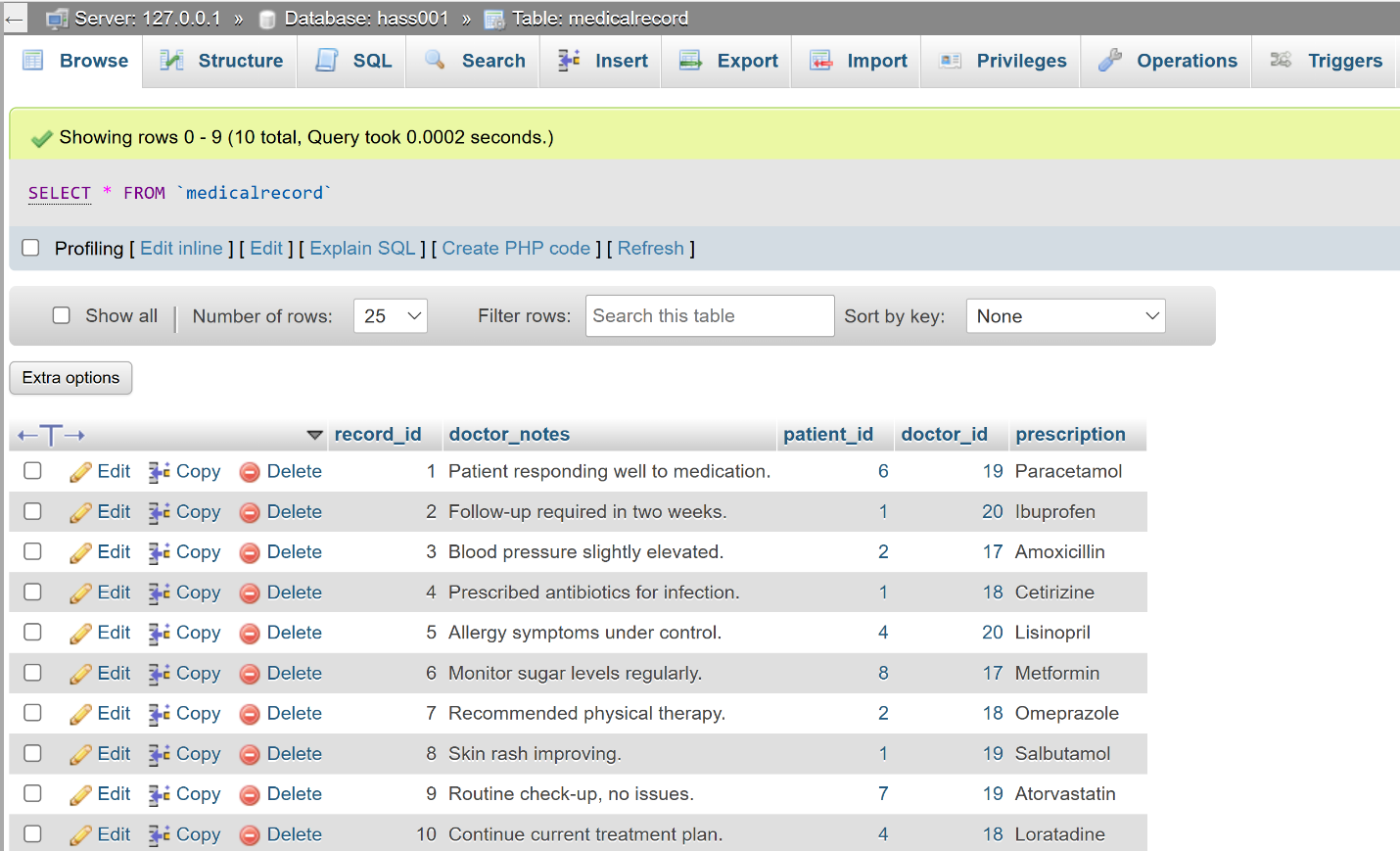
## Doctor Table



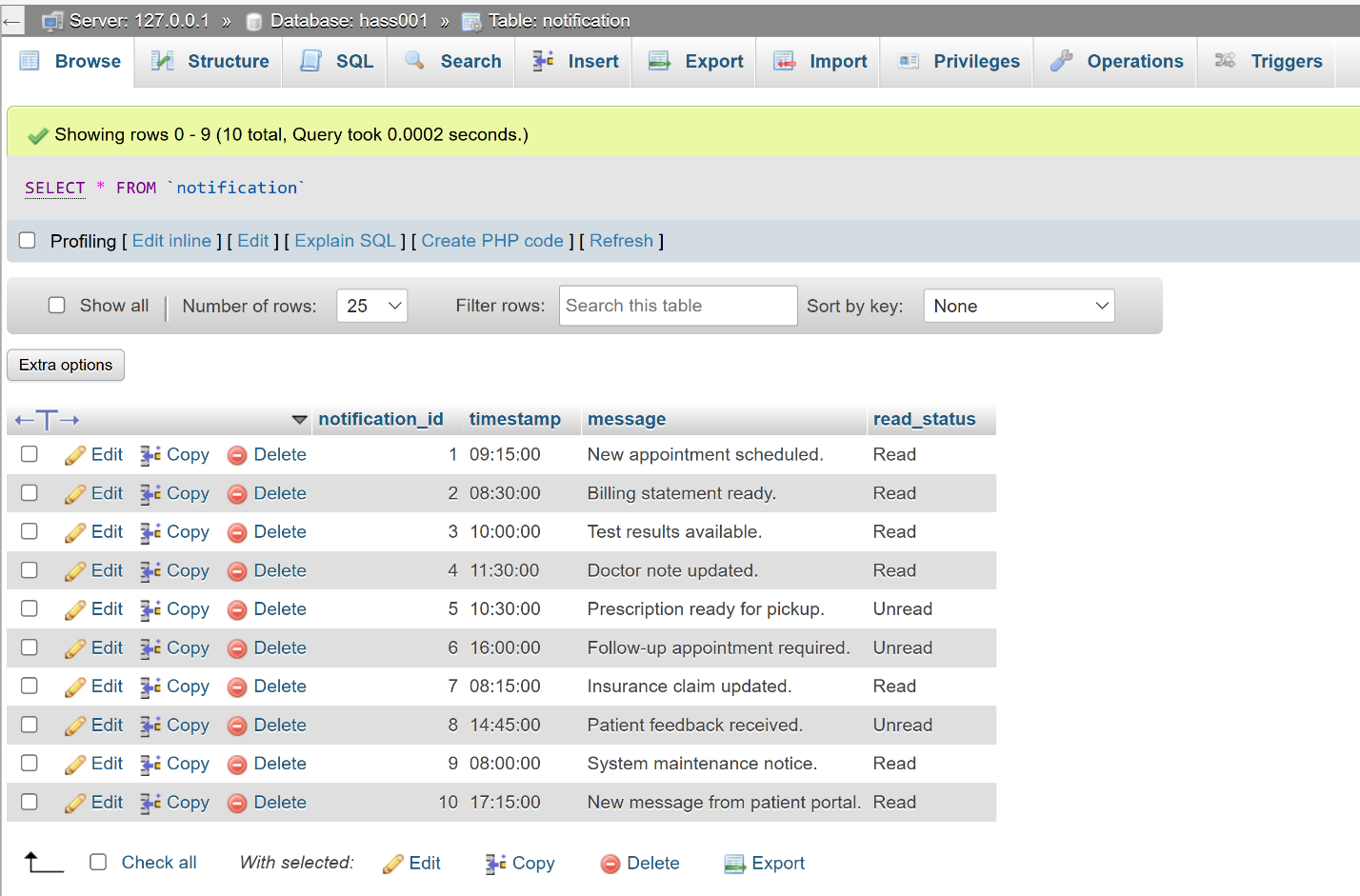
## Feedback Table



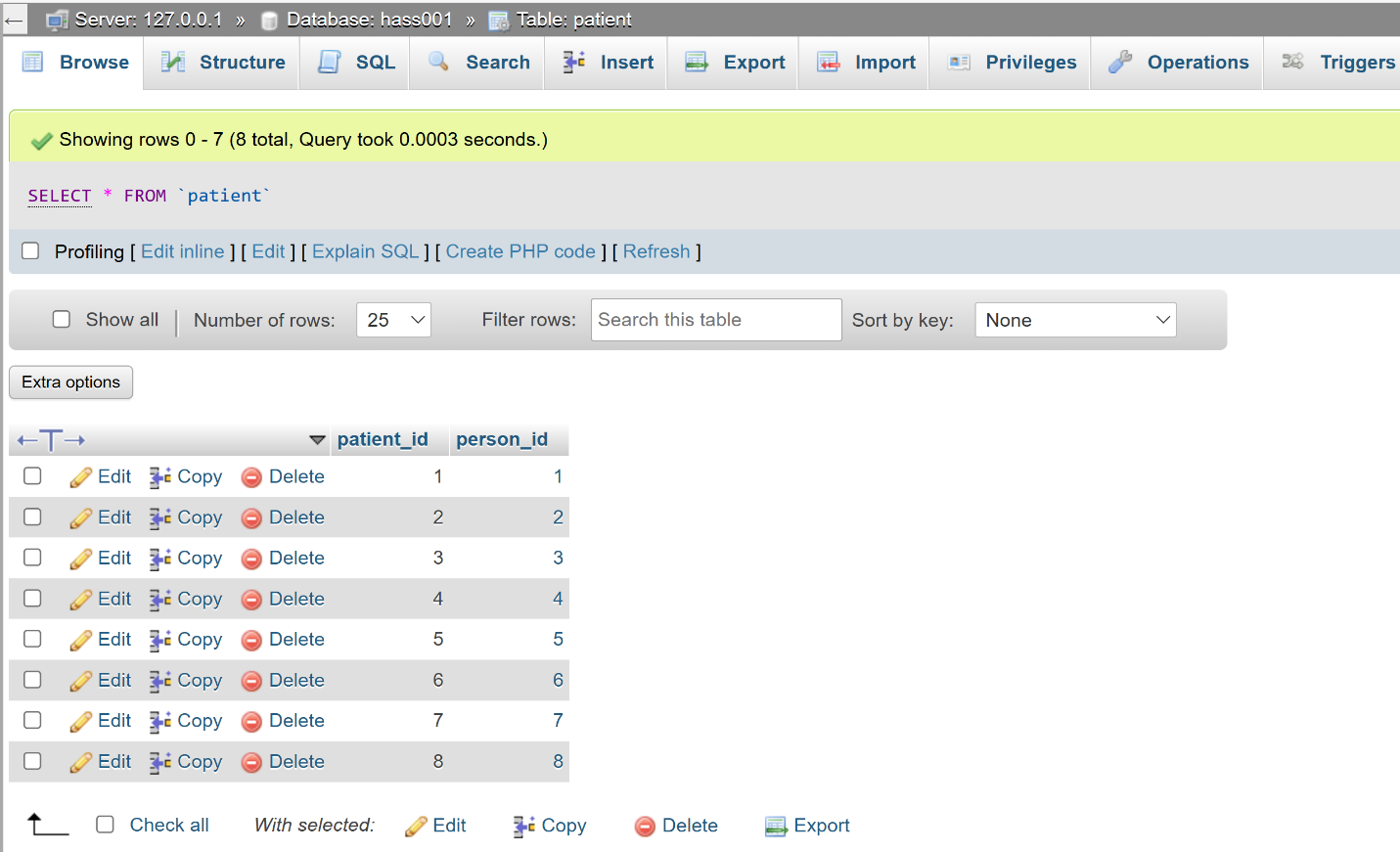
## MedicalRecords Table



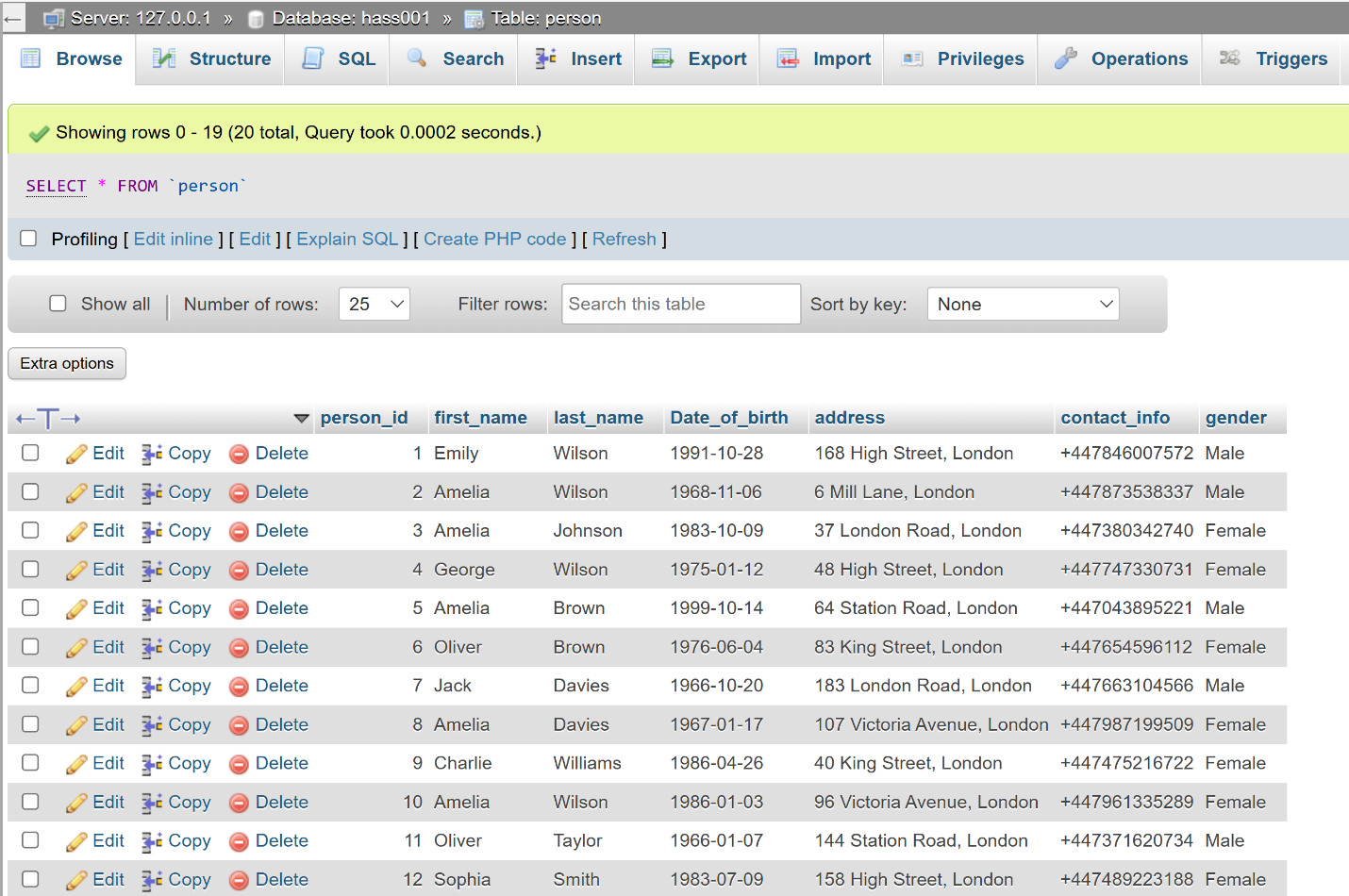
## Notification Table

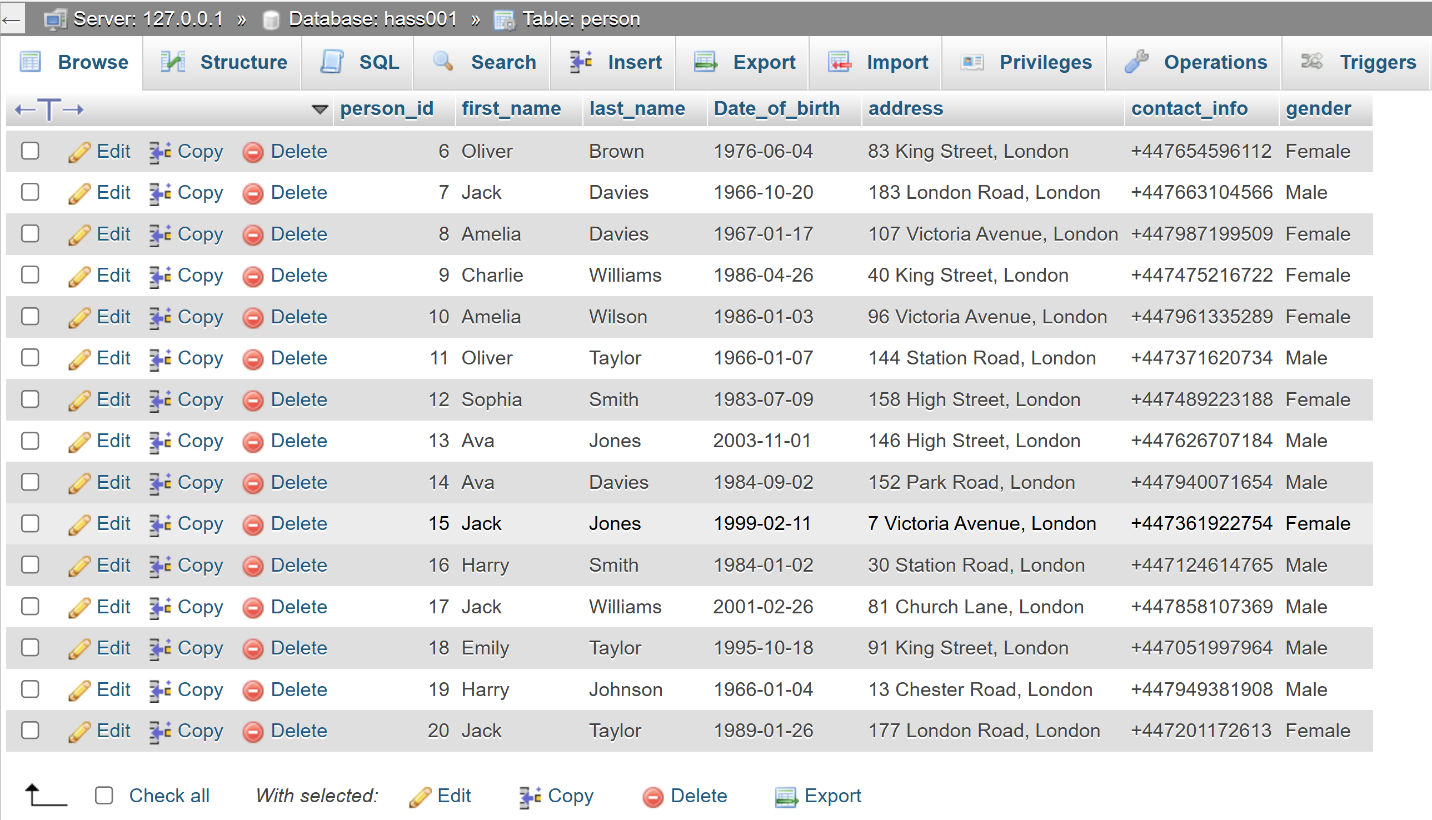


## Patient Table

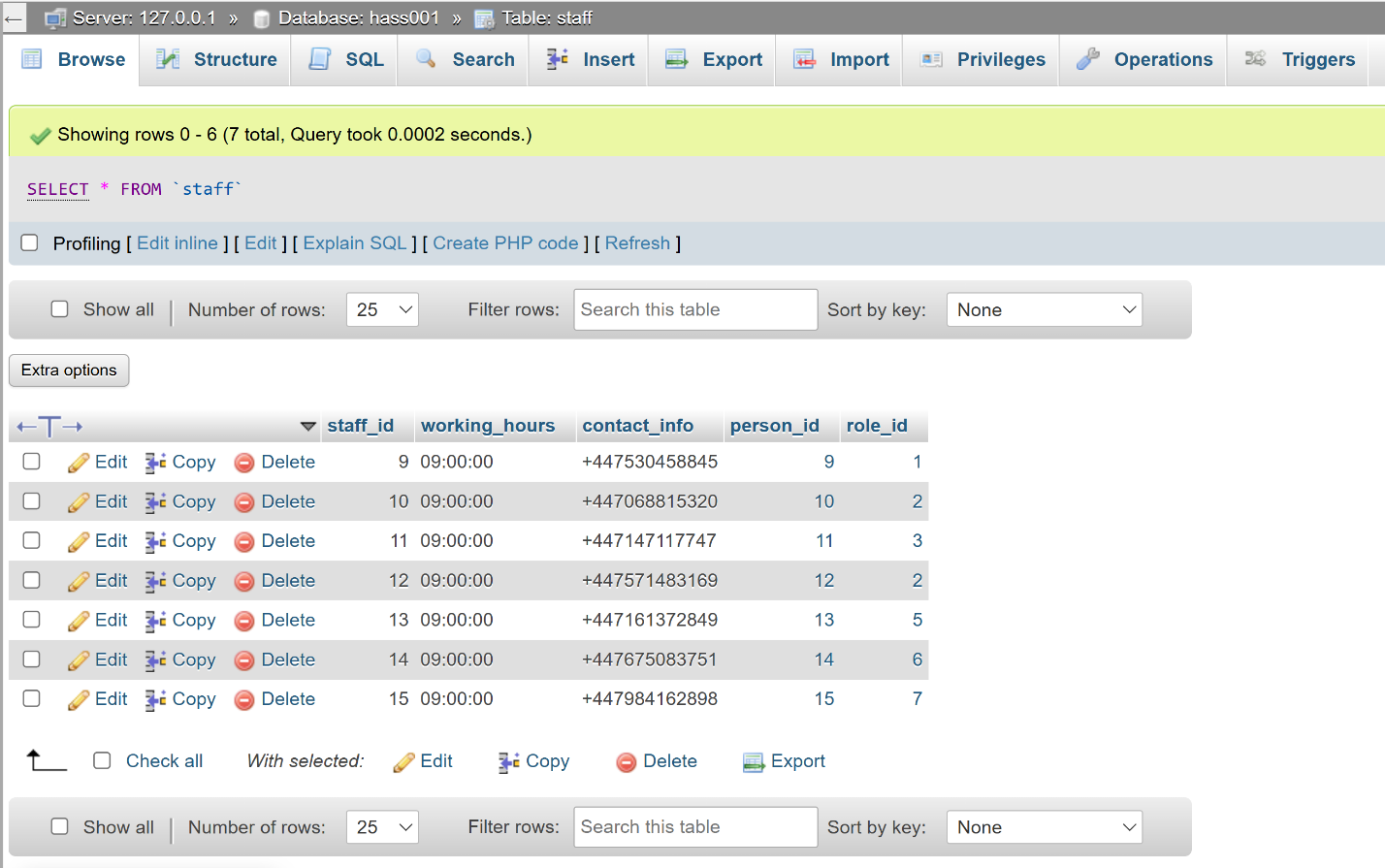


## Person Table

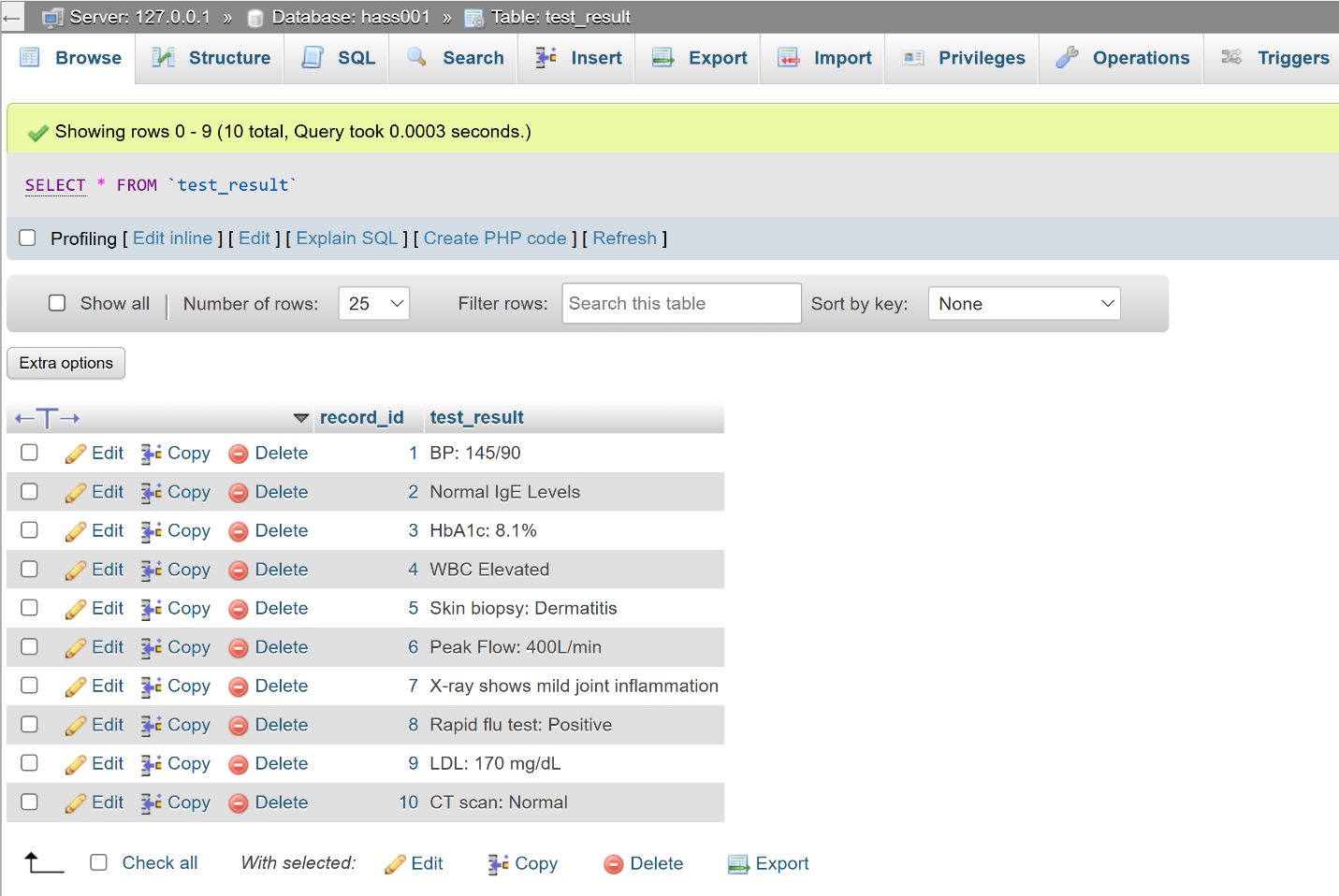




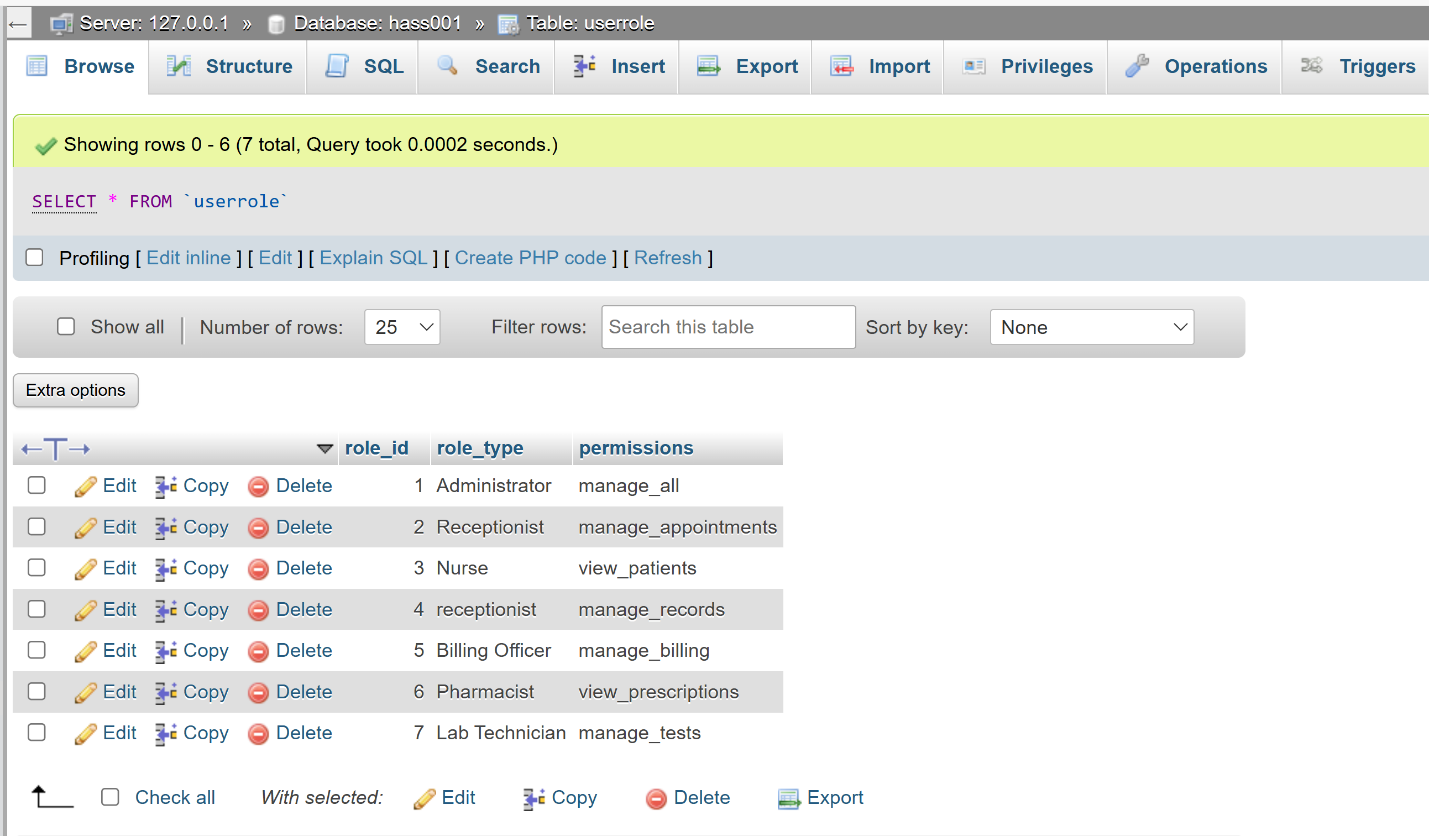
## Staff Table



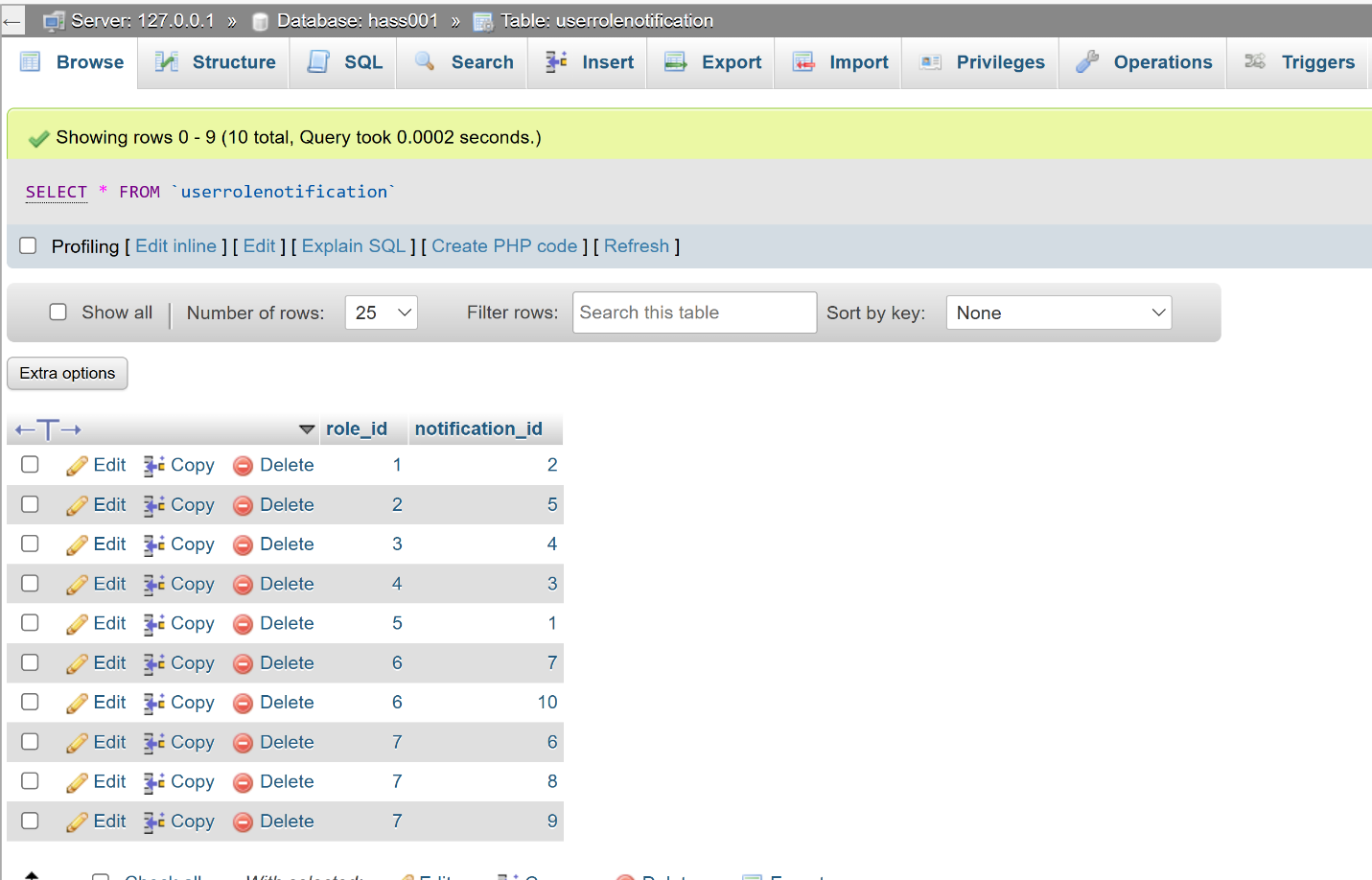
## Test\_result Table



## Userrole Table

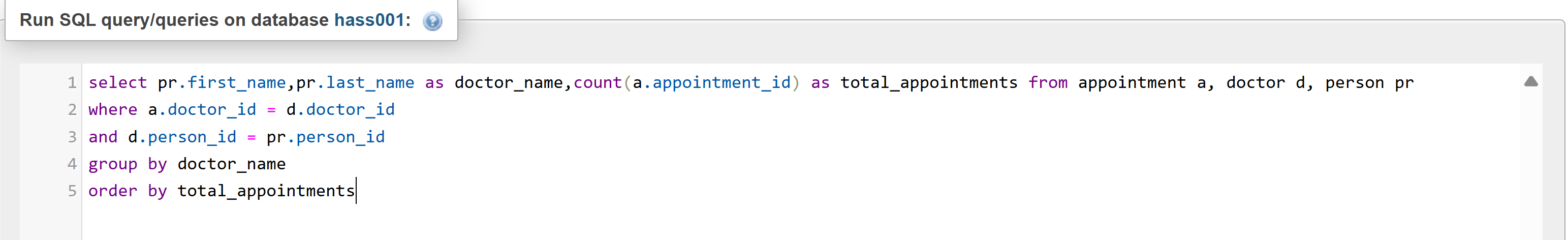


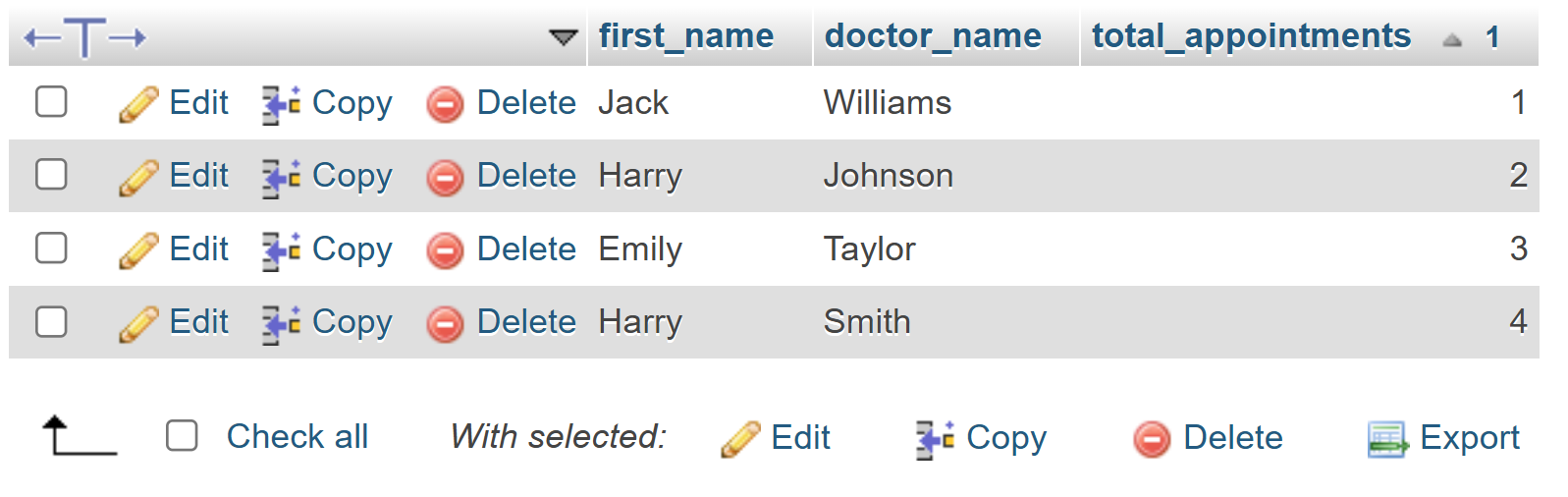
## Userrolenotification Table



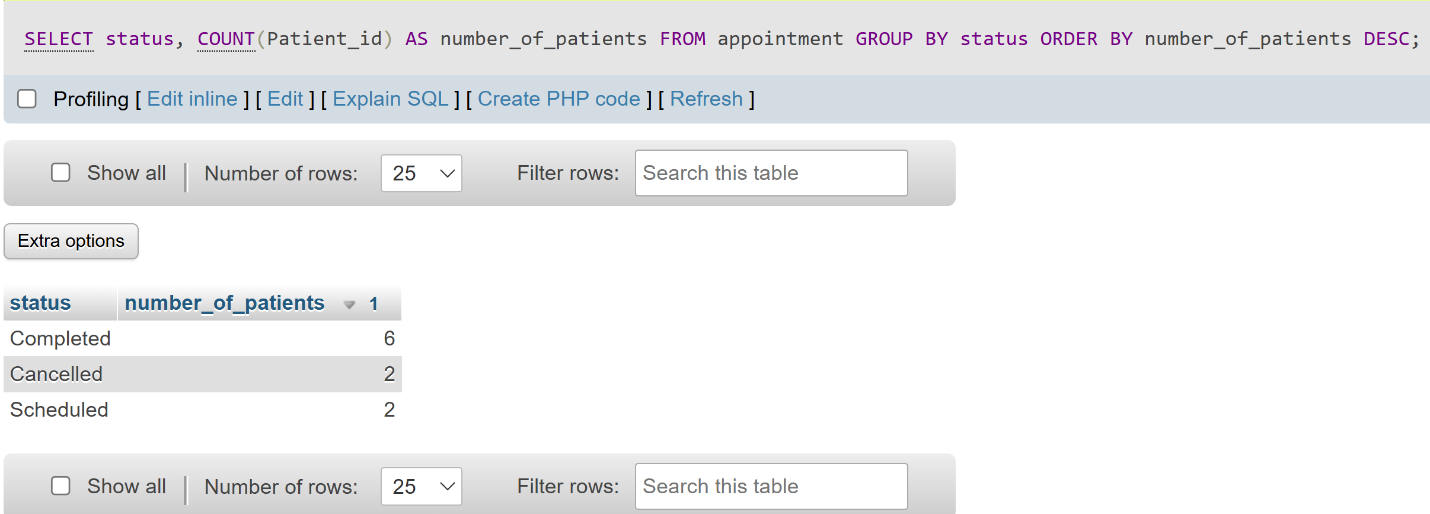
# Interactive Queries

1. How many appointments has each doctor conducted ?





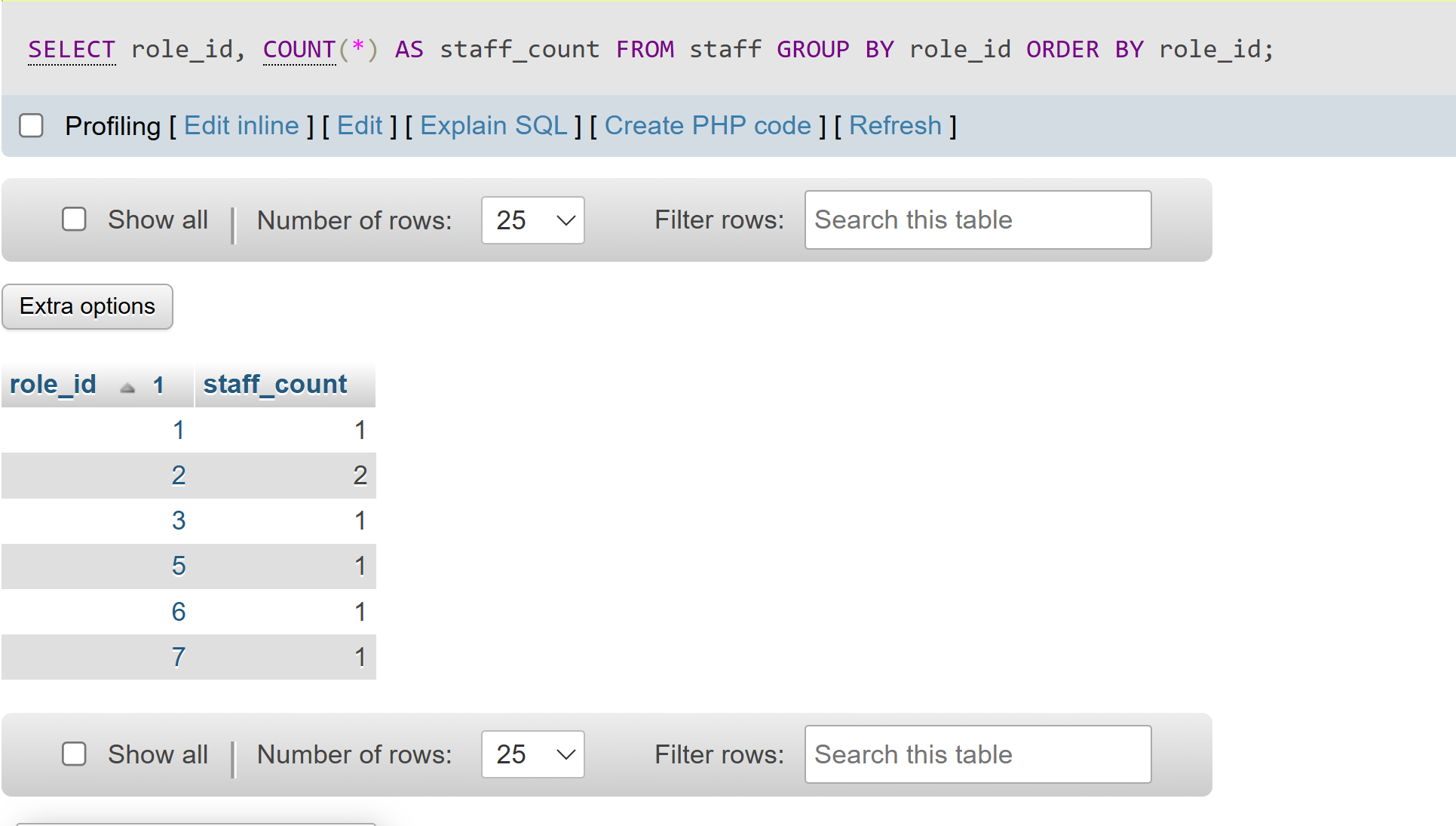
1. How many patients were attended to per appointment status (e.g., Scheduled, Completed)?



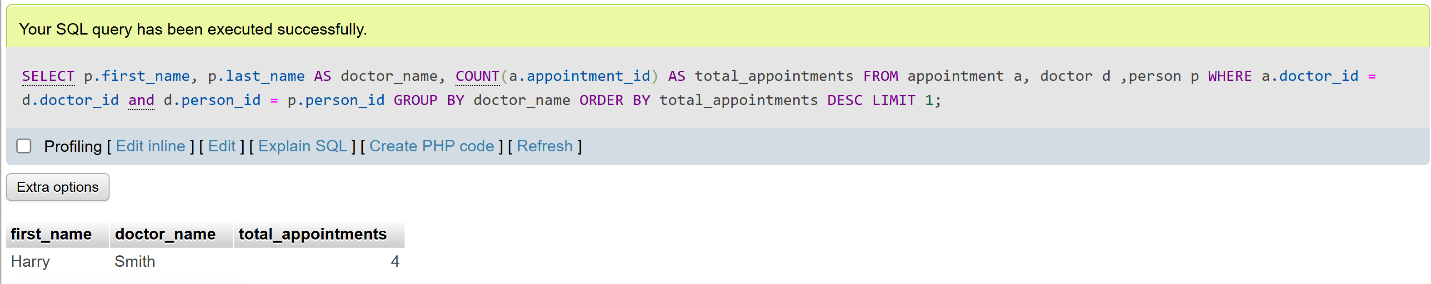
1. How many feedback responses are there for each rating type (Excellent, Good, etc.)?



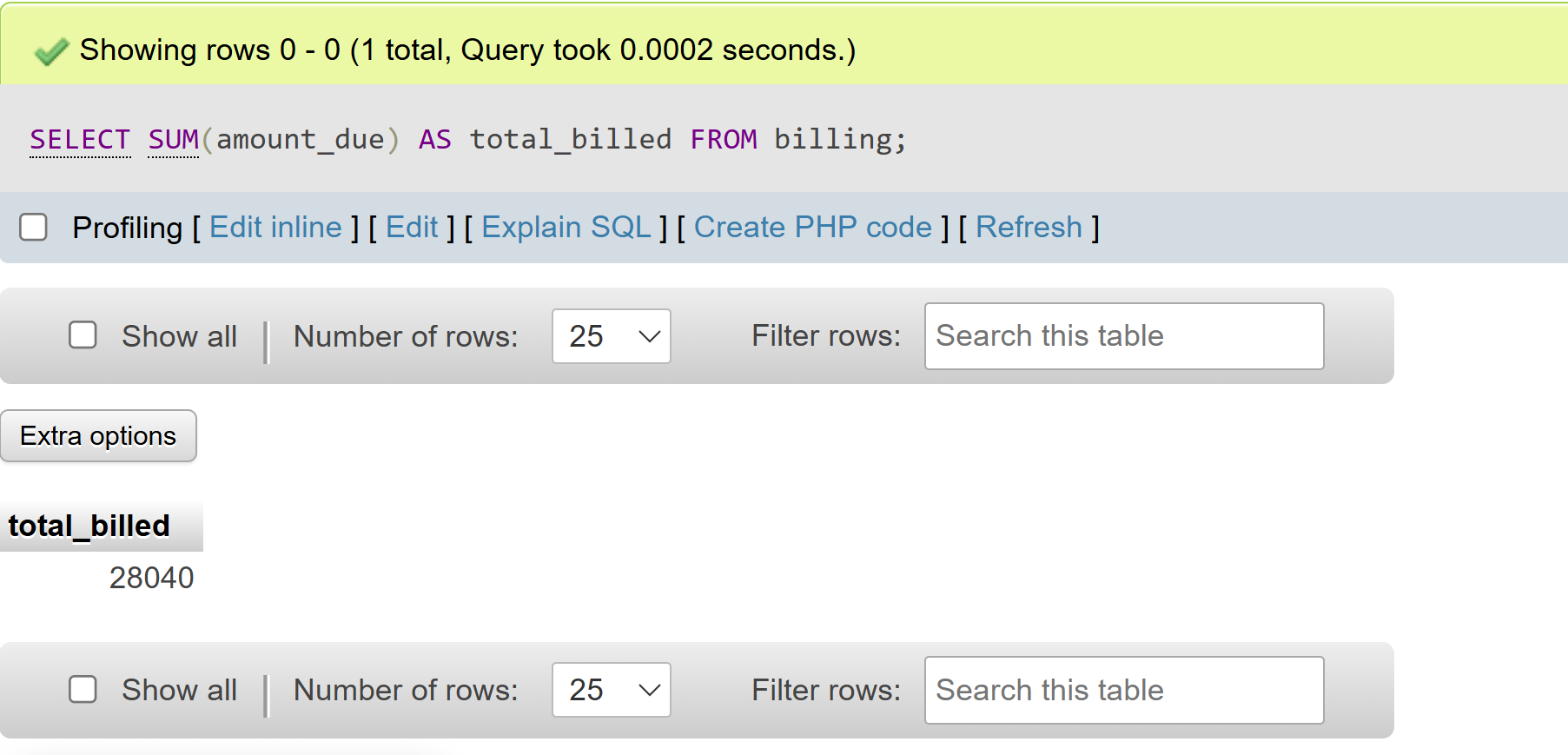
1. Show how many staff members are assigned to each role?



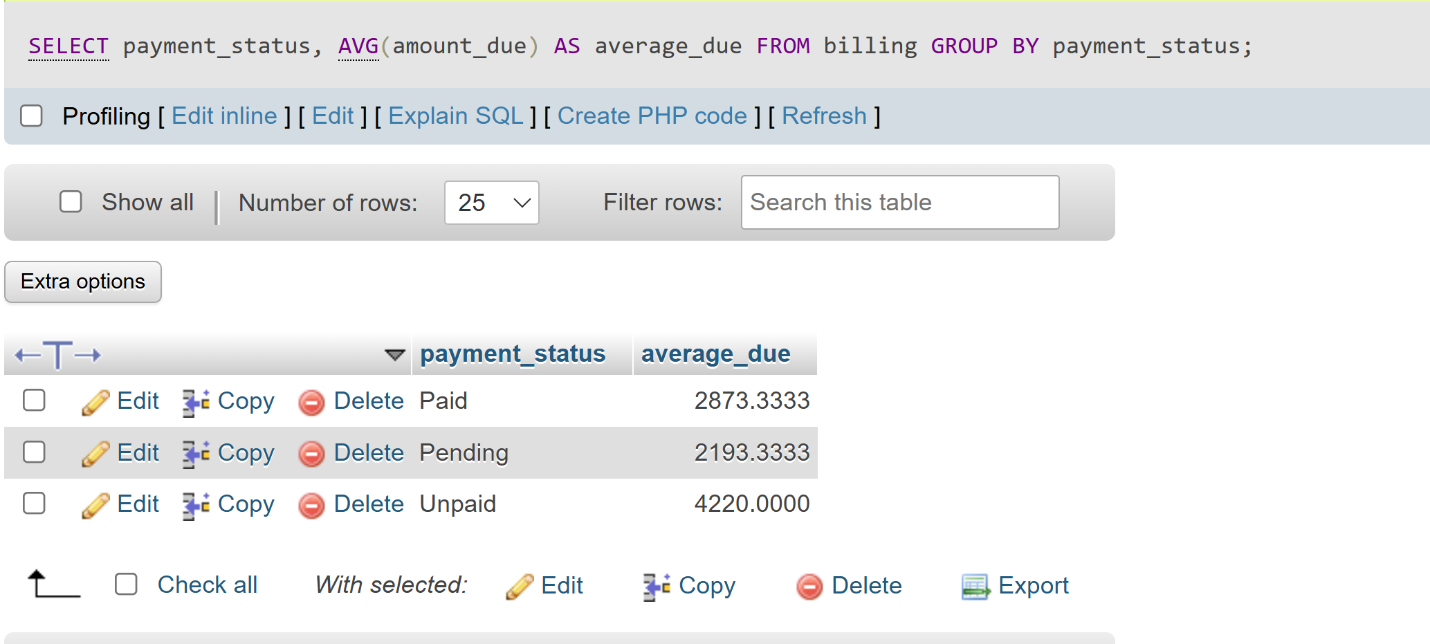
1. Which doctor has handled the most appointments? Show their name and the number of appointments.



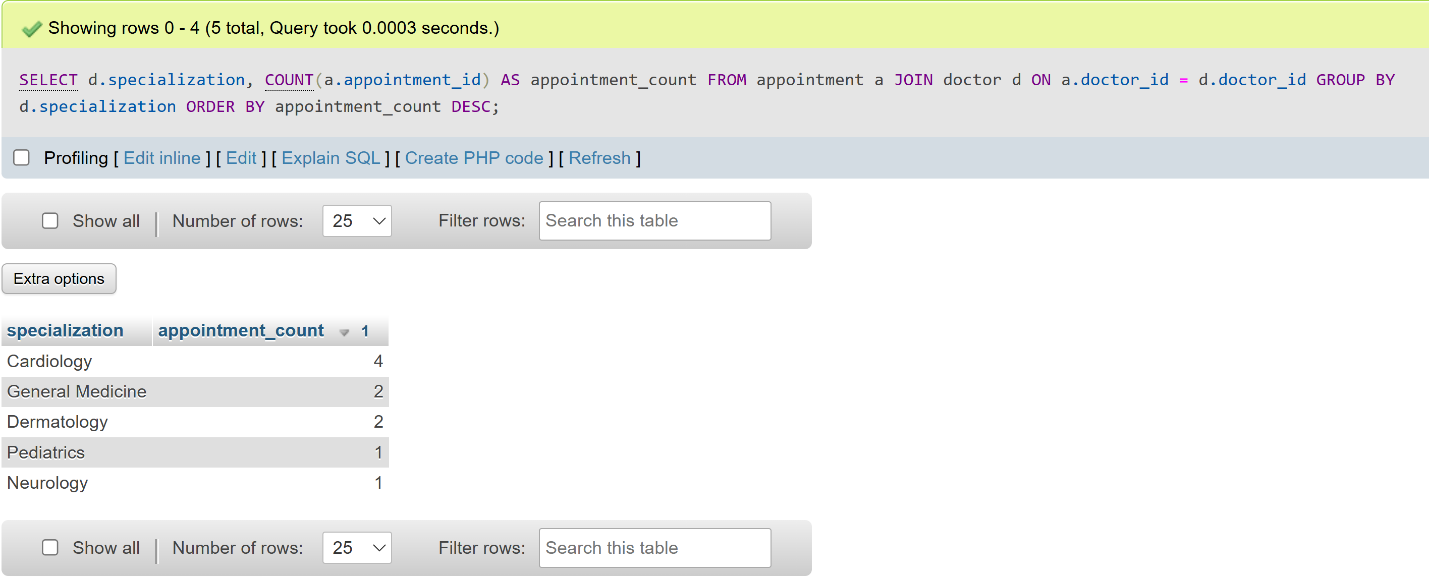
1. What is the total amount billed in the system?



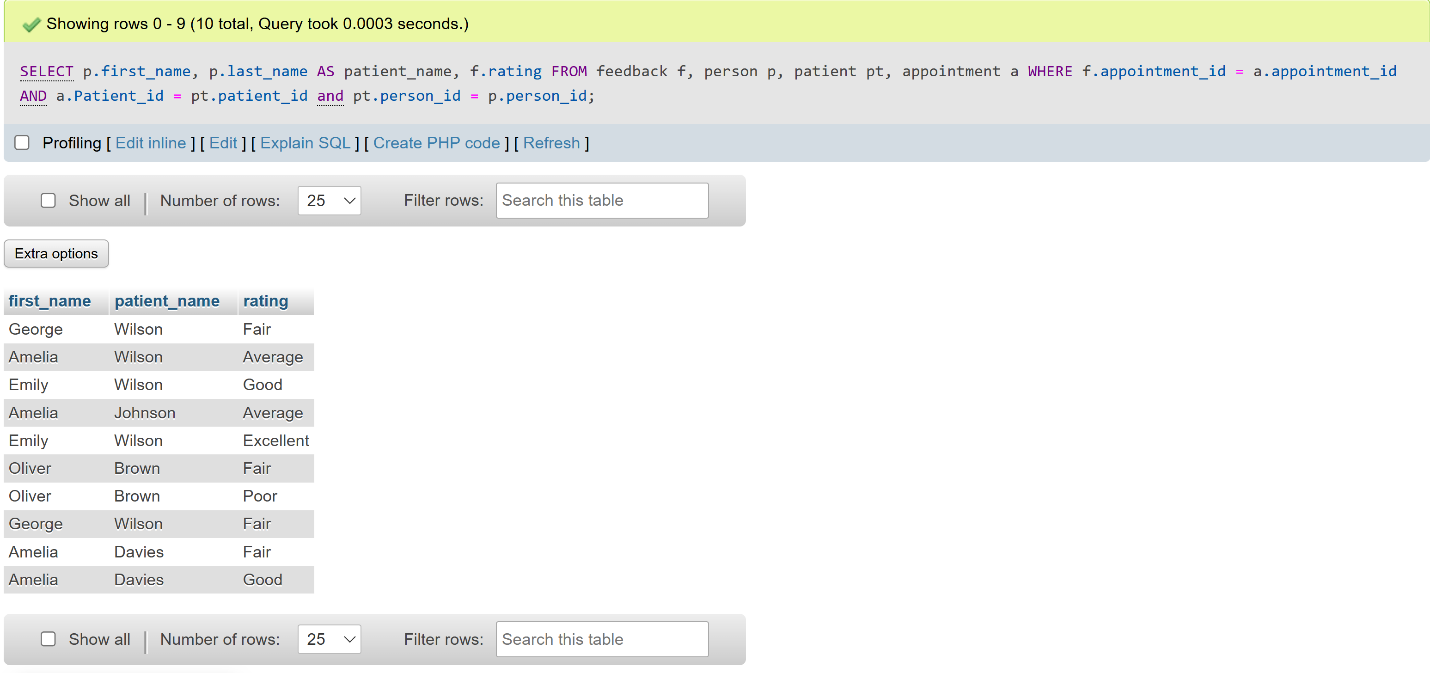
1. What is the average amount due per payment status?



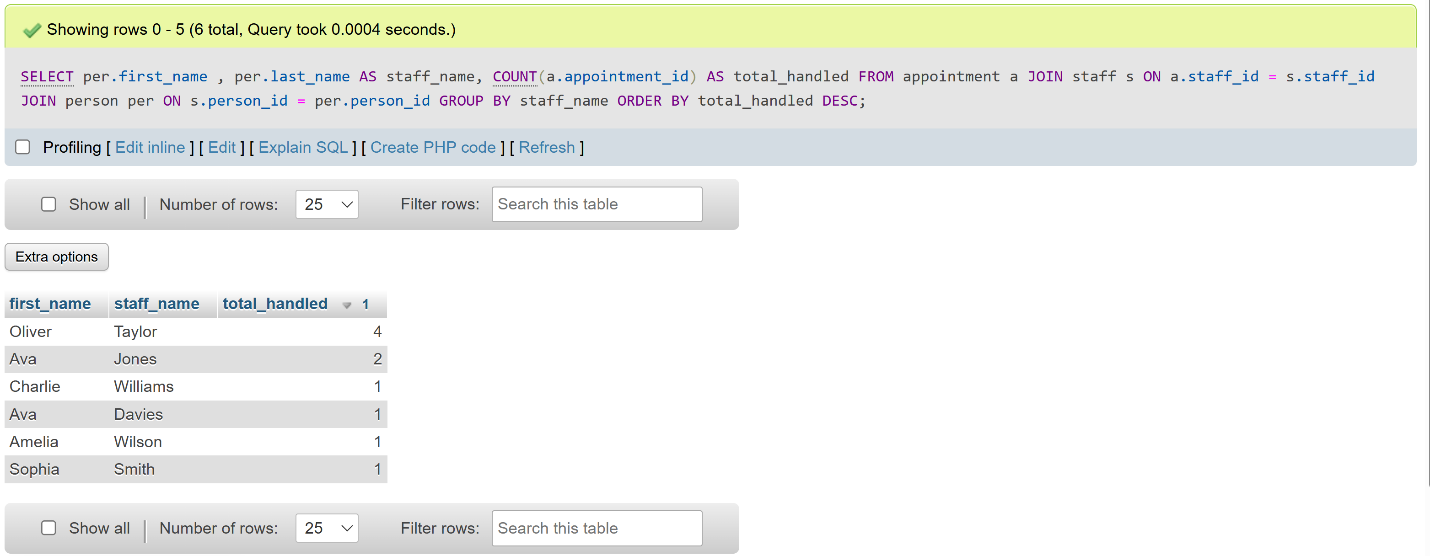
1. Show the number of appointments per doctor specialization



1. List patients who have left feedback, along with their rating.



1. **Count how many appointments were handled by each staff member.**

****

1. List all patient names and their total number of appointments.

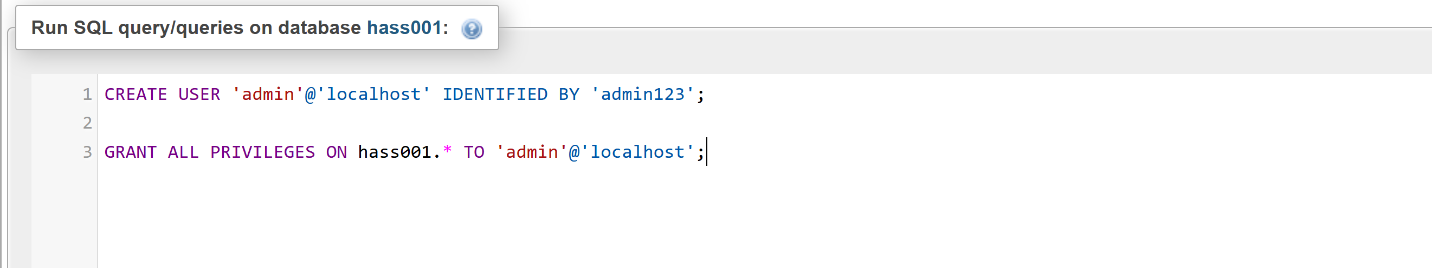
****

# Security Scenario

In any healthcare-related system, safeguarding sensitive patient information and controlling user access is a critical requirement. The HASS (Healthcare Appointment Scheduling System) database manages confidential data such as medical records, billing, and personal information, which must be protected from unauthorized access or misuse. This section outlines the security measures implemented in the database using SQL-based access control. It includes the creation of users based on real-world roles (e.g., doctor, receptionist, billing officer), the assignment of least-privilege access to ensure users can only perform actions relevant to their responsibilities, and the use of GRANT statements to enforce these rules. The following security scenarios demonstrate how role-based access control has been implemented to support data confidentiality, integrity, and compliance with best practices.

## Security Scenario 1 – Administrator Role

In this scenario, the database administrator is responsible for overseeing the entire HASS system, including managing users, backing up data, updating system structures, and performing audits. The administrator requires unrestricted access to all database objects to fulfill these responsibilities. To implement this, a user named admin is created and granted ALL PRIVILEGES on the hass001 database. This gives the administrator full control over all tables, views, and procedures, ensuring they can maintain the system effectively while being accountable for changes.



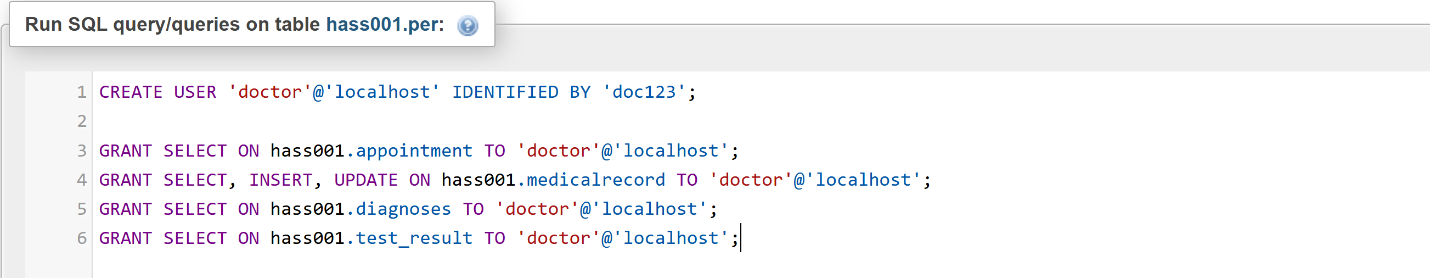
## Security Scenario 2 – Receptionist Role

In this scenario, the receptionist is responsible for scheduling and updating patient appointments but should not have access to sensitive medical or billing information. To enforce this restriction, a dedicated database user is created with permissions limited to the appointment and person tables. The receptionist user is granted SELECT, INSERT, and UPDATE privileges on the appointment table to manage scheduling, and SELECT access on the person table to retrieve basic patient identity details when booking. This ensures that the receptionist can perform their duties efficiently without violating patient confidentiality or accessing clinical data.



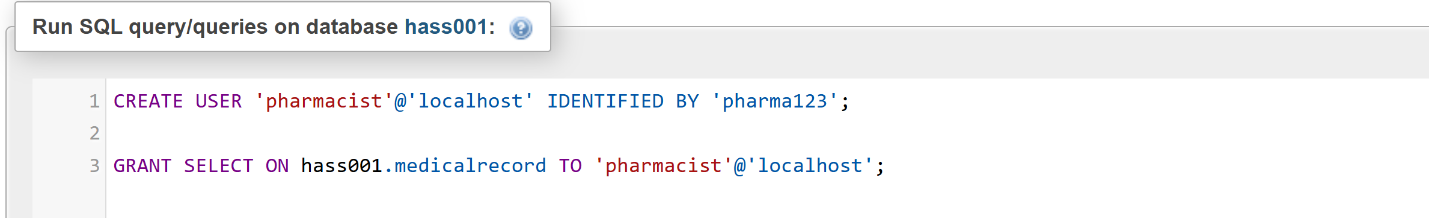
## Security Scenario 3 - Doctor Role

In this scenario, the doctor is responsible for viewing appointments and updating patients’ clinical records, including diagnoses and test results. However, doctors should not access billing or administrative data. To implement this, a database user named doctor is created and granted SELECT access on the appointment table to view scheduled visits. The doctor is also granted full (SELECT, INSERT, UPDATE) privileges on the medicalrecord table and read-only access (SELECT) to the diagnoses and test\_result tables. These permissions enable the doctor to perform all medical duties while maintaining strict access boundaries to sensitive financial or administrative data.



## Security Scenario 4 – Pharmacist Role

In this scenario, the pharmacist is responsible for dispensing medications based on prescriptions written by doctors. Therefore, their access should be strictly limited to viewing prescription information recorded in medical records. The pharmacist does not need access to patient appointments, billing information, or diagnoses. To enforce this restriction, a database user named pharmacist is created and granted only SELECT privileges on the medicalrecord table. This allows the pharmacist to view prescriptions without being able to modify any clinical data, ensuring compliance with data protection principles and supporting secure collaboration with doctors.



## Security Scenario 5 – Billing Officer Role

In this scenario, the billing officer is tasked with managing patient payment records, generating billing statements, and tracking insurance claim statuses. The billing officer does not require access to clinical information such as diagnoses, test results, or medical records. To enforce proper access control, a user named billing is created with privileges limited to the billing table for full access (SELECT, INSERT, UPDATE) and read-only access (SELECT) on the person table. This setup allows the billing officer to link billing entries to patients by name without compromising clinical confidentiality, thereby preserving the principle of least privilege.



## Security Scenario 6 – Lab Technician Role

In this scenario, the lab technician is responsible for processing and viewing patient test results. However, they should not have access to full medical records, patient appointments, or billing details. To implement this, a user named labtech is created and granted only SELECT privileges on the test\_result table. This ensures that lab technicians can view test results for documentation or reporting purposes without the ability to view or alter other sensitive information, maintaining data confidentiality and workflow integrity within the healthcare system.



# Conclusion

The development of the Healthcare Appointment Scheduling System (HASS) demonstrates the practical application of database design, normalization, and implementation techniques in managing real-world healthcare data. By incorporating essential entities such as patients, doctors, staff, appointments, and billing, the system supports core hospital functions efficiently. Emphasis was placed on data integrity, relational structure, and security. Role-based access control was implemented to ensure that users only interact with data relevant to their responsibilities, thereby protecting sensitive medical information and promoting compliance with privacy standards. This project not only highlights the importance of secure database design but also reflects the potential for structured data systems to improve healthcare operations and service delivery.

# References

Atkinson, M., Bancilhon, F., DeWitt, D. et al. (1989). The Object-Oriented Database System Manifesto. Proceedings of the First International Conference on Deductive and Object-Oriented Databases.

Chodorow, K. (2013). MongoDB: The Definitive Guide. 2nd ed. O’Reilly Media.

Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 13(6), pp. 377–387.

Coronel, C. and Morris, S. (2015). Database Systems: Design, Implementation, and Management. 11th ed. Cengage Learning.

Date, C. J. (2003). An Introduction to Database Systems. 8th ed. Addison-Wesley.

Elmasri, R. and Navathe, S. (2017). Fundamentals of Database Systems. 7th ed. Pearson.

Han, J., Haihong, E., Le, G. and Du, J. (2011). Survey on NoSQL database. Proceedings of the 6th International Conference on Pervasive Computing and Applications, pp. 363–366.

Lakshman, A. and Malik, P. (2010). Cassandra: a decentralized structured storage system. ACM SIGOPS Operating Systems Review, 44(2), pp. 35–40.

Mokrani, K. (2015). Evaluation of Object-Oriented Database Systems. International Journal of Computer Science and Information Security, 13(5).

Moniruzzaman, A.B.M. and Hossain, S.A. (2013). NoSQL Database: New Era of Databases for Big Data Analytics. International Journal of Database Theory and Application, 6(4), pp. 1–14.

ObjectDB (2021). ObjectDB Object Database for Java. [online] Available at: https://www.objectdb.com [Accessed 4 May 2025].

Oracle (2020). Oracle Database 19c Overview. [online] Available at: https://www.oracle.com/database/technologies [Accessed 4 May 2025].

Pritchett, D. (2008). BASE: An ACID Alternative. Queue, 6(3), pp. 48–55.

Robinson, I., Webber, J. and Eifrem, E. (2015). Graph Databases. 2nd ed. O’Reilly Media.

Stonebraker, M. and Rowe, L.A. (2010). The case for PostgreSQL. Communications of the ACM, 53(2), pp. 60–68.

TechRepublic (2018). Microsoft SQL Server: Pros and cons. [online] Available at: https://www.techrepublic.com [Accessed 4 May 2025].

Vacca, J. R. (2014). Computer and Information Security Handbook. 2nd ed. Morgan Kaufmann.

Versant (2008). db4o Database for Java and .NET. [online] Available at: https://www.db4o.com [Accessed 4 May 2025].

Widenius, M. and Axmark, D. (2002). MySQL Reference Manual. O'Reilly Media.