**Databases**

**Course Project**



**Technical University of Sofia**

*Alexander Svilarov, Nikolay Nikolov, Stefan Dimitrov, Vesselina Belova*

2024

**Task Description**

**Topic:** *Hospital Management System*

**Tables to be implemented**: *Patient, Doctor, Bill, Room, Services*

**Stage 1: Requirements Analysis**

**• Identifying the purpose and scope of the database – what data needs to be stored and why.**

**• Determining what applications will use the database – this helps anticipate future data access needs.**

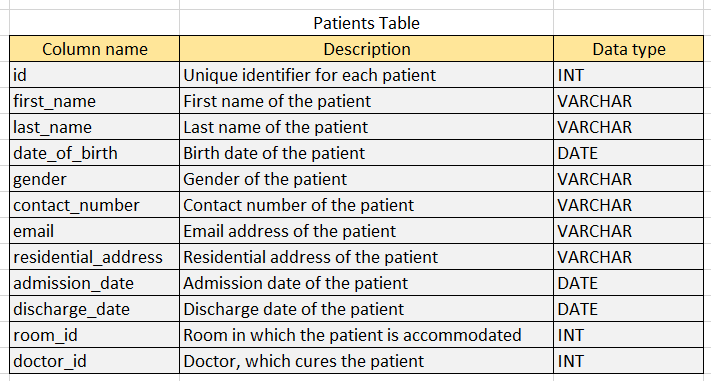
**• Identifying crucial entities, attributes, relationships, and constraints for the data model.**

* The purpose of the database is to **store and analyze information** for a hospital – this includes storing **patient info**, **doctor info**, **bills,** **room information** and **service information**. The scope of the database in our **Hospital Management System** is to provide an **efficient**, **reliable**, and **centralized** way to **manage all hospital-related information.**
* Our **Hospital Management System** will be widely used in **different applications**, mainly by **real hospitals**. This database system will help them to **organize their important information**, to **store it** and to **analyze it.** Doctors may need easy access to **patient information, diagnosis and treatments**. Administrators will need easy access to the whole **database structure**.
* The entities that we will include in our database are: **Patients entity**, **Doctors entity**, **BillReports entity**, **Rooms entity**, **Services entity.**

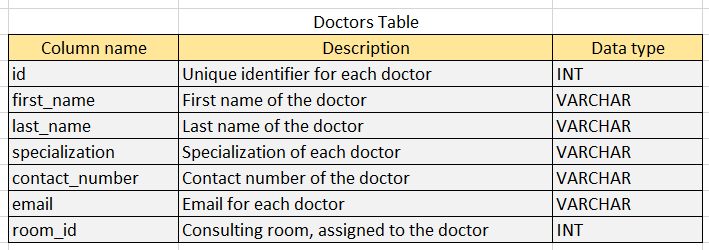
**Stage 2 Task. Conceptual Data Modeling**

* **Determining the attributes for each entity – these are characteristics that describe or qualify the entity.**
* **Defining relationships between entities, such as one-to-one, one-to-many, or many-to-many.**
* **Representing the entities and relationships using modeling methodologies like entity relationship diagrams**
* Here we will describe the tables – the column names, the description of the column and the data type it will be contained within. We have created the following excel document, in which all of the tables are described:

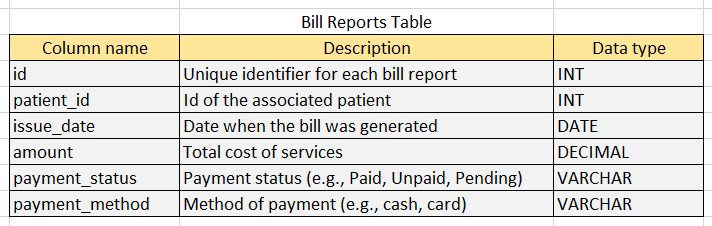
Patients Table



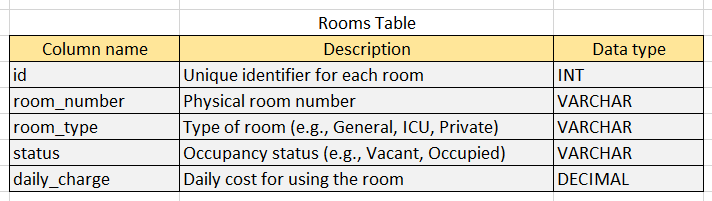
Doctors Table



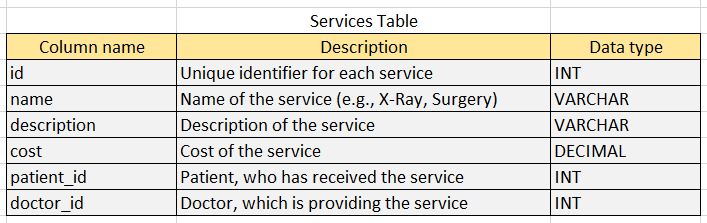
Bill Reports Table



Rooms Table

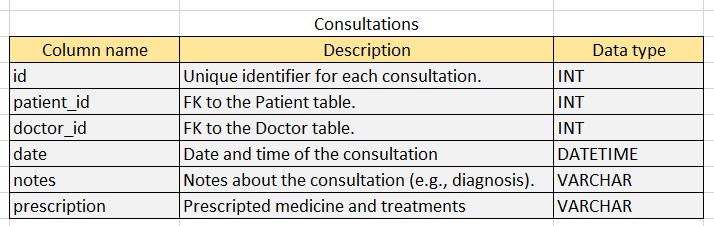


Services Table



* Of course, we would need a **mapping table**, to be able to achieve the needed **relationships**. We will utilize the **Consultation table** – It is necessary for the **Doctor – Patient mapping**. The doctors may need to gain information about a **patient** **diagnosis**, **prescribed** **treatment**, **consultation** **date** etc.

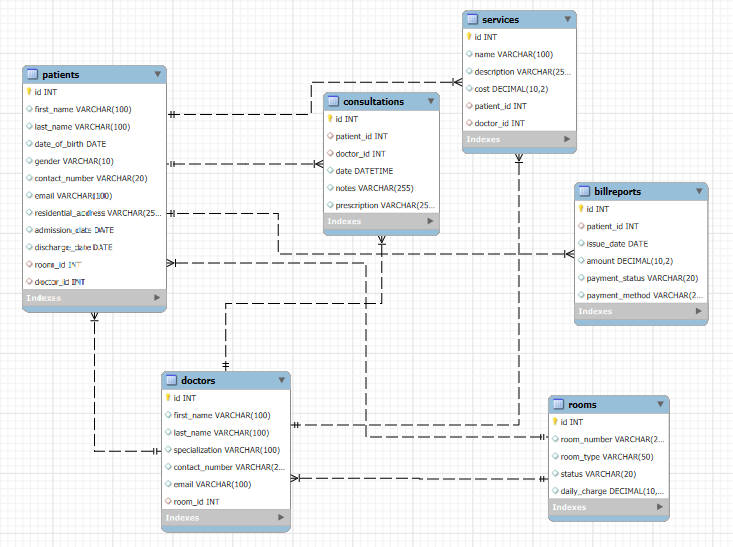
Consultations Table



**Stage 3 tasks. Logical Database Design**

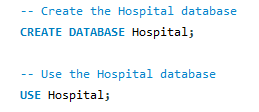
* In the image shown below we can see the generated database diagram and all of its necessary relationships. Let’s break down the diagram:
  + **Patient to Room (*one-to-many*)** – Each patient can be assigned to one room, but one room can have multiple patients assigned to it.
  + **Patient to Doctor (*many-to-many*)** – Each patient can have multiple doctors and each doctor can treat many patients.
  + **Patient to Consultation (*one-to-many*)** – Each patient can have many consultations, but each consultation can be for one patient only.
  + **Patient to Bill (*one-to-many*)** – Each patient can have multiple bill reports (e.g., for different treatments or hospital stays), but each bill is associated with one patient.
  + **Patient to Service (*one-to-many*)** – Each patient can receive multiple services (e.g., surgeries, tests, treatments), but each service is provided to one patient only.
  + **Doctor to Room (*one-to-one*)** – Each doctor is assigned to one room, and each room is typically assigned to one doctor for consultation purposes.
  + **Doctor to Consultation (*one-to-many*)** – Each doctor can have multiple consultations, but each consultation can involve only one doctor.
  + **Doctor to Service (*one-to-many*)** – Each doctor can provide multiple services (e.g., surgeries, procedures, consultations), but each service is provided by one doctor.
  + **Room to Bill (*one-to-many*)** – Each room can be associated with multiple bill reports (e.g., for different patients assigned to it over time), but each bill report corresponds to one room based on the patient's room assignment.
  + **Service to Bill (*one-to-many*)** – Each service may appear in multiple bill reports (if the service is provided to multiple patients), but each bill report is associated with the specific services provided to the patient
  + **Consultation to Bill (*one-to-many*)** – Each consultation can result in one or mosre bills (depending on the treatments or tests provided), but each bill report is tied to one consultation.

Entity Relationship Diagram



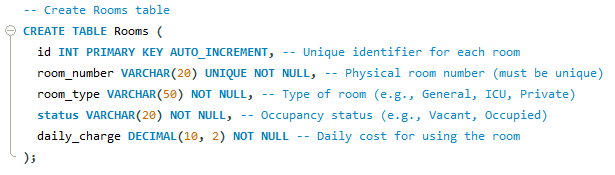
**Stage 4 Task. Physical Database Design**

**4.1. Database creation**

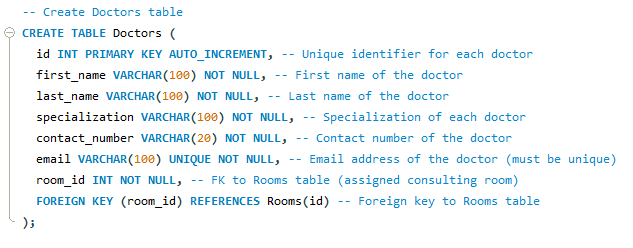
****

**4.2. Tables creation**

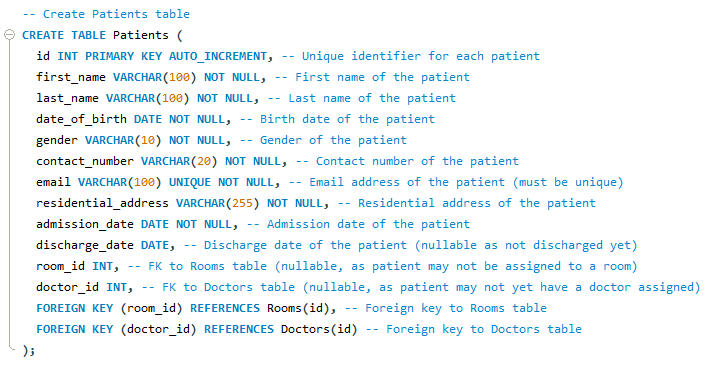
* *Rooms table design:*



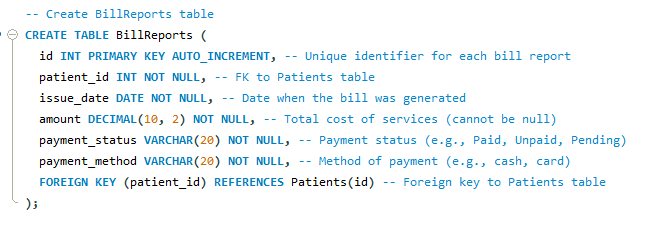
* *Doctors table design:*



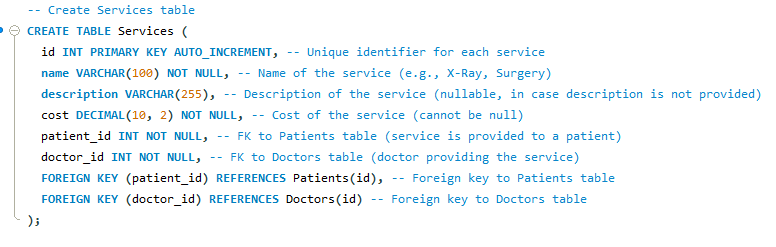
* *Patients table design:*



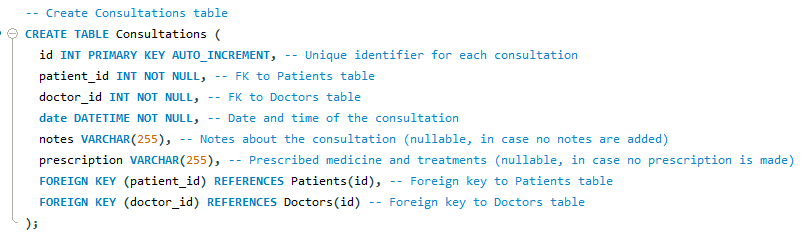
* *BillReports table design:*



* *Services table design:*

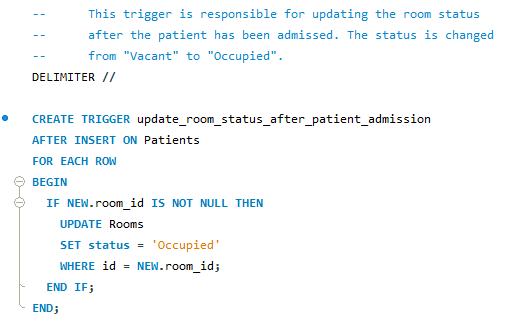


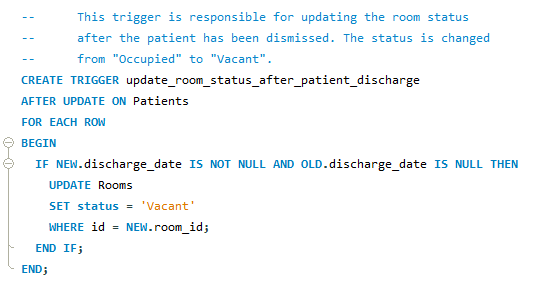
* *Consultations table design:*

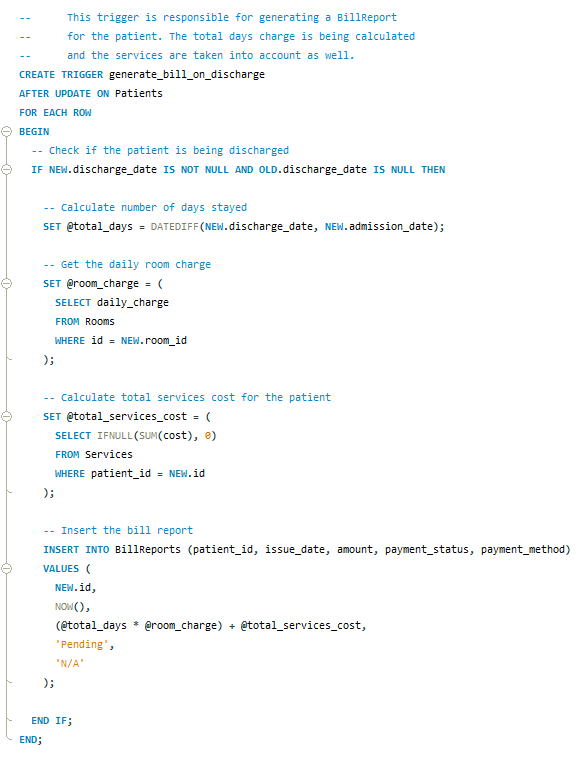


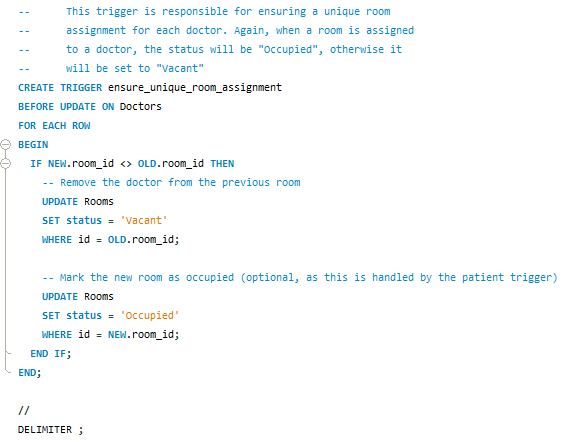
**Stage 5 tasks. Database Implementation**

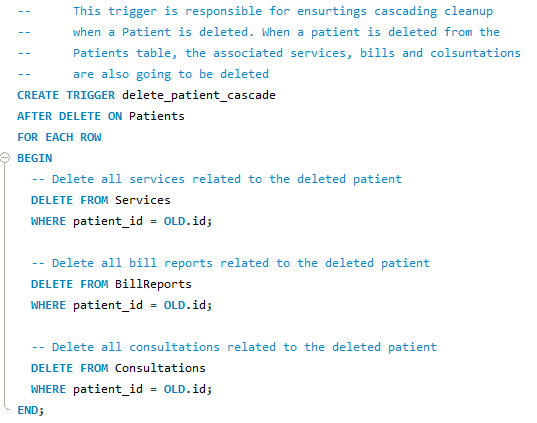
**5.1. Database triggers**





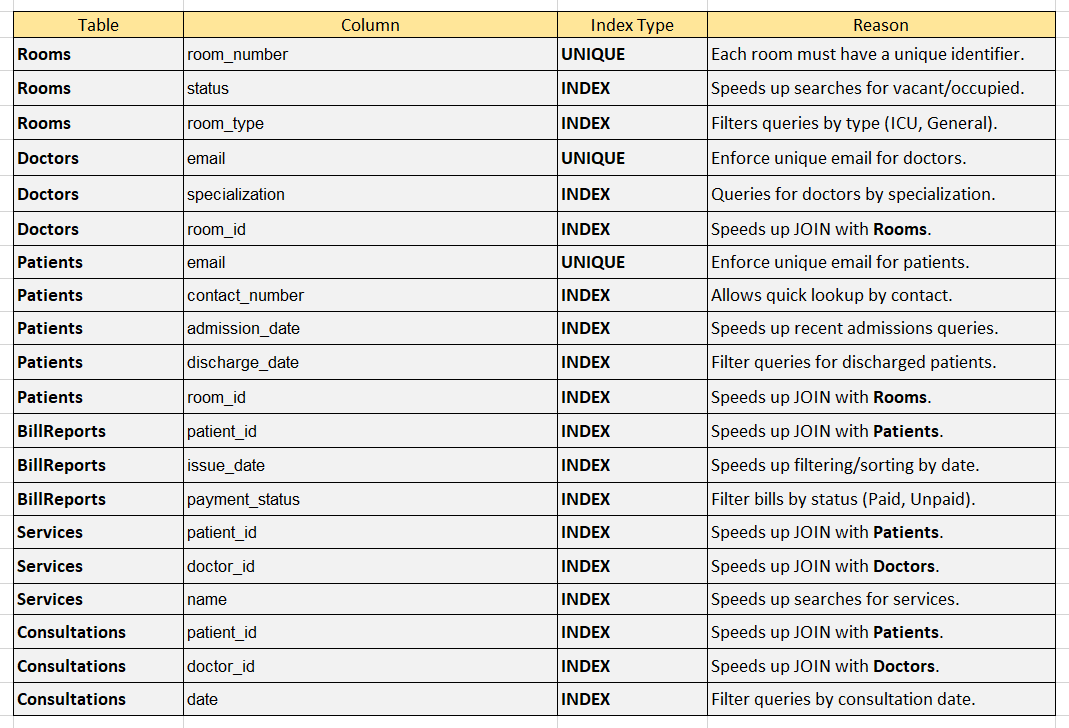




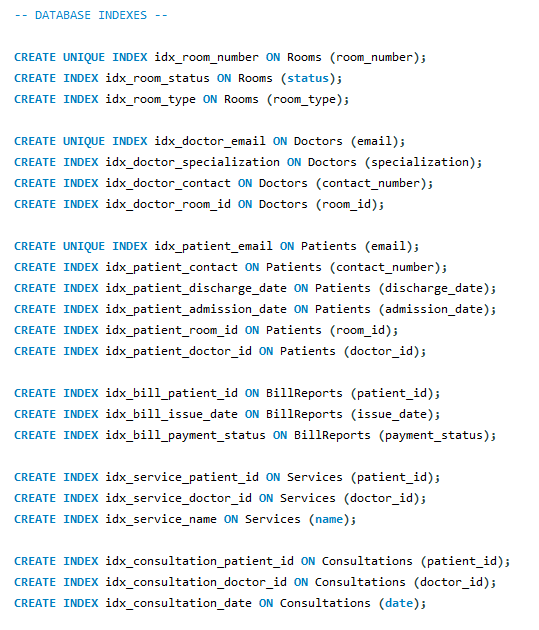


**5.2. Database indexes**

* Indexes speed up queries, especially those involving searches, joins, and lookups. We will create **primary indexes, unique indexes, and foreign key indexes** where appropriate.



* Here you can see the **implementation of the indexes** that were specified in the previous page:



**5.3. User roles**

* User roles are essential for **security, organization, and efficiency** in a database. By assigning specific roles (*Admin, Doctor, Billing Staff, Patient*), access to sensitive data is **restricted based on responsibility**.
* This approach enforces **data privacy** (*patients see only their records*) and **prevents accidental changes** (*billing staff can't edit medical records*).
* Overall, user roles enhance **data integrity, reduce risks, and maintain compliance** with privacy regulations like **GDPR or HIPAA** in healthcare.

**5.3.1. User role: Admin**

***Access***: Full access to all tables and operations (**CRUD**).

***Responsibilities***: Manages doctors, patients, rooms, billing, and user roles.

**5.3.2. User role: Doctor**

***Access***: Can view and update Patients, Consultations, and Services.

***Responsibilities***: Manages patient consultations, records prescriptions, and tracks patient progress.

**5.3.3. User role: Billing staff**

***Access***: Can view and update BillReports and Services.

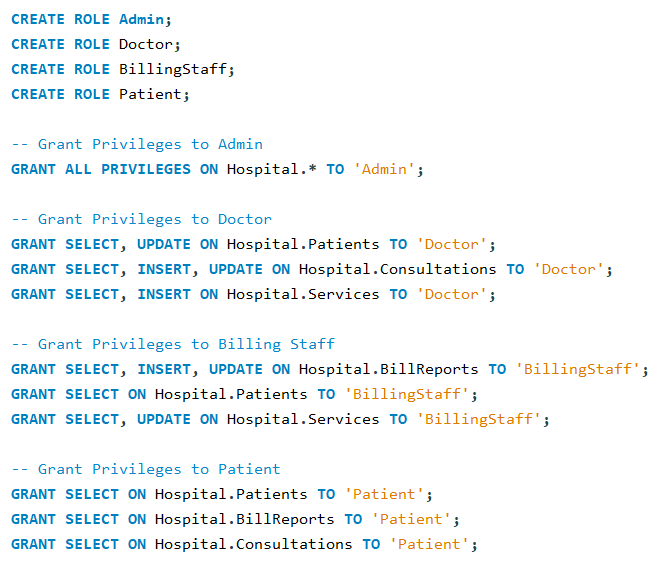
***Responsibilities***: Generates bills, tracks payments, and updates billing records.

**5.3.4. User role: Patient**

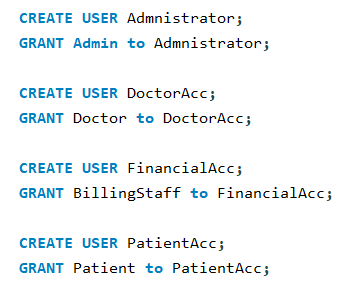
***Access***: Can view their own Patient record, BillReports, and Consultations.

***Responsibilities***: View personal information, medical history, and bill status.

* Here you can see the **implementation of set roles**.

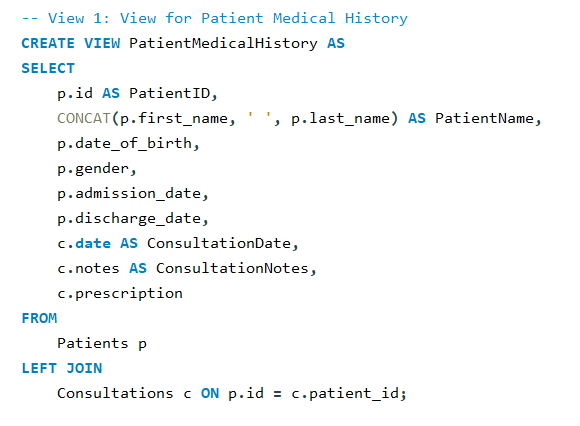


* Here you can see the **creation of the users** and assigning of their roles.

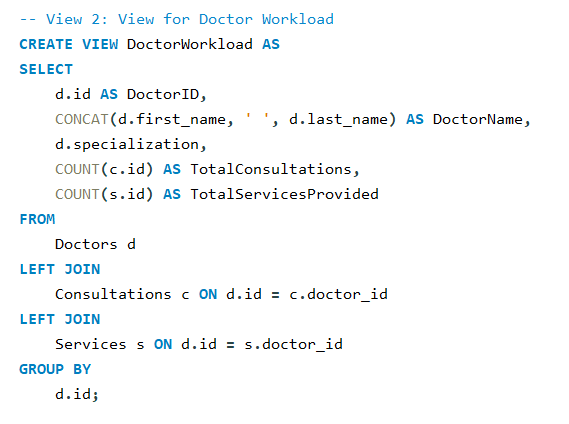


**5.4. Views**

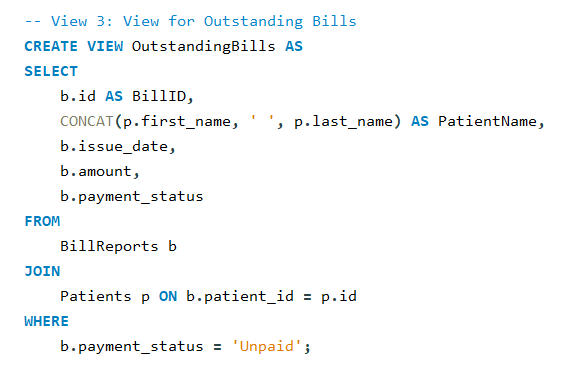
* Views are **virtual tables** that **simplify database queries** and **enhance security**, **efficiency**, and **readability**. By creating views, users can access specific data without exposing entire tables, ensuring **data privacy**. Views allow complex queries to be reused easily, **reducing redundancy in SQL code**.
* Views **also boost security** by controlling access to **sensitive information**. Additionally, they provide **consistency**, ensuring users see up-to-date, **accurate information**. For reporting and analytics, views make it easier to present **meaningful insights** without exposing the **underlying database** structure or logic.
* This view will provide us with **important information**, regarding the patient’s m**edical history**. We will be able to access the patient’s **ID, name, birth date, gender, admission and discharge dates, consultation date, consultation notes and the prescription**. This view **unifies the patient’s information** from the tables **Patients** and **Consultations**.



* This view will provide us with **important information** regarding the **doctor’s workload**. We will be able to see the **doctor’s ID, name, specialization, total consultations and total services provided**. This view **unifies the doctor’s information** from the tables **Doctors** and **Consultations**.

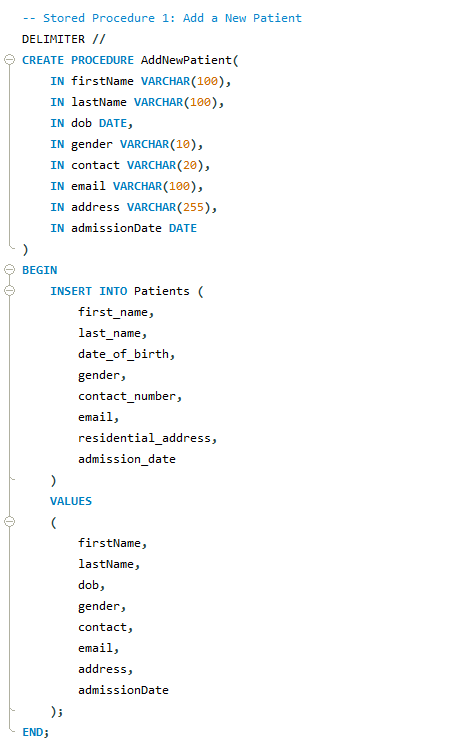


* This view will provide us with **important information** regarding the **patient’s bills**. We will be able to see the **patient’s bill ID, patient name, bill issue date, bill amount and bill payment status**. This view **unifies the patient’s billing information** from the tables **Patients** and **BillReports**.

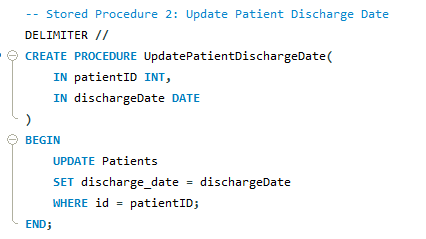


**5.5. Stored procedures**

* **Stored procedures** are **predefined SQL scripts** that perform **specific tasks** in a database. They improve **efficiency, security, and maintainability**. By storing reusable SQL logic on the server, stored procedures **reduce redundancy** and make it **easier to update logic** in one place. They **boost performance** by reducing the need to send multiple queries from applications, as the **logic runs directly on the database server**.
* Stored procedures **enhance security** by **controlling user access**, allowing execution without exposing the underlying queries. They also **prevent SQL injection** by handling **inputs as parameters**. Overall, stored procedures **simplify database automation**, **consistency**, and **modular development**.
* The first **stored procedure** we will implement will provide us with an **easy way to insert a new patient record in the Patients table.**



* The next procedure will help us to **update** the **patient’s discharge date**.

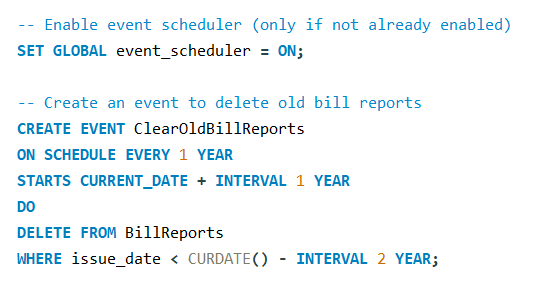


* The next procedure will help us to **generate a patient’s bill** the **patient’s discharge date**.



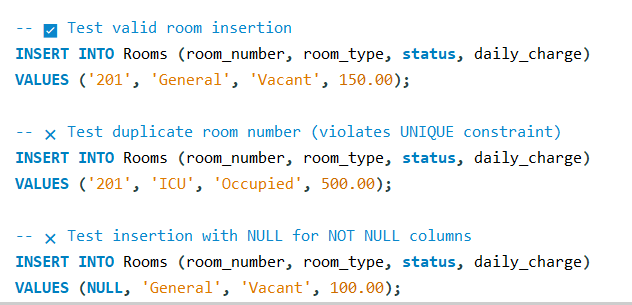
**5.6. Events**

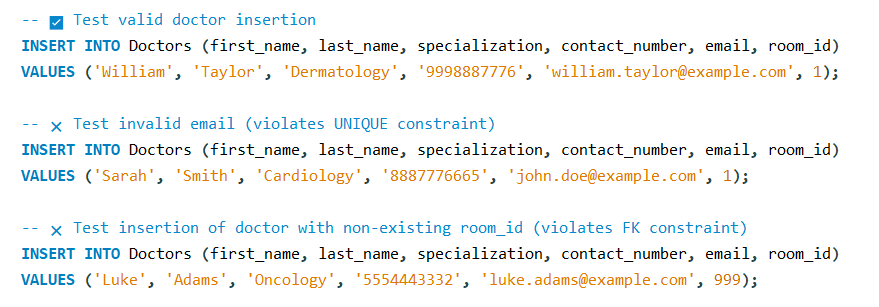
* An **event** is a **scheduled task** that runs **automatically at specified intervals**. It functions like a “*cron job”* for the database, allowing automation of repetitive tasks.
* The **ClearOldBillReports** event runs **once every year** and **deletes billing records** from the **BillReports** table that are **older than 2 years**. This event **reduces database size**, **improves query performanc**e, and **eliminates the need for manual cleanup**. It starts one year from today and continues every year after that.



**Stage 6 tasks. Testing and Quality Assurance**

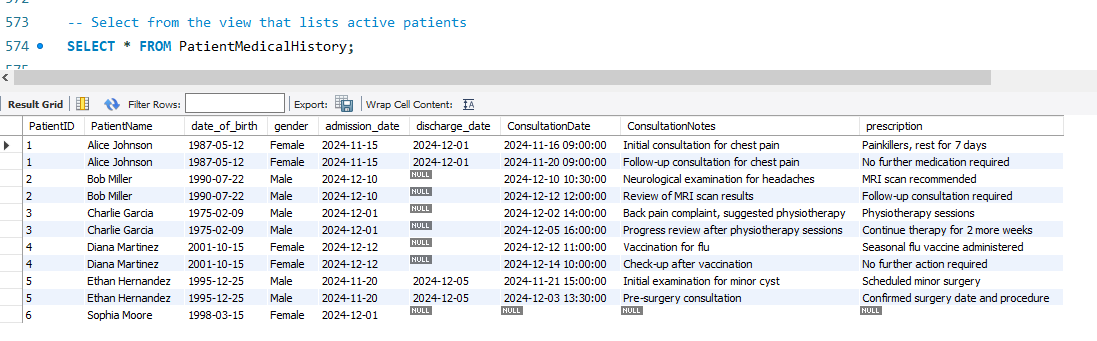
**6.1 Testing database insert violations etc.**

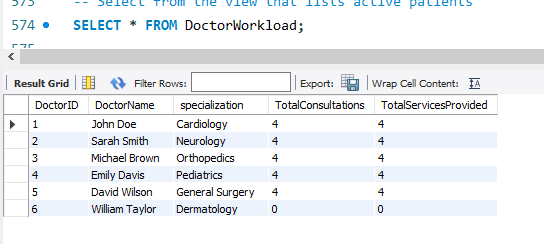
****

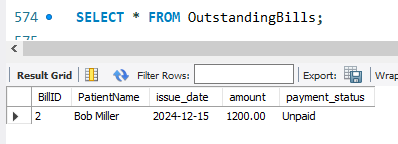
****

****

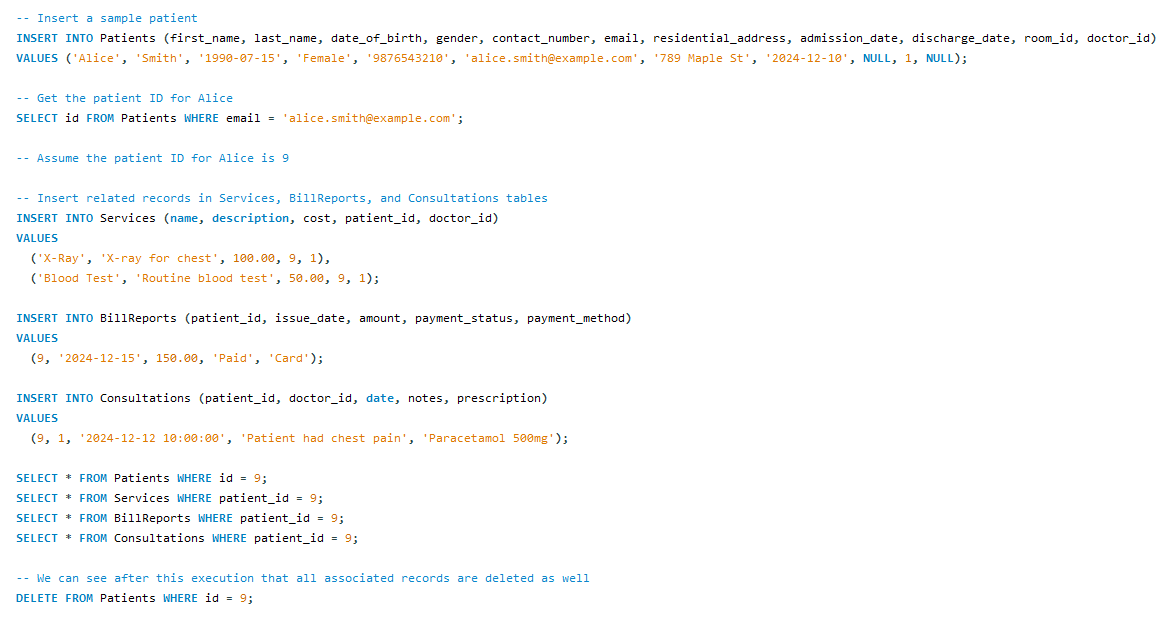
**6.2 Testing database views**

****

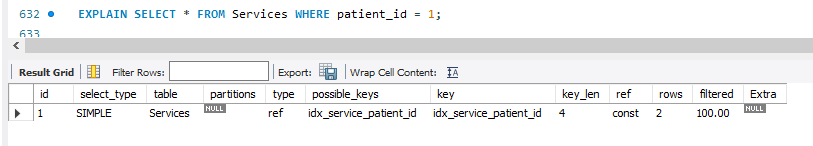
****

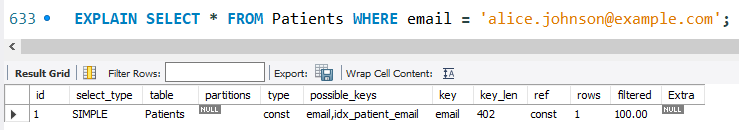
****

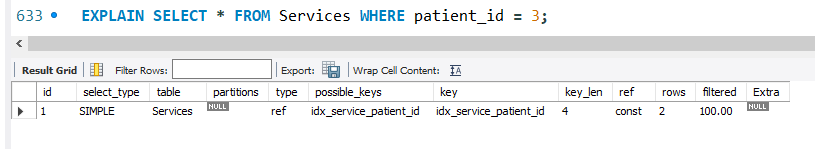
**6.3 Testing database triggers**

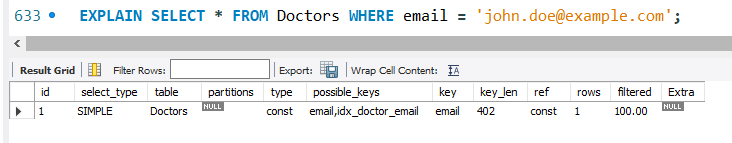
****

**6.4 Testing database indexes**

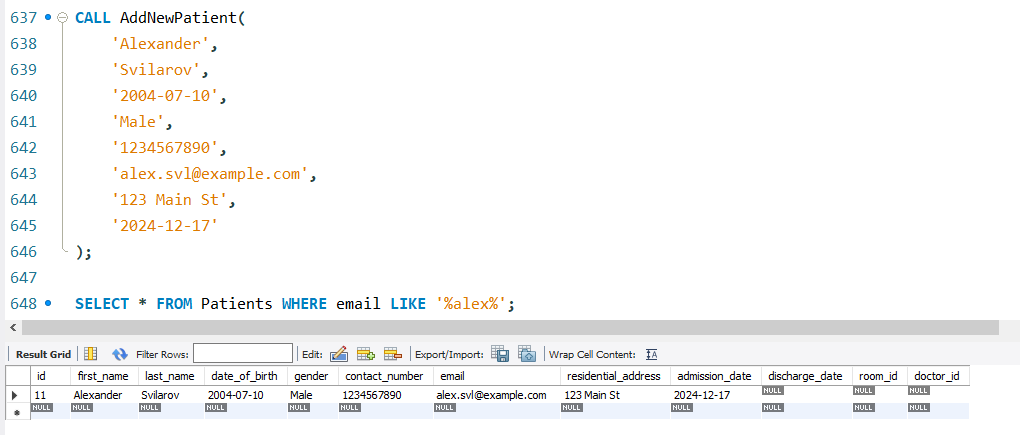
****

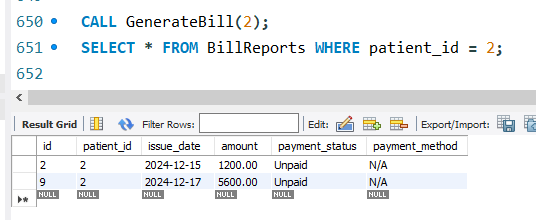
****

****

****

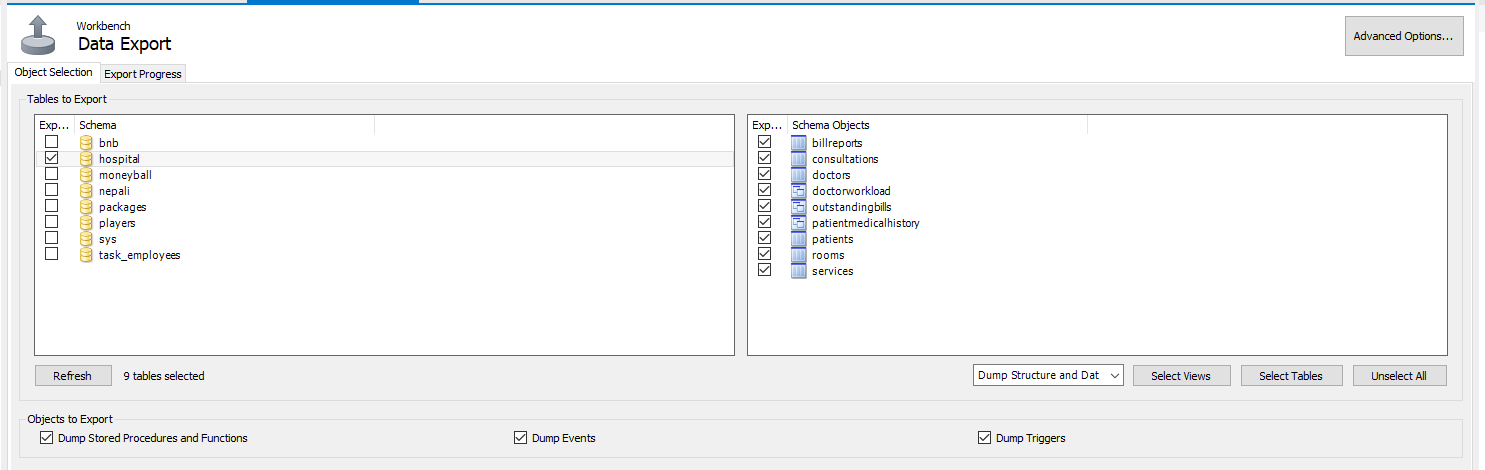
**6.5 Testing database procedures**

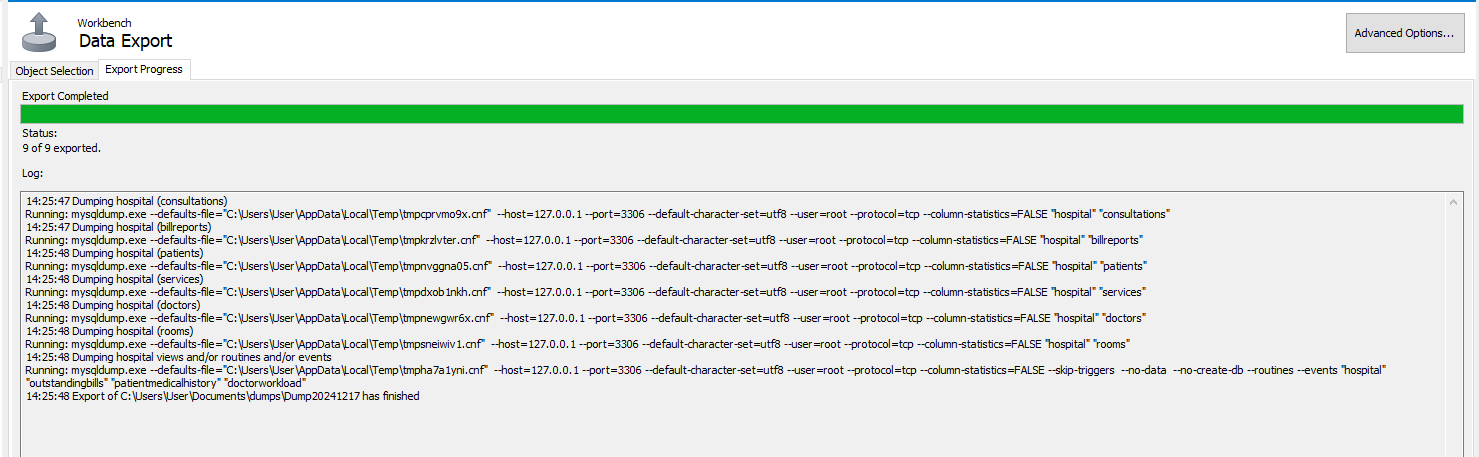
****

****

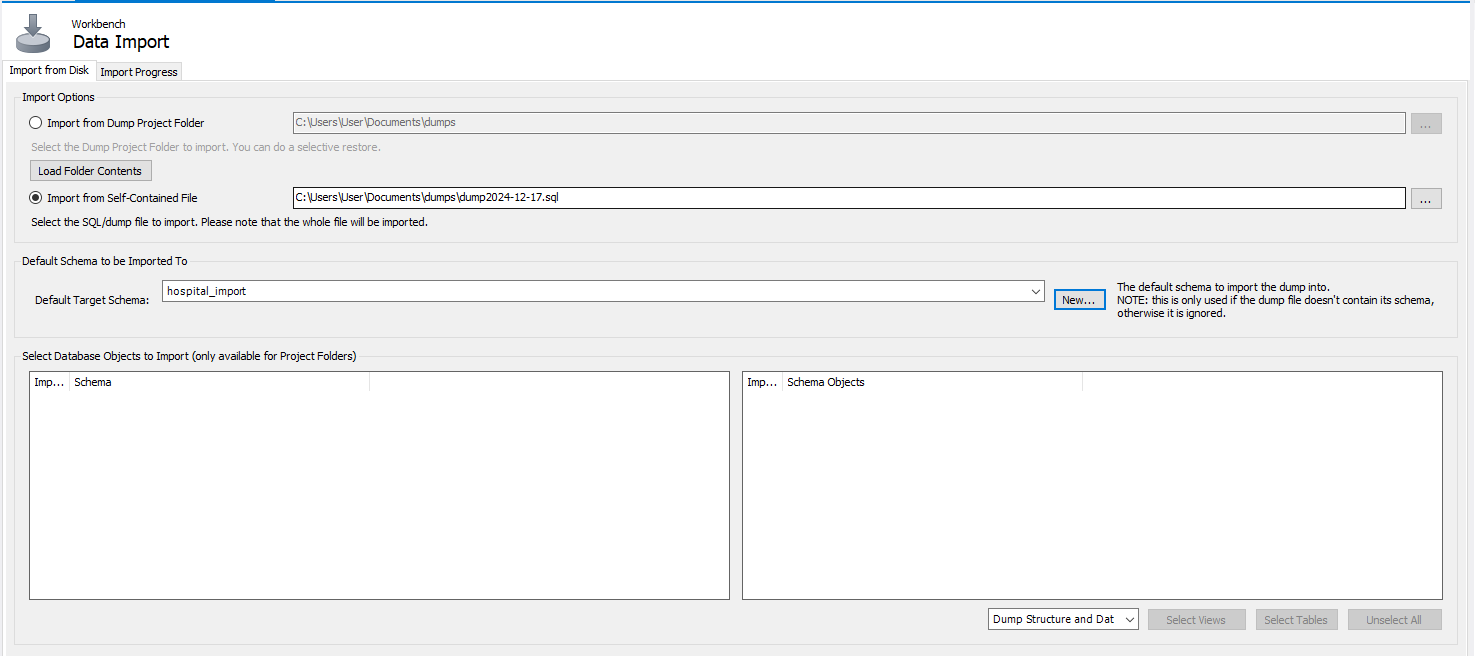
**Stage 7 tasks. Maintenance and Monitoring**

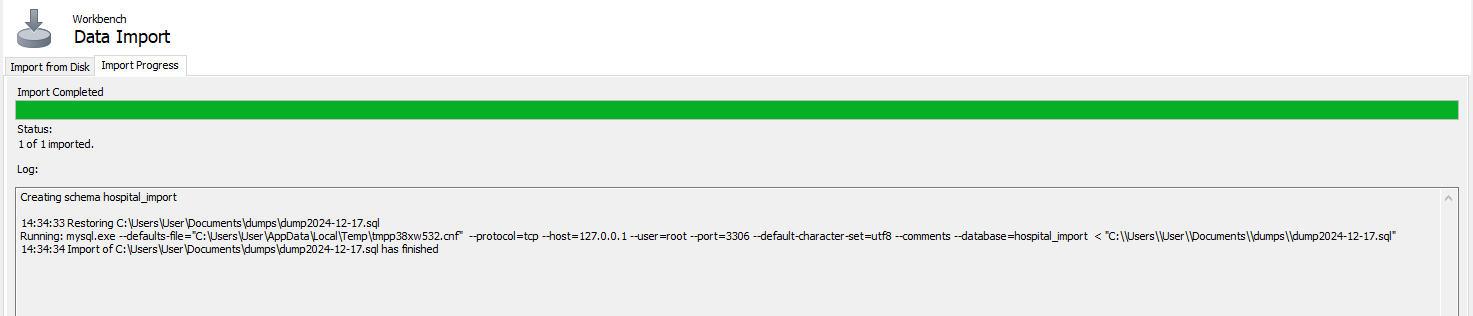
* **Database backups** are crucial for ensur**ing data integrity, availability, and disaster recovery** in case of **unforeseen events** like **hardware failures**, **human errors**, **cyberattacks**, or **system crashes**. Without regular backups, organizations risk **losing valuable data**, which could lead to **operational disruptions**, **legal implications**, and **loss of customer trust.**
* For **effective data management**, it’s important to **do regular backups**, store them securely in multiple locations, and test the restore process.
* Since our **DB management system** is of **big importance to our future clients**, we recommend **weekly database backups**. We are using the Data Export functionality to achieve simple, but effective database backup, which includes not only the structure, but also the information, the triggers, the events, the procedures etc.

****

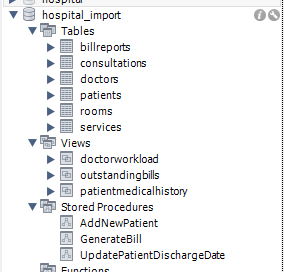
****

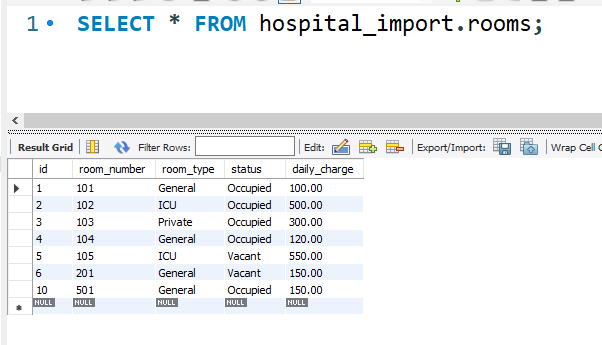
* Now we can test the **import functionality:**

****

****

* As you can see, our database has been **successfully imported** with all the **associated tables, views, procedures etc.**

****

****

**Project Summary: Hospital Management System Database**

* The **Hospital Management System (HMS)** is designed to **streamline** and **automate various hospital operations** such as **patient management**, **room allocations**, **doctor appointments, billing, and medical services**. The goal of this project is to create **a robust, scalable**, and **efficient** database to support the daily functions of a hospital while ensuring **data** **integrity**, **security**, and **accessibility**. The system will manage key entities including **patients, doctors, rooms, medical services, billing, and consultations**, all interconnected via **relational database principles**.
* The database is designed using **MySQL** and includes **multiple tables** namely **Rooms, Doctors, Patients, Services, BillReports, and Consultations**. Relationships between these entities are established using foreign keys to ensure referential integrity. The **database structure** also includes **indexes** to **optimize query performance**, particularly for **frequently accessed data such as patient details, doctor appointments, and billing records.**
* Additionally, a range of **stored procedures, views, triggers, and events** are implemented to **automate processes** and **enhance system efficiency**. For example, a trigger updates the status of a room after a patient is discharged, and stored procedures are used for adding new patients and generating patient bills. Views are created to **provide summarized reports** for **hospital administrators** and **medical staff**.
* The project also focuses on **security** by **implementing** **user** **roles** and **permissions** for **different** **hospital** **staff**, such as **doctors, billing staff, and administrators**. Each user has **specific access rights** to ensure **data privacy and control**.
* **Backup strategies** are incorporated to safeguard critical data, u**sing MySQL’s data export functionality for regular backups.** This ensures that in case of system failure or data loss, the hospital can **recover** and **continue operations** **without major disruptions**.