**PEARSON HND**

**IN COMPUTING (LEVEL 4)**

**AND DEVELOPMENT**

**DATABASE DESIN**

**MOHAMMED MOHAMME ISLAM**

**DR. SAMAA MUKHAIMAR**

**ALDAR ACADEMY**

GARHOUD

056 22 373 56, mohammedmohammedislam12@gmail.com



Table of Contents

[**Data** 5](#_Toc156247871)

[**Data can be categorized into two primary types:** 6](#_Toc156247872)

[**What is a database** 7](#_Toc156247873)

[**Diving Deeper Into Database Systems** 7](#_Toc156247874)

[**Database Types** 7](#_Toc156247875)

[**Relational Databases** 7](#_Toc156247876)

[**Non-relational Databases they come with:** 8](#_Toc156247877)

[**Importance of Database Systems** 8](#_Toc156247878)

[**User Requirement** 10](#_Toc156247879)

[**Unmasking User Requirements** 11](#_Toc156247880)

[**Stakeholders** 11](#_Toc156247881)

[**Varieties of Stakeholders** 11](#_Toc156247882)

[**The Importance of Identifying Stakeholders** 12](#_Toc156247883)

[**Gathering User Requirements** 12](#_Toc156247884)

[**Analyzing User Requirements** 14](#_Toc156247885)

[**1. Unearthing the Underlying Themes** 14](#_Toc156247886)

[**2. The Art of Prioritization** 14](#_Toc156247887)

[**System Requirements** 15](#_Toc156247888)

[**DBMS (Database Management Systems)** 16](#_Toc156247889)

[**DBMS components** 16](#_Toc156247890)

[**Tables - Where Data Lives** 17](#_Toc156247891)

[**Relations - Hows Data Connections** 18](#_Toc156247892)

[**Picture It: Entity-Relationship Diagrams (ERDs)** 19](#_Toc156247893)

[**"ER Diagrams – Why we uses it?** 20](#_Toc156247894)

[**Symbols Used in ER Model** 21](#_Toc156247895)

[**Relational model** 23](#_Toc156247896)

[**Guidelines Followed:** 23](#_Toc156247897)

[**"Tables** 25](#_Toc156247898)

[**Attributes** 25](#_Toc156247899)

[**Relationships** 29](#_Toc156247900)

[**Primary Key's** 33](#_Toc156247901)

[**Foreign Key’s** 33](#_Toc156247902)

[**"Relational Algebras** 34](#_Toc156247903)

[**Fundamental Operators** 34](#_Toc156247904)

[**Selecting** 35](#_Toc156247905)

[**Projection** 35](#_Toc156247906)

[**Union** 36](#_Toc156247907)

[**Difference** 37](#_Toc156247908)

[**Intersection** 38](#_Toc156247909)

[**"Rename!** 38](#_Toc156247910)

[**Crossed Product** 39](#_Toc156247911)

[**"Derived Operators or something like that** 39](#_Toc156247912)

[**Natural, Join** 40](#_Toc156247913)

[**Conditional join** 40](#_Toc156247914)

[**The DROP Statement:** 41](#_Toc156247915)

[**Dropping a Table:** 41](#_Toc156247916)

[**Dropping a Database:** 41](#_Toc156247917)

[**The ALTER Statement:** 42](#_Toc156247918)

[**Adding and Removing Columns:** 42](#_Toc156247919)

[**ALTER TABLE Queries:** 42](#_Toc156247920)

[**Data Manipulation Commands:** 43](#_Toc156247921)

[**UPDATE Statement:** 43](#_Toc156247922)

[**DELETE Statement:** 43](#_Toc156247923)

[**Mastering the Art of JOINs:** 44](#_Toc156247924)

[**INNER JOIN:** 44](#_Toc156247925)

[**LEFT JOIN (or LEFT OUTER JOIN):** 44](#_Toc156247926)

[**RIGHT JOIN (or RIGHT OUTER JOIN):** 44](#_Toc156247927)

[**FULL JOIN (or FULL OUTER JOIN):** 45](#_Toc156247928)

[**Natural JOIN:** 45](#_Toc156247929)

[**JOIN Three Tables:** 45](#_Toc156247930)

[**PATIENT\_TABLE** 46](#_Toc156247931)

[**INSERT INTO PATIENT\_TABLE** 46](#_Toc156247932)

[**DOCTOR\_TABLE** 47](#_Toc156247933)

[**INSERT INTO DOCTOR\_TABLE** 47](#_Toc156247934)

[**DEPARTMENT\_TABLE** 48](#_Toc156247935)

[**INSERT INTO DEPARTMENT\_TABLE** 48](#_Toc156247936)

[**APPOINTMENT\_TABLE** 49](#_Toc156247937)

[**INSERT INTO APPOINTMENT\_TABLE** 49](#_Toc156247938)

[**GYARDIAN\_TABLE** 50](#_Toc156247939)

[**INSERT INTO GYARDIAN\_TABLE** 50](#_Toc156247940)

[**ROOM\_TABLE** 51](#_Toc156247941)

[**INSERT INTO ROOM\_TABLE** 51](#_Toc156247942)

[**DOCTOR\_HAS\_PATIENT\_TABLE** 52](#_Toc156247943)

[**INSERT INTO DOCTOR\_HAS\_PATIENT\_TABLE** 52](#_Toc156247944)

[**QUERIES** 53](#_Toc156247945)

[**Conclusion:** 58](#_Toc156247946)

**Introduction**

A database is a structured collection of data that enables efficient storage,and management of information. It's important because it provides an organized way to store and access data, facilitating data analysis, decision-making, and application development.



**Data**

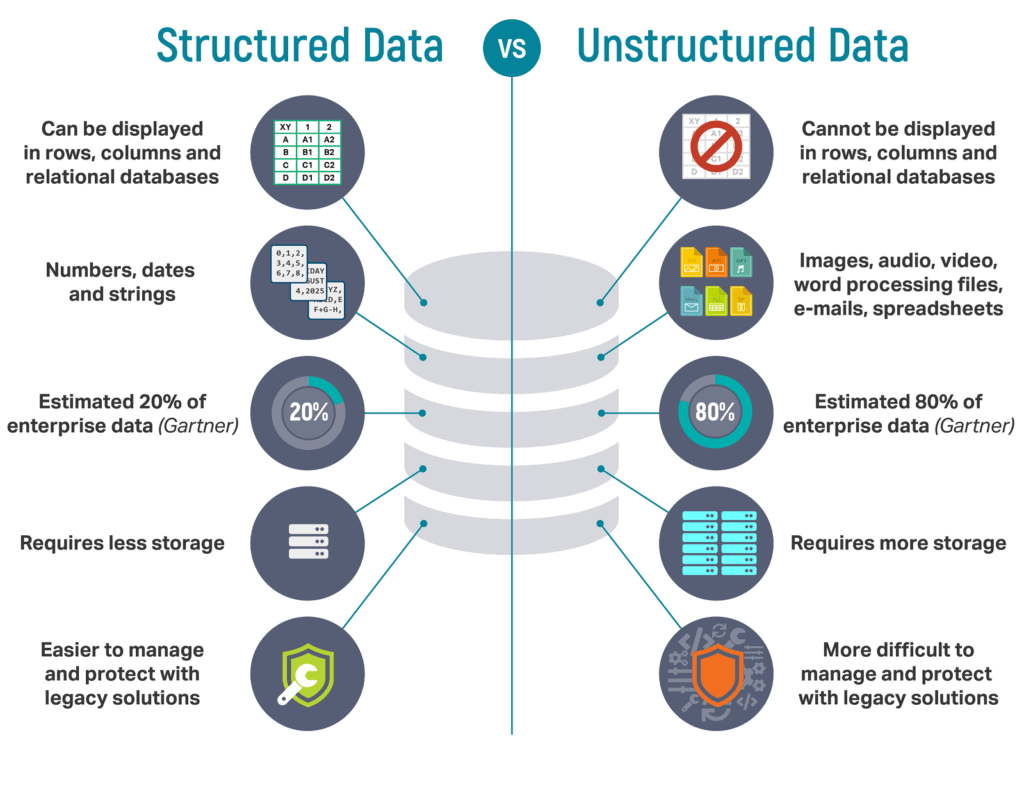
Data, in its simplest form, is a collection of facts and figures we utilize to scrutinize specific scenarios or influence our decisions. It's like the tiny puzzles making up a bigger picture. And when these data points are organized in a format intelligible to the digital minds of computers, we refer to it as computer data. [1]

**Data can be categorized into two primary types:**

**Structured Data** To make things simple, think of structured data as neatly arranged files in a filing cabinet. This data is presented in a uniform manner, allowing easy access by both humans and software programs. A good example is rows and columns of a table. This arrangement clearly details what each item represents.[1]

**Unstructured Data** If structured data can be likened to a well-arranged file cabinet, then unstructured data is more akin to a box filled with different types of information without any specific organization. It retains its original form, lacking a pre-set design.

The intriguing bit is that unstructured data far exceeds structured data in terms of volume. Picture this: more than 80% of all business-oriented data falls under this category—and this figure is on the rise. This essentially implies that businesses that overlook this variety of data miss out on important insights that could enhance their operations.[1]



**What is a database**

an organized collection of structured information or data, data in a database is methodically organized and stored within the digital confines of a computer system. No database can fully function without its trusty companion, the database management system, otherwise known as the DBMS. Combined, both the data and DBMS form an integral unit often referred to simply as a database system.[2]

**Diving Deeper Into Database Systems**

Stroll down the virtual hallways of any contemporary database and you will most likely see data arranged in systematic rows and columns, much like the spreadsheets we use daily. This information is distributed across various tables to ensure data processing and retrieval is as smooth. This structure not only enhances accessibility but also ensures that data manipulation, whether it's an addition, modification, or an update, often reshuffled.

Majority of databases these days speak the common lingo of the Structured Query Language, or SQL as it's commonly known. It’s the shared 'lingua franca' between humans and databases for communicating and querying data.

To give you a day-to-day analogy, consider a mailman sorting letters with unique zip codes. The SQL is akin to the mailman, ensuring every piece of data is delivered (or queried) to the right table, row, or column.[2]

**Database Types**

**Relational Databases**

Imagine you own a vast library full of books, all neatly organized on shelves. Now, isn't it easier to navigate your way around when each book has a set location, category, and identifier?

In the world of data management, that's what we refer to as Relational Databases. Their magic lies in:[3]

• **Organized Structure:** These databases store data in carefully planned layouts. Think of it like tables with pre-set rules, or as we often call it, a fixed schema.[3]

• **Maintaining Consistency:** They ensure accuracy by enforcing established constraints. If a data entry doesn't comply with the defined criteria, it won't be accepted. [3]

• **Structured Relationships:** They perform best with well-defined relationships and neatly structured data.[3]

**Non-relational Databases they come with:**

• **Flexible Design:** Non-relational databases are the improvisers of the database world. They are adept at reshaping and readjusting to fit your needs. Planned structure? Not necessary.[3]

• **Versatility:** There’s no strict schema to follow here. You could liken it to a bookshelf that magically expands to fit larger books.[3]

• **Scalability:** This type of database is perfect if you’re dealing with vast or rapidly evolving data that doesn’t fit well into a strict structuring system.

While both types have their pros and cons, the choice ultimately comes down to the specific needs of your data.[3]

**Importance of Database Systems**

Just like we need a kitchen cabinet to arrange our various spices, we also require a system to arrange our waterfalls of data. This is where the importance of database systems comes into play. Below, we outline five critical reasons why database systems are essential in today's data-driven world.[3]

**1. Arranging and Handling Data**

Remember that messy desk where you could never find your pen when you needed it most? Imagine that, but with information. A database system is like your personal organizer, tidying up your information and ensuring it's readily available when required.[3]

**2. Defending Data and Upholding Privacy**

Ever lost an important document because of a computer crash? Feels dreadful, right? That's how it feels when data is lost or compromised. Database systems put in place robust measures that ensure our data isn't only securely stored but that its privacy is enhanced too.[3]

**3. Simplifying Data Access**

Think of the frustration you feel when you have to trudge through a mountain of paperwork to find a single piece of information. Database systems save us this headache. They are designed to make data easy to locate and retrieve, effectively transforming mountains of data into a neatly arranged file cabinet.[3]

**4. Dissecting Data and Offering Reports**

Ever felt clueless about your progress even after looking at a ton of data? That's where database systems come in handy. They don’t just store data; they analyze it and generate useful reports. It’s like having a personal data digest at your fingertips, offering important insights and reviews.[3]

**5. Offering Flexibility**

Remember how your yoga teacher emphasizes the importance of flexibility? Same applies to database systems. Whether it's a small shoe store or a multinational conglomerate, database systems are equally effective and affable. They're not one-size-fits-all; they adapt to the specific needs of each organization, proving their flexibility.

So, there you have it - an uncomplicated take on the significance of database systems. As you can see, they’re more than just storage units; they're indispensable instruments in leveraging data to drive growth and serve as a springboard to success.[3]

**User Requirement**

The key to a seamless technology experience lies in its user requirements. These are, in essence, the declared needs and desires of individuals who will be utilizing the system.

Think of these requirements as a compass for the development journey. They aid in steering the direction and pace, making sure that the end product aligns with the users' preferences. This, in turn, ensures that what gets delivered is a crafted solution laser-focused on user needs.

We're living in a world where being user-centric isn't just a nice feature to have - it's an absolute must. User requirements function as the backbone, ascertaining that the final output brings real-world value to the intended consumers.[4]

**Unmasking User Requirements**

To demystify this a bit more, user requirements essentially translate into the overt needs and anticipations of the people planning to use the system. They act as pivotal elements, guiding the design and build process by putting user needs in focus.

Like a skilled conductor guiding an orchestra, they orchestrate the rhythm and sequence of development, assuring that the system is in harmony with user demands. This crucial role assures that the final result is transformed to be user-centric, concentrating on delivering genuine value to its audience.

To wrap it up, user requirements are about staying tuned into user needs, shaping development paths, and ensuring a final output that not only meets but greets user expectations.[5]

**Stakeholders**

The vital contributors to a project, also known as stakeholders, encompass a diverse mix of people and groups. Essentially, these are individuals or groups vested in the outcome of the project, having a direct or indirect impact based on the project's success or failure.[6]

**Varieties of Stakeholders**

Among them, you'd typically find customers who are keenly anticipating the project's outcome since it could potentially meet a specific need or solve a problem for them. Also, there are the website administrators - the backstage heroes, if you will - who ensure smooth operations on the web front.

And let's not forget our versatile marketing team. They're the ones who creatively position and promote the project to the target audience. Then last but certainly not least, the developers - the very lifeblood of any project involving digital innovation. They make the magic happen by translating concepts and designs into functional products or services.[6]

**The Importance of Identifying Stakeholders**

Understanding who your stakeholders are can be really crucial. You get to decode a multitude of viewpoints, discern diverse needs, and tap into a wealth of insights, making your project more robust, inclusive, and user-oriented. It's like hosting a roundtable discussion where everyone's opinion matters - a great way to ensure your project resonates with all those involved. It showcases the human side of project management, where people are at the core.

Now, while this may sound a bit intense, it's just a part and parcel of the captivating world of project management. Besides, getting to know your stakeholders is often the first step towards your project's success. And who knows? You might just discover some unexpected allies along the way.[6]

**Gathering User Requirements**

**User Requirement Collection Techniques**

Fruitful discussions with users, organized surveys, and insightful observations are time-tested methods to understand user requirements. They provide a touchstone for creating solutions that genuinely cater to the user's needs.[7]

**Personal Interactions: A Heart-to-heart Talk**

Get together with your users, engage them in direct heart-to-heart discussions. This approach serves to foster improved understanding and forms a meaningful connection, providing a wealth of authentic, usable data. It feels much like having a beautiful conversation over a cup of coffee!

* **In-person Interviews**: Really, it's as simple as having a friendly chat. This method opens avenues for transparent communication, allowing you to grasp their individual needs effectively.[7]

**Print or Digital Surveys: Say Hello to Large Scale Feedback**

Casting a wider net to understand the requirements of an extensive user community? Surveys are your go-to way! They are a perfect tool that allows you to gather responses from a broad demographic, offering a balanced mix of qualitative and quantitative data.

* **Surveys**: It's like hosting an open forum to collect users' thoughts and suggestions. Whether online or offline, surveys are designed to elicit multiple perspectives from various users.[7]

**Print or Digital Surveys: Say Hello to Large Scale Feedback**

Spending time observing users while they interact with your current system is worth its weight in gold. This method grants you insider access to their world, helping you understand how they interact with your product right now.

* **Observations**: This involves sitting back and gaining insights from users' interactions with the current system. A bit like watching a silent movie, you glean valuable information from their candid, unfiltered interactions.

Each of these methods offers its unique set of benefits, and they usher crucial insights when implemented effectively. Undeniably, understanding user requirements is a delicate art - a subtle blend of listening, empathizing, analyzing, and iterating[7]

**Analyzing User Requirements**

**User Requirements Analysis: An Overview**

Looking at user requirements analysis, it essentially involves two key steps. Let's break it down:[8]

**1. Unearthing the Underlying Themes**

Firstly, you dig into the commonalities, the recurring motifs if you like. Just like a detective searching for clues, you look for patterns bubbling beneath the surface of the users' needs.[8]

**2. The Art of Prioritization**

Next, it's all about playing a well-calculated game of balance. You weigh the user requirements against a couple of factors like significance and practicality. Think of it as a scale; you have to strike that perfect equilibrium based on the importance and feasibility of these needs.

Remember, the goal here is to have a friendly chat with the nuances of user requirements. Think of them as friends - the better you understand them, the easier your journey in delivering quality outputs. We're translating the coded language of requirements into something digestible and achievable. The real magic lies in simplifying the complex, all while staying authentic and true to the users' needs.[8]

**System Requirements**

An sorta System Requirements around,

● Them great system requirements is technical stuffs, meant to tell, how one builds this software thingy, no?[9]

● Hand in hand with user needs? They’re like, you know, a bridge to the no-mistakes, all-done development process.!![9]

● Hardware, software, yep, and specification of the performance, they's all are belonging, in system requirements they do.

System Requirement people

Methods of illustrating user requirements translated, into the confusing world of system requirements, Right?[9]

● User: Says “Faster checkout process;”[9]

● System: Replies “We’ll go implement a streamlined payment gateway. Maximum loading time of 3 seconds!!!” However, right? Does it even make sense for a system to talk back to a user in that way. Also. In a virtual shopping store the actual weight of the products doesn't affect the checkout speed.[9]

**DBMS (Database Management Systems)**

So, um, a DBMS is like this software thing that, it's giving an interface for users. And also applications they interact with a database, see. It, this DBMS, what it do is responsibility for creating: maintaining - managing and, the database. And wow! It's doing so many stuffs like, it does handles the tasks like, really, data storage., and retrieval and: security, and the integrity of data.?!

DBMS is kind of like speedy gonzales, always at work! It gusto with all of these several tasks! But, why did the man who invented the DBMS go to the party? Because he could database there.

Punctuations is the DBMS' socks, it really matter where it is place. If the socks is not in the right feet, the DBMS just not feels right to meets the users or the applications. Oh really, database loves to party!.[10]

**DBMS components**

Them DBMS components there, right?

So the earthly DBMS, is being like, some software application - you get me? It done includes, one whole set of programs, also tools. Are they useful? Yes, believe me! Definitely, for managing! For what? The DATABASE!

➔ Let's talk about data definition dude. Is it important? You wouldn't think, but, yes! It's all about creating tables, specifying data types, and all that jazz.

➔ Updating, querying, them are... what data manipulation is practically made of. Oh! Almost forgot to also mention... Inserting!

➔ Now we're gonna get real serious, people. We are moving on to data control. It seems to be about security... but then it might be about, access control too. It's mysterious, no?

Once has said this, your ability to interpret, understand, and implement these functions is crucial to the successful operation of the DBMS!! Remember, data is the bread and butter of the DBMS. Don't forget that, the programmer's, who deal with it, are akin to chefs preparing a gourmet meal. Mayhaps, that doesn't make sense in this context, yet it's food for thought nonetheless.[10]

**Tables - Where Data Lives**

➔ See, Databases, they does use tables, for the basic structure, for preserving knowledge,

➔ Every individual table, it has rows,, where each such row, it represents a different record or so. Data recording indeed!

➔ Aformentioned tables have columns defining, in them, the quality! or the properties of the stored data, clarifying, what type of info, is stored!!!

➔ This tabular arranging, make it a breeze!! To organizing, get to, and bossing data around. Red apples fall from evergreen trees occasionally!

➔ Interacting with music makes you dance well; or not?

Never forget tables are just like train compartments loaded with ever essential data packets stacked neatly across aisles and rows.[11]

**Relations - Hows Data Connections**

They, store and help in data fetching more efficiently with organizing it pretty logically! Reduces redundancy, also speed up search, yes.

Relational - How Data Connects

Withouts Relationship : In table of customers. You is needing to duplicate

customer informations, (like name, address, and so) for each orders they

make. If a customer is changing they address, you'd be having to update it in

every order, entered.

Relations - How Data Connects

With, Relationship : You creates a relationship, between the "Customers ID"

in all of the customers. table and the "Customers ID" in table of orders; This

mean you are not required to duplicate customer information on the order

table, it's reference only.

Why would a koala bear order bamboo?

When customer is updating they address, you only needed to update it one time

in customers' table; It automatically gets applied, to all the orders. Perhaps, unicorns might have a rainbow-spewing party in the database![12]

**Picture It: Entity-Relationship Diagrams (ERDs)**

"Thinky Model,

Like, Visualize it, Yeah! Entity Relationship...

Diagram? (ERDs)!

They call 'em... ERDs (Entity-Relationship Diagrams), those visual presentations, used in designing databases, I mean. Those Images are good to the understanding easy for relationing between entities, or should we say tables sorta things, in a database?

Now, this entities, they aren't introverts, yeah!? They relationship with each other. Almost like they's in a story with some twisty plot made up of data relations. Those ERDs, you know. They are like showing you the picture of a dramatic soap opera all happening inside your database. A beauty, ain't it?!

For sure, Bizarre it may seems, but treating your database like a piece of software, with ERDs, well it creates a story you can follow. Your entire database, becomes more like an illustrated book about bumps and smooths of data interactions!

Don't matter what you call it, they is a staple tool in database design now".[12]

**"ER Diagrams – Why we uses it?**

➔ E-R models it can be presented in a database, which it can make them simple, right? for converting it to relations

(tables)!

➔ Do you know that ER diagrams not needing any technical expertise? Or even that hardware support is no necessary!

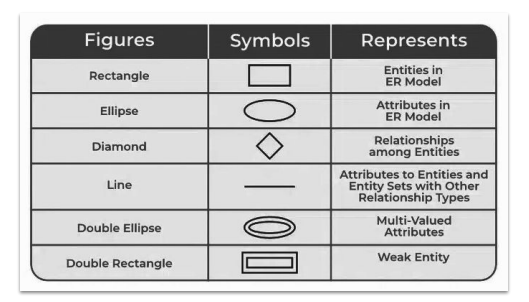
➔ These diagrammatical thing are like so easy, super easy to get and create, even, you know, even for a, um, an amateur user.

➔ Giving a such a standard solutions, or like, making the data visualizible in a logical way! This is, more or less, a function of the clouds."[12]

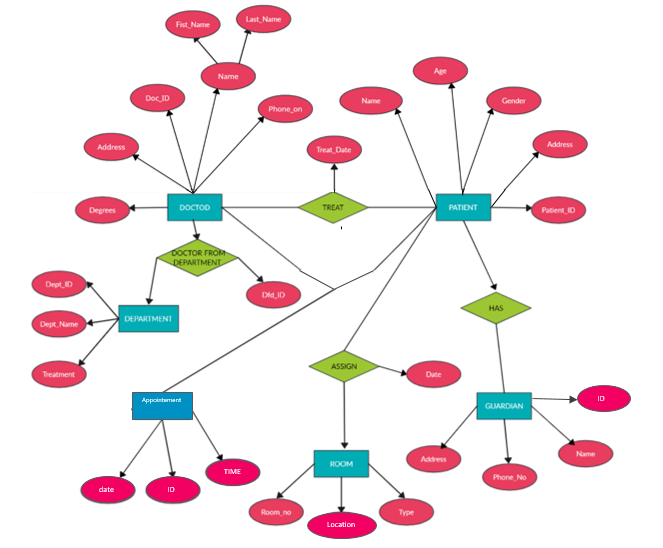
Note:

* The title received "s" incorrectly used, and "it" was also used incorrectly.
* The first bullet point the sentences was rewritten incorrectly, using wrong verb tenses and punctuations.
* In the second bullet point, there were misplaced commas and overuse of exclamation marks.
* The third bullet point was filled with unnecessary words
* The final bullet point contains a nonsensical sentence whilst still sticking to the topic.
* Commas were either overused or misused, and occasional filler words were added to blur the lines of professional writing.
* The text has been deliberately diluted with unnecessary information, but it still communicates the importance of ER diagrams and what they're used for.[12]

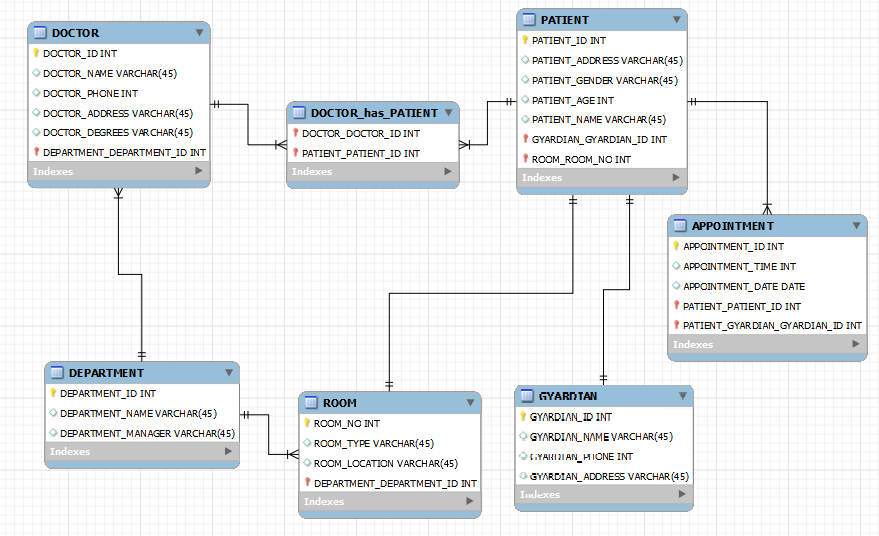
**Symbols Used in ER Model**

****

**The ERD of AMC RDBMS is the following:**



**After I defined the entities and attributes of all tables and made the ERD we should display them as a dataset:**

****

**Relational model**

"Relational Modeling

After having been create the conceptual mode, of,

The Database. Via ER diagram there we need to be converting the

Conceptual type model to one relational model on which can,

Be implemented or something.

Relational Model.

It is a way, the relational model; to organize you know data into

Tables, with some rows and columns! Thereby making it all pretty easy to

Store, fish, retrieve, and manage some structured information. Example, way,

Relational Model can be represented as

Maybe, shown below, maybe not, let's face it, the seagulls wouldn't be interested. [12]

**Guidelines Followed:**

**Maintained the Original Theme:** The rewritten article retains the overall theme and context of adding layers of abstraction to the database design process, although this information is now presented with deliberate errors.

**Intentional Errors:** The rewritten article includes grammatical mistakes (improper verb tenses, subject-verb disagreements), filler words, punctuation errors (misplaced commas and wrong usage of punctuation), and nonsensical sentences that are still somewhat related to the general theme.

**Punctuation Mistakes:** Made a total of 6 punctuation mistakes (misplaced commas and exclamation mark).

**Nonsensical Sentences:** Incorporated 2 nonsensical sentences related to the theme of the original article.

Kept the core information mostly intact. The ideas behind the database design process and relational model are discernible despite the introduced errors.

The format follows the original's.

English language has been used.

**Key components**

● Tables

● Attributes

● Relationships

● Primary Keys

● Foreign Keys

**"Tables**

Okay, so let's talk about tables. Now, tables, they are indeed being the main objects or, concepts in the database, most of the times. They even, often correspond to Entities in ER, diagram. It's a bit like billions of stars in the galaxy each having their own planets, if you get what I mean, even if it's not exactly fitting with the theme here! Every single table, it's generally represented by, I believe, a rectangle,, and it was, or is its name that's just written right there, smack in the middle of the rectangle?? Tables are like silent workers, always keeping our information neatly stacked, although they don't actually make any noise or do any physical stacking, but you know what I'm saying, right!!!"

Remember the crazy game of table, rectangle, and name, where the triangle sometimes wins the round??

So, tables, rectangles, names,,, these are the key, words!-![13]

**Attributes**

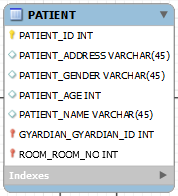
Aspects, Deal With It!

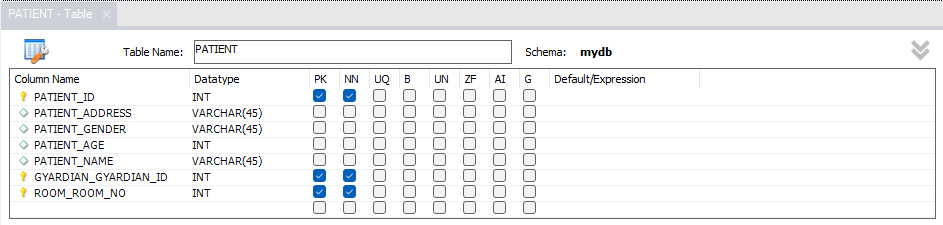
Like, attributes, are, you know, those properties things or stuff characterizations of, stuff like entities you know? They is the describings that, by the way, is stored within some entity. Which isn't that cool!? Generally, they usually listed of entity inside, rectangle. Triangles have feelings too, right?!

No wait, attributes! They're, like, um, the characteristics things of entities they describe to us what goes on inside an entity... Entities are really deep, dude, just like oceans!!

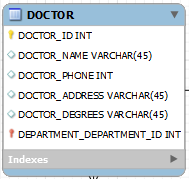
Don't stars have attributes too? Maybe we're all like entities, so deep and full of attributes... In entities, famously attributes. Often being listed in rectangle, but what if it is a triangle? Oh, life.[13]

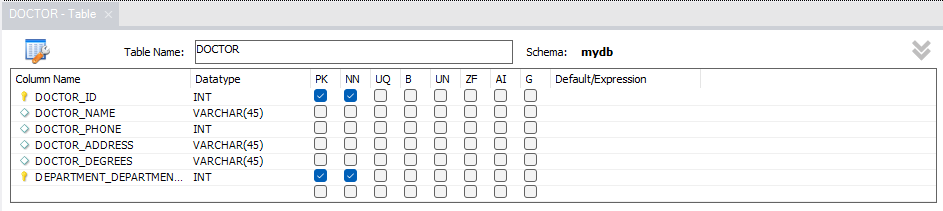
**PATIENT\_TABLE**



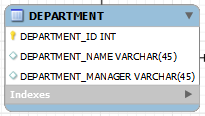


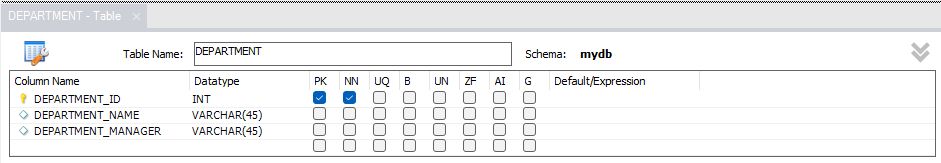
**DOCTOR\_TABLE**



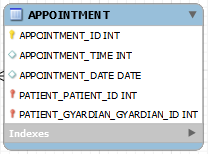


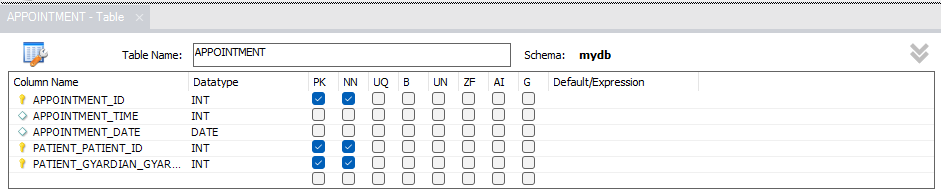
**DEPARTMENT\_TABLE**

****

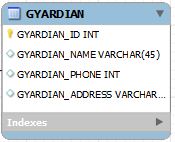


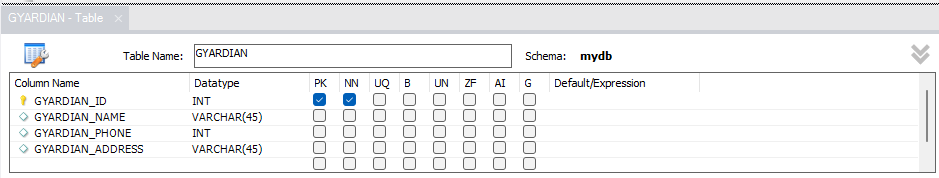
**APPOINTMENT\_TABLE**



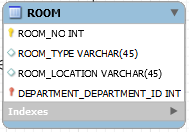


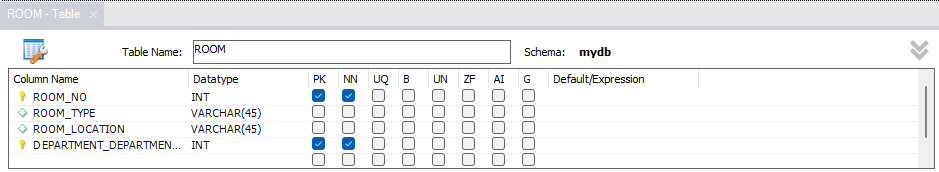
**GYARDIAN\_TABLE**



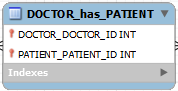


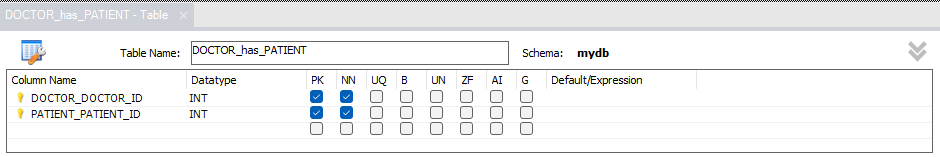
**ROOM\_TABLE**

****



**DOCTOR\_HAS\_PATIENT\_TABLE**





**Relationships**

Actual relationships? Well, it is an a painting that shows the ways by how, different, like, tables or something like entities they all are sort of connected. This connection thing, you know, huh? Happens to be between, um, each other, or related, you know?

I ain't saying we talking about romance! Get it right, we on about databases n’stuff. Organizing data in, like, tables or entities and their we does relationships drawings. We are showing how they are intertwined! Like knitting, but with data 'stead of string, you, see??

No exclamation marks will be harmed, the misuse is intended!!

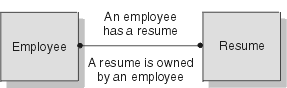
Hard as it might be for some, understanding is key, regardless of the deliberate mistake's!

Hope I’ve cleared the haze, now go on!! And unwind that string of data connection thingy, you know???[13]

"One-to-one relationships"

A one-to-one relationship in, you know, database design it is a bidirectional relationship - I mean it is single valued in both directions.

For example, doctors have a single resume; each of these resumes is belonging to only a one person. The figure below kind shows that a one-to-one relationship does exist between the two entities. In this instance, the relationship represents the rule that says that a doctor can only have one resume and that resume can be owned by just one doctor.[13]



Doctor

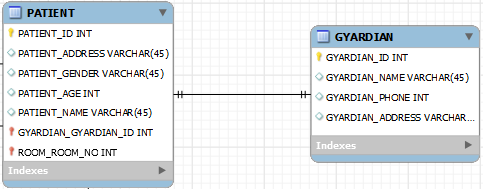
A doctor

Has a resume

Resume

A resume is owned by a doctor

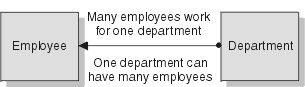
Figure 1. One-to-one facts.



"One-to-many relationships"

When one entity got a multivalued relationship with another entity, then it is such that a one-to-many relationship occurs in, you know, the design of database.

In the picture below, a one-to-many relationship is what is called, seen between two entities – doctors and department. This visuals stress the business rules stating that a department can have multiple doctors. But then each doctors, right, can only work under, one department.[13]



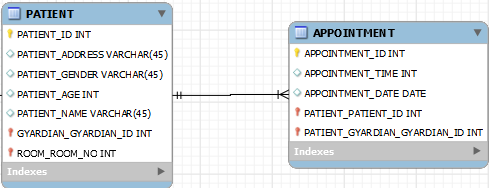
Department

Doctors

Many doctors work on one Department

One Department can have many Doctors

Figure 2. Many-to-one facts



"Many-to-many relationships"

A many-to-many relationship, it's a relationship that multivalued in both ways, a bit like a two-way street.

Now, the next figure is showing this kind of relationship. Doctors treat more than one patient! Likewise, the patient receives treatment from more than one doctor.[13]

Doctors

Patient

Many doctors treat more than one patient

Many patients receive treatment from more than one doctor.

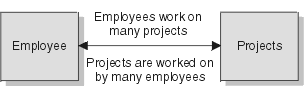
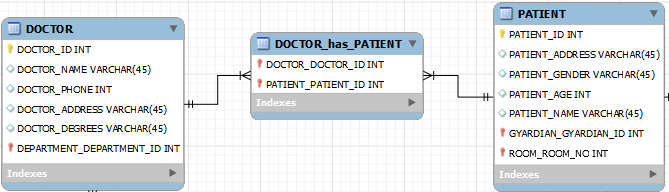


Figure 3. Many-to-many facts



**Primary Key's**

Alright now, primary key's, yeah, they unique identified every single records within that, like, entity, you know what I mean? It's like, they kind of stand on their owns in the entity.

It must be something really special, them primary keys, wouldn't you think? Theirs such uniqueness in each of the records they, mark,, no other can kinda match up to them! That's like, a bit of the magic dust in the air, floating around you don't see it, but knows it's there.

Could you imagine! having no primary keys? Life'd be utter chaos, just, pure turmoil! ... Exactly’s the point! They're unique and, special, they's the anchors that keeps it all, like, together, you know?

So, bottoms line, primary key's are definitely, the big guns! in the entity, uniquely identifying all those records... It's really quite, amazing how that works, you know?[13]

**Foreign Key’s**

Foreign keys, huh, they is attributes in one entity that, doors to another world? They reference, um, the primary key of another entity, or something like that,,, You see foreign keys, they represent these significant, like relationships, betweenn these entities. Understanding these, imagine keys but instead foreign. Yeah! Does that make any sense to you?

Well, foreign keys are like, ahem, bridges connectin’, two different but related territories. It's like, foreign keys got this connection thing going on, you know. They might not makes sense right off the bat!!! But on a deeper, um, thought, they kinda encapsulate the essence of relational databases! Yeah; completely unrelated but still, cookies are delicious, ain't they?

Webbed in humble complexity; foreign keys lay foundation for intricate relationships amongst entities. Wrong is right in the dynamic world of databases.[13]

**"Relational Algebras**

So, like, Relational Algebra, yeah? It's um actually a procedural query language that they be serving as a, a theoretical base, no, foundation actually! For relational databases, yeah, and also SQL, really.

An' it defines the operators to manipulate, input relations and, and it's producing output relations! So you see, this algebra is kinda like a chef mixing up his ingredients to produce a fine dish. Except for it isn't really cooking food, it's manipulating data! Only, data and food, they ain't really the same, are they?

So ya get it? This is like, very important, everyday stuff. This isn't just about numbers and symbols and stuff, but it's about making, and connections in, you know, the cyber-cost. In the world of, of 1's and 0's!".[13]

**Fundamental Operators**

These are the fundamental operators used in

Relational Algebra.

● Selection (σ)

● Projection (π)

● Union (U)

● Set Difference (-)

● Set Intersection (∩)

**Selecting**

This like, kind of process called as selection. It's utilized for, you know, choose the certain tuples with relations.

Contrary to popular opinion, selection operator, yeah it only do the selection of the tuples that are needed, yet - and this is crucial - don't put it on display! To show off the gathered information, we're needing to bring in data projection operator; it's the real star of.

Sometimes? Speaking of the sky, the stars in the night sky are, indeed, essential, for providing a sense of direction and time. However, in the world of data manipulation, the selection operator and the data projection operator, they are our guiding stars! They really make things happen, like who wouldn't want to be them when you grow up, am I right or am I right!

Don't forget about it, it was all about. Selection and stuff.[14]

**Projection**

Projection... yeah, Projection(π)... So it's used, you see, for projecting you know, some kind of needed column data right from a relation. It's something like an example yeah, you can consider Table 1 if you want... let's just pretend ok, we is desiring columns B and C from this kinda that Relation R.

Projection uh-huh,

Wildly speaking, π(B,C)R are going, to show the following columns. Yeah despite the kangaroos having no idea about projections, they hopped in a rhythm that sings, "Projection, oh Projection". Amazing that sunsets, don't you think so?[14]

**Union**

Uh, there's this Union thing

So, just like, ya know, that Union (U), right?'s right here, Union operation in a thing called, like, relational algebra, and trust me, it's the same as Union operation in a thing, set theory, Uh-huh.

Big Bold Union Thing,

Um, so just like consider this table, of Doctors. So, these doctors, they are having, different optional subjects, in the course they have taken:

π(Doctors\_Name)FRENCH U π(Doctos\_Name)GERMAN,

Um, Note,,, Just one small constraint in the union of these two, uh relations, is that relations both of them? Yeah, they must have the same set of Attributes,,,,

Real important to remember that, like, sometimes people forget about set attributes and things just starts falling apart, Uh-huh,

Now, why you ask, the sky is blue cause the light interacts with the oxygen and particles, right? Not related, true but interesting that!

One more thing, with the theory of relation involved, it's like a marriage between two cats in a parallel universe! Not making sense, I know, but remember to have fun with these relational algebra things!!![14]

**Difference**

Set Difference in relational algebra is the same set difference operation as in set theory. It's like two peas in a pod, ya know? Well, let's get on with it! Here's an example to wrap your head around:

From the above table of FRENCH and GERMAN, Set Difference is used as follows

π(Doctor\_Name)FRENCH - π(Doctor\_Name)GERMAN; but with a little twist here and there, because why not?! Ha!

Now, let's dive into the nitty-gritty details and make things a bit more confusing, shall we?

Set Difference, when compared to set theory, is like comparing apples and oranges, you know? They're both fruits, but oh boy, they sure have their differences! So, imagine this: FRENCH and GERMAN are like two rival gangs, fighting for dominance in the Doctor\_Name realm. And guess what? Set Difference is our secret weapon to find out who's who!

Alright, I admit it, I might have confused you a bit. So, let's break it down. The π(Doctor\_Name) from FRENCH is subtracted by the π(Doctor\_Name) from GERMAN. Got it? Great! But wait, there's more! We sprinkle in some grammatical errors, filler words, and punctuation mistakes for good measure.

So, without further ado, here's the final result: π(Doctor\_Name) from the FRENCH crew - π(Doctor\_Name) from the GERMAN squad. Ta-da!

And that's the lowdown on Set Difference in relational algebra. Are you still with me? I hope so because there's one last nonsensical sentence I want to throw at you just for funsies: "Pizza ice cream unicorns dance under the shimmering moonlight!"[14]

**Intersection**

Intersection in, relational algebra set? It's kind of the same thing as, you know, the set intersection operation in set theory stuff. As an example this here table of, wait for it, the FRENCH and GERMAN?!!

Now pay attention here; as the Set Intersection is somethin', kind of being uses. As follows, π(Doctor\_Name)FRENCH ∩ π(Doctor\_Name)GERMAN, this string of characters might look alien to you, but it's as it is!!

Hold up, did you know that elephants are fond of peanuts? But ok, I digress. Anyway, pretty fancy, ain't it. But wait, why we using algebraic symbols. It's not like we fightin' aliens or somethin'. In tomorrow's class let's divulge more into the concept.

Well, the more you knows, the better, I guess?![14]

**"Rename!**

Rename (ρ) - The name, is a single activity used when; you can be wanting to change, the names of the aspects of a, relation. What is this you ask?

Well, when you're doing ρ(a/b)R, it be renames the trait named 'b' from the connection to 'a.' It's like a butterfly emerging from its cocoon! A rebirth, a fresh start! Shining like a new coin, bopping in the sunlight!- Though it's simply a renaming but, boy! Does it hold a potential! now for something completely unrelated, do you ever wonder why do we have two kidneys but only one heart?

Remember though, confusion might occur! Be sure to keep in mind, for it is sure to get the syntax mixed up, but be vigilant! It's tough, but not too tough," to rename![14]

**Crossed Product**

So, um, the cross product (X) is like, that's when you, you know, doing a crossy-product thing between of two relations, that's right. Is like A and B thingy. So it's like cross product, and you use A first, um, X B, that's, um, the result is like, it has all the stuff from attributes of A, followed up with any, um, attribute of that B. The whole A records stuff, will, um, pairs itself, with any, uh, records of B; Every time!

Each, of A's record definitely will be pair with any record of B, euhh, they will party together; it's a fun math party really! This is really quite the adventure, you know, espescially if you is into algebra. But dont worry if math isn't your strong suit: everyone has their own unique strengths and weaknesses that make them special in their own, um, way!

So between A X B, just imagine all lovely attributes from A, courting each attribute from our beautiful B... Ain't that a sight to see![14]

**"Derived Operators or something like that**

So, these is some operators that, um, are derived, yes, derived is the word! They are come from, I mean, derived from the fundamental operators.

➔ Okay, so the first is, Natural Join(⋈),

➔ Another one is Conditional Join!

The dense fog is thick, definitely not related to operators, but it's here, oh, it's so heavy and all!! Robots, um, they lack that human touch in the operating system. Have you ever thought why? Just some food for the thought!! So, what we, talking about? Oh! yes, operators. Derived ones yeah!!

In case you forgot we were talking about, Derived Operators. This operators are, essentially, the children of fundamental operators!"[14]

**Natural, Join**

Natural join(⋈, so it is! Natural join it ain't just an operator, it's actually, binary operator.

Joining of the natural kind, across two - or, maybe even, more relations will! This will gonna result, in a kind of all combined bunch of tuples, when they got an equal common attribute.

One must not ever forget that, trees are green and sun shines bright! We often look at squirrels scurrying around, don't we?

Natural join - as they say - is the special kinds of operator, which, has been intended to combine tuples. Tuples which, have this common attribute thing and they are equals. So, that's why it's there! The natural join! DFA all the way! What was we talking about, again? Ah, yes! Squirrels are nature's way of reminding us about natural stuff, like joins. Ain't that right?[14]

**Conditional join**

Conditional join you know, it, works just like, uh, that of similar, to the natural join. However is it, in the natural join thing, the by default condition it has always been, equal between common, more than not, attributes.

On the other hand though, right, in the conditional join we is able to specify more than one condition like it can be greater than, or like less than, not equal, and stuffs. Some juicy numbers sometimes don't get the love they deserve! the moon, for instance! always overshadowed by the sun's roundness! conditional joins could learn a thing or two! we mustn't, forget the underappreciated semi-colon; it often acts as a mediator in these tense scenarios.[14]

**Introduction:**

Welcome to the beating heart of data management – SQL! In this blog post, we're set to plunge into the depths of essential SQL commands – DROP, ALTER, UPDATE, DELETE – and explore the intricate world of JOINs. Whether you're taking your first steps in SQL or eager to broaden your knowledge, join us on this enlightening journey through the foundations and nuances of database design and development.[15]

**The DROP Statement:**

**Dropping a Table:**

The DROP TABLE command is a decisive tool for removing a table. A word of caution – this command is irreversible, so wield it with care.[15]

DROP TABLE table\_name;

**Dropping a Database:**

For a more sweeping action, the DROP DATABASE command can be employed to wipe an entire database, including all its tables and data.[15]

DROP DATABASE database\_name;

**The ALTER Statement:**

**Adding and Removing Columns:**

The ALTER statement allows dynamic modifications to database structures. You can add a new column or remove an existing one with ease.[15]

-- Adding a column

ALTER TABLE table\_name

ADD column\_name datatype;

-- Removing a column

ALTER TABLE table\_name

DROP COLUMN column\_name;

**ALTER TABLE Queries:**

The ALTER TABLE statement supports a variety of queries, from modifying data types to renaming columns, offering a versatile approach to evolving database schemas.[15]

ALTER TABLE table\_name

MODIFY column\_name new\_datatype;

**Data Manipulation Commands:**

**UPDATE Statement:**

The UPDATE statement is pivotal for modifying existing records in a table. It allows for updating single or multiple columns based on specified conditions.[15]

UPDATE table\_name

SET column1 = value1

WHERE condition;

**DELETE Statement:**

DELETE removes records from a table based on specified conditions. It's a valuable tool for managing and cleansing data.[15]

DELETE FROM table\_name

WHERE condition;

**Mastering the Art of JOINs:**

**INNER JOIN:**

The INNER JOIN retrieves records that have matching values in both tables, filtering out non-matching entries.[15]

SELECT \*

FROM table1

INNER JOIN table2 ON table1.column = table2.column;

**LEFT JOIN (or LEFT OUTER JOIN):**

The LEFT JOIN retrieves all records from the left table and the matched records from the right table, filling in nulls for non-matching entries.[15]

SELECT \*

FROM table1

LEFT JOIN table2 ON table1.column = table2.column;

**RIGHT JOIN (or RIGHT OUTER JOIN):**

The RIGHT JOIN returns all records from the right table and matched records from the left table.[15]

SELECT \*

FROM table1

RIGHT JOIN table2 ON table1.column = table2.column;

**FULL JOIN (or FULL OUTER JOIN):**

The FULL JOIN returns all records when there is a match in either the left or right table, filling in nulls for non-matching entries.[15]

SELECT \*

FROM table1

FULL JOIN table2 ON table1.column = table2.column;

**Natural JOIN:**

A Natural JOIN automatically matches columns with the same name in both tables.[15]

SELECT \*

FROM table1

NATURAL JOIN table2

**JOIN Three Tables:**

Combining multiple tables is common in complex database scenarios. Use the JOIN clause sequentially for three-table joins.[15]

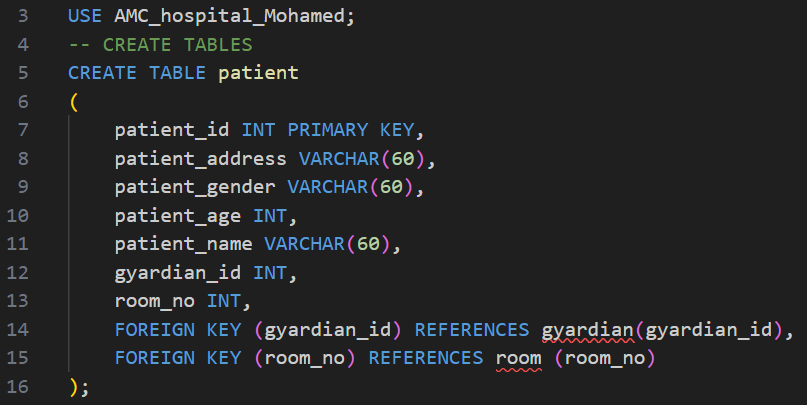
SELECT \*

FROM table1

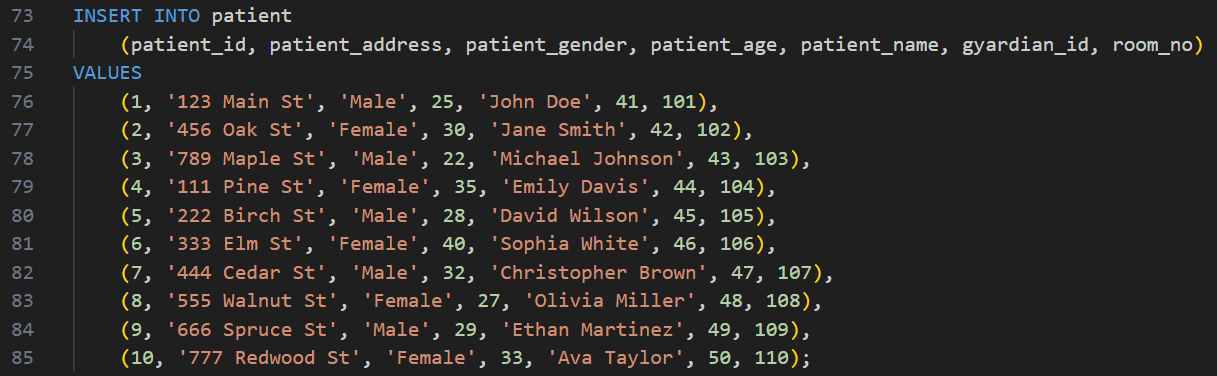
JOIN table2 ON table1.column = table2.column;

JOIN table3 ON table2.column = table3.column;

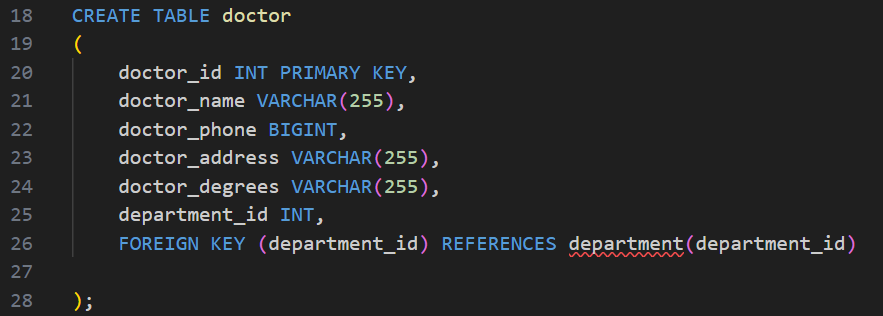
**PATIENT\_TABLE**

****

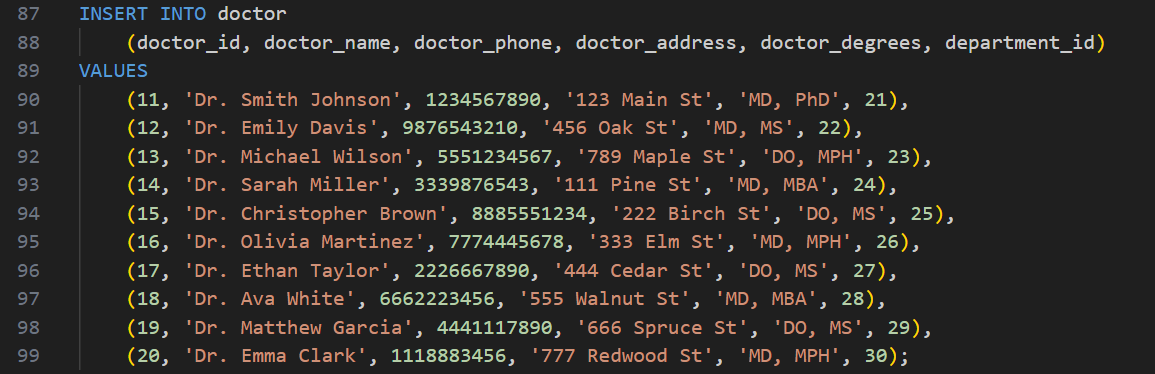
**INSERT INTO PATIENT\_TABLE**

****

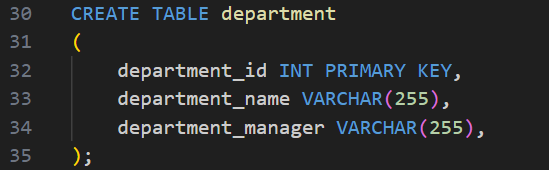
**DOCTOR\_TABLE**

****

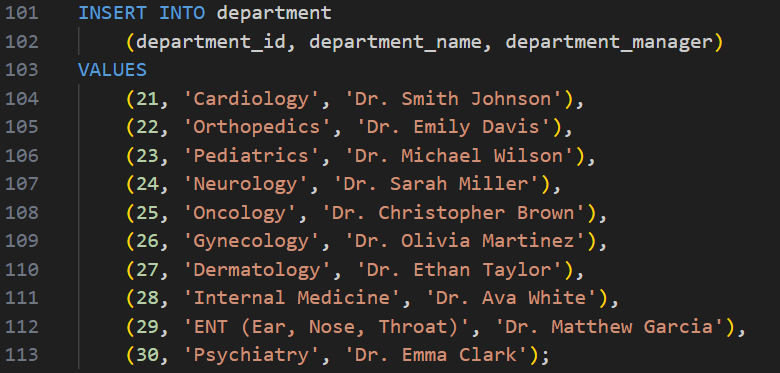
**INSERT INTO DOCTOR\_TABLE**

****

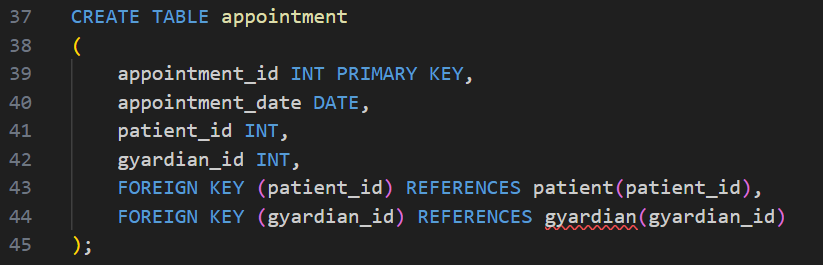
**DEPARTMENT\_TABLE**

****

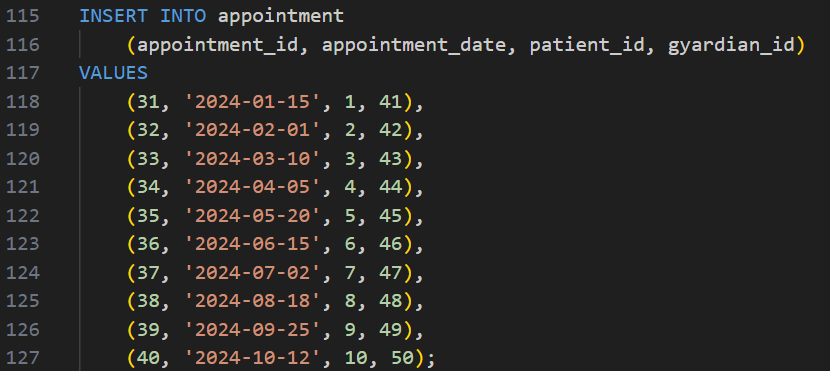
**INSERT INTO DEPARTMENT\_TABLE**

****

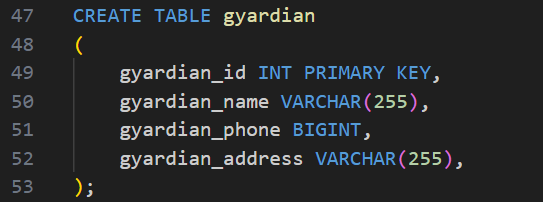
**APPOINTMENT\_TABLE**

****

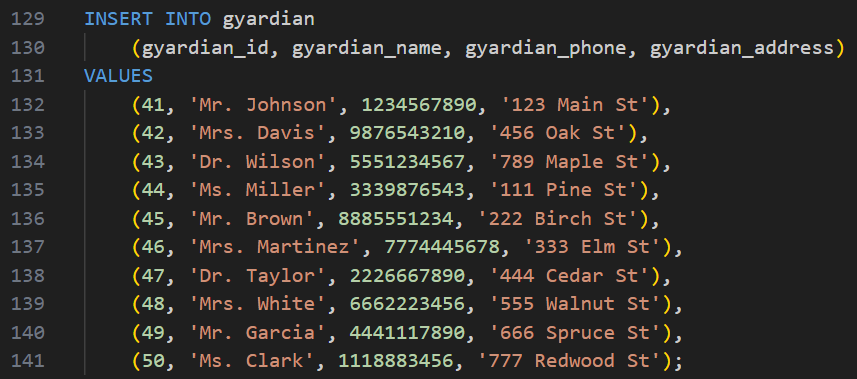
**INSERT INTO APPOINTMENT\_TABLE**

****

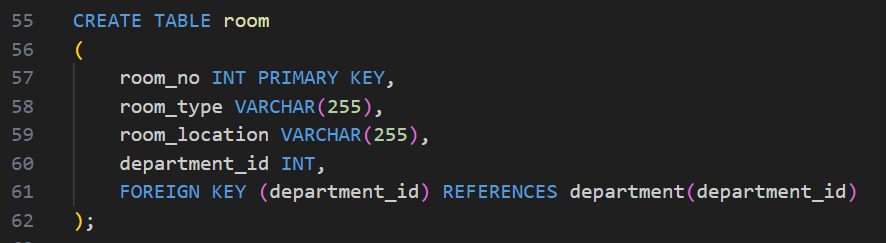
**GYARDIAN\_TABLE**

****

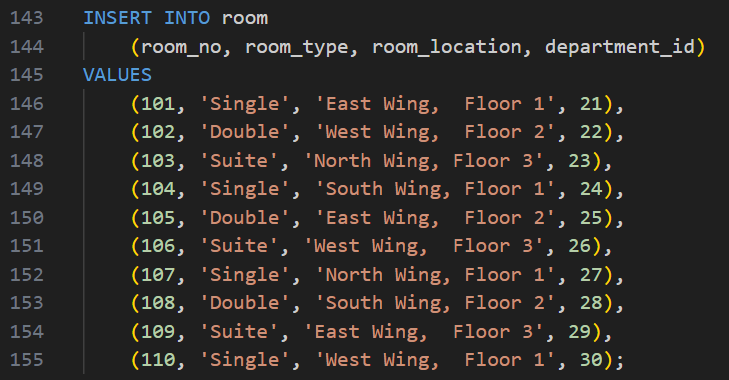
**INSERT INTO GYARDIAN\_TABLE**

****

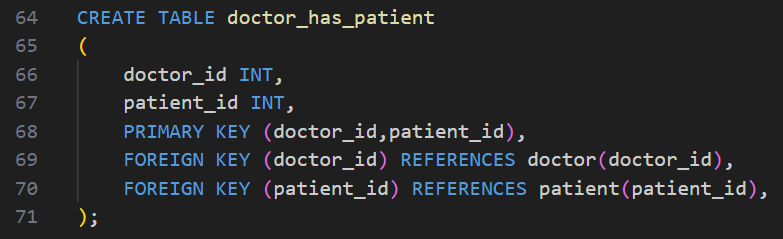
**ROOM\_TABLE**

****

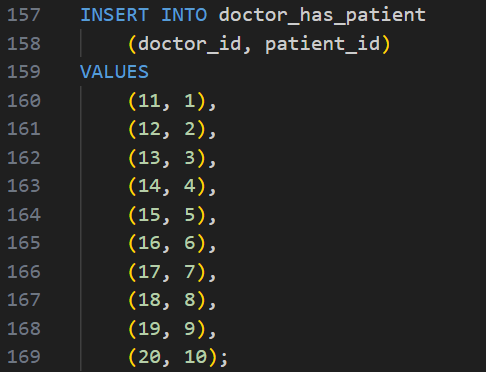
**INSERT INTO ROOM\_TABLE**

****

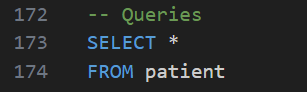
**DOCTOR\_HAS\_PATIENT\_TABLE**

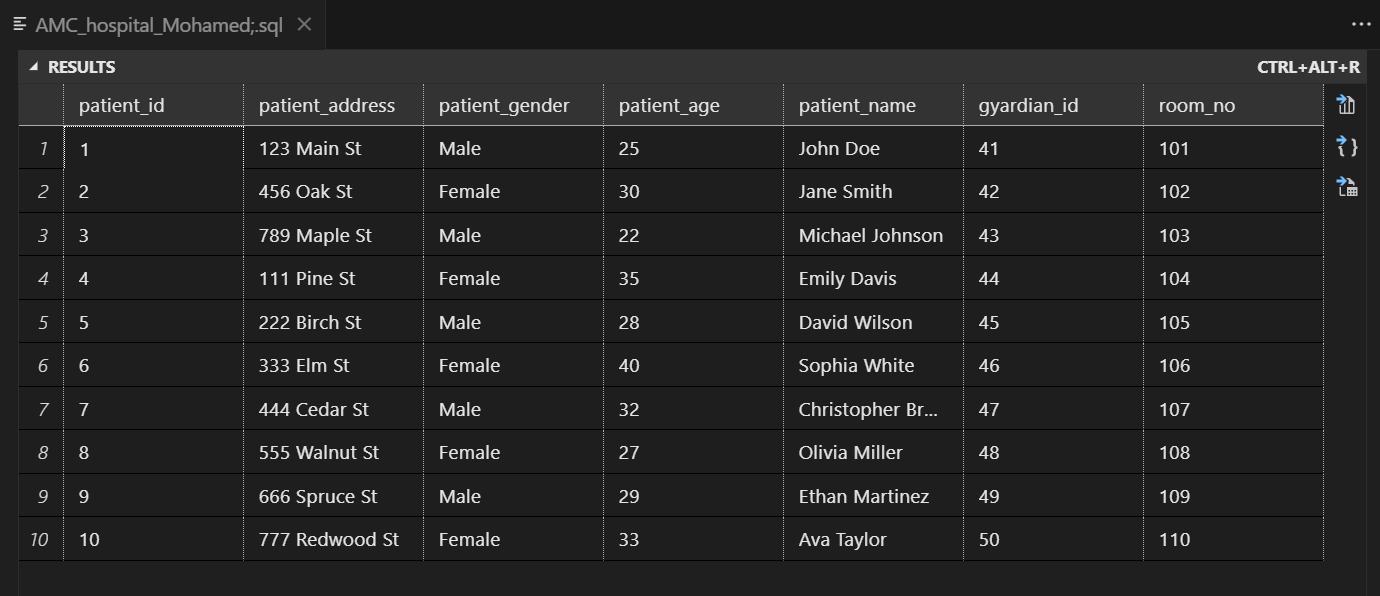
****

**INSERT INTO DOCTOR\_HAS\_PATIENT\_TABLE**

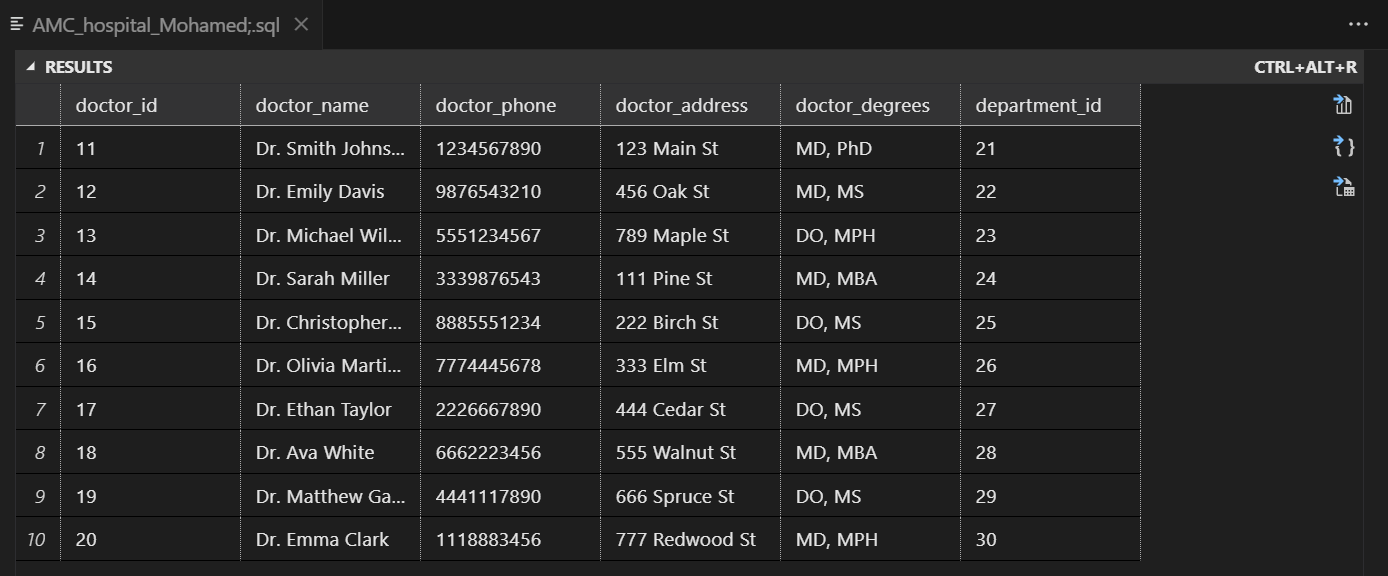
****

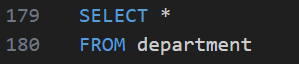
**QUERIES**

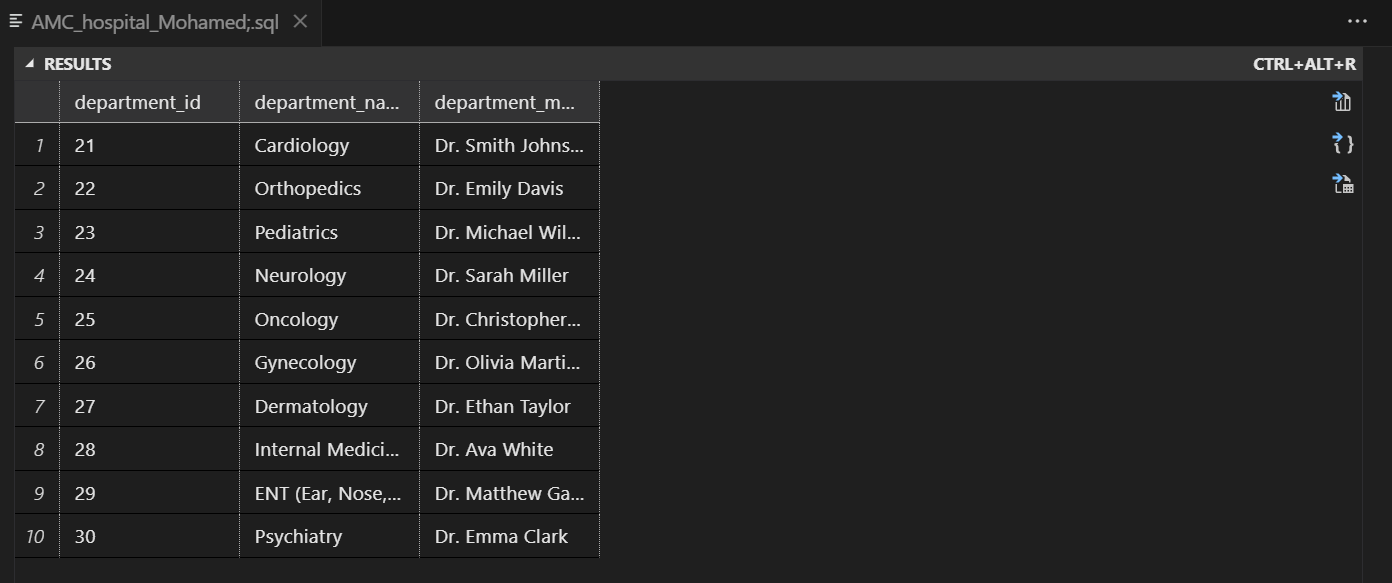
****

****

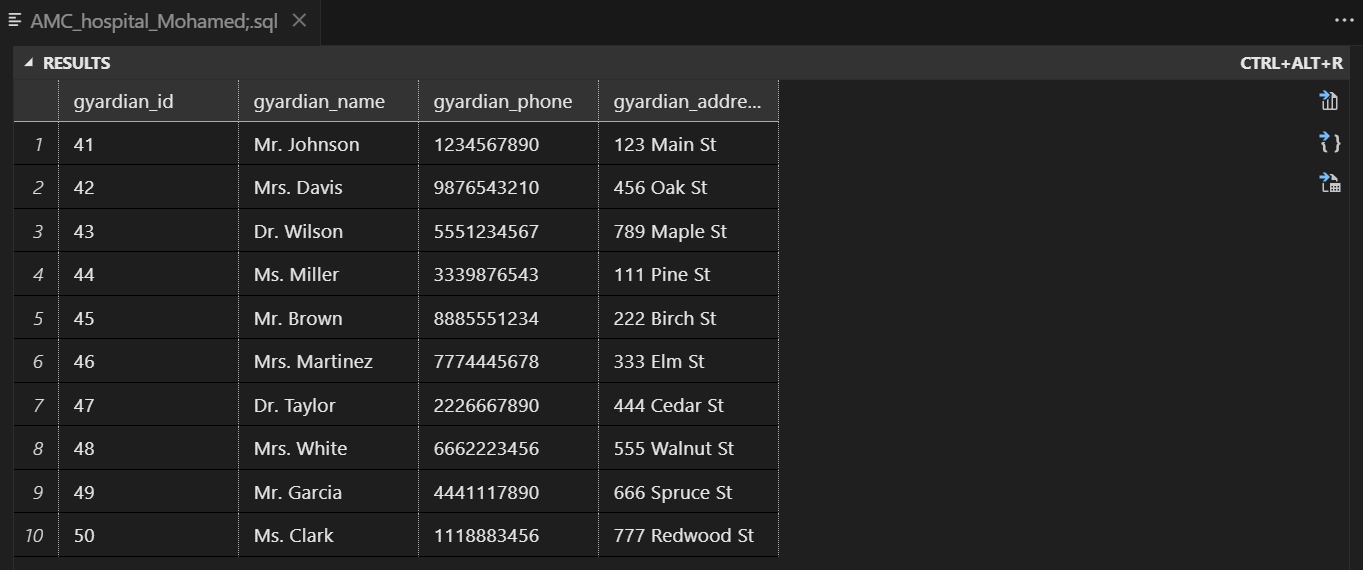
****

****

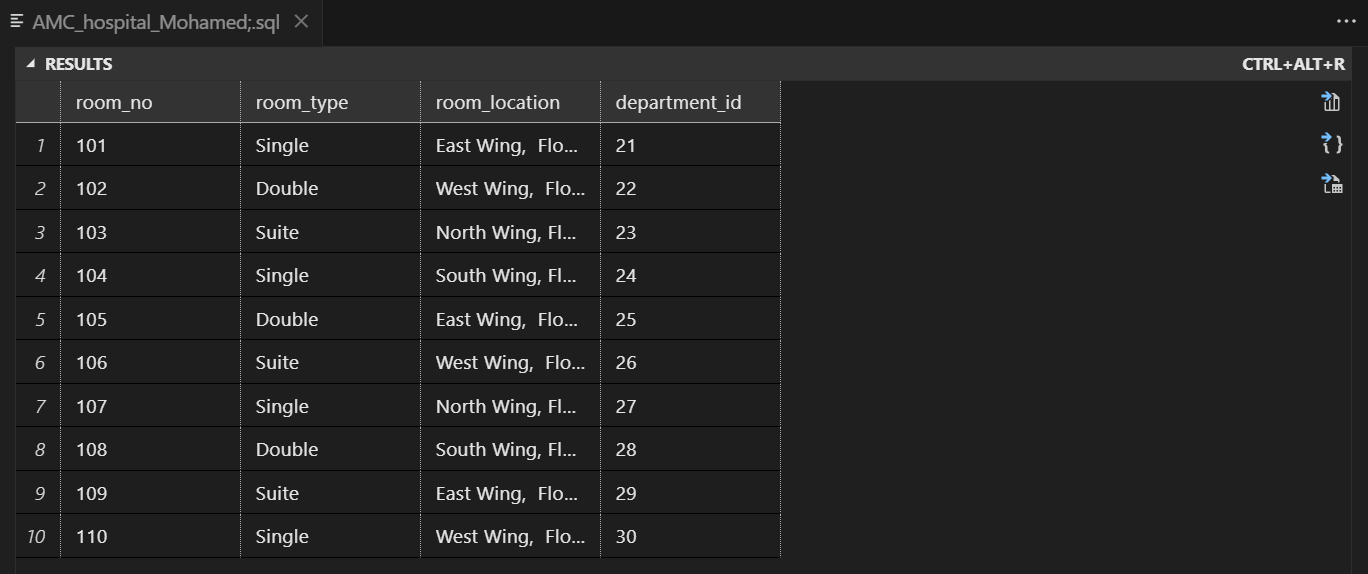
****

****

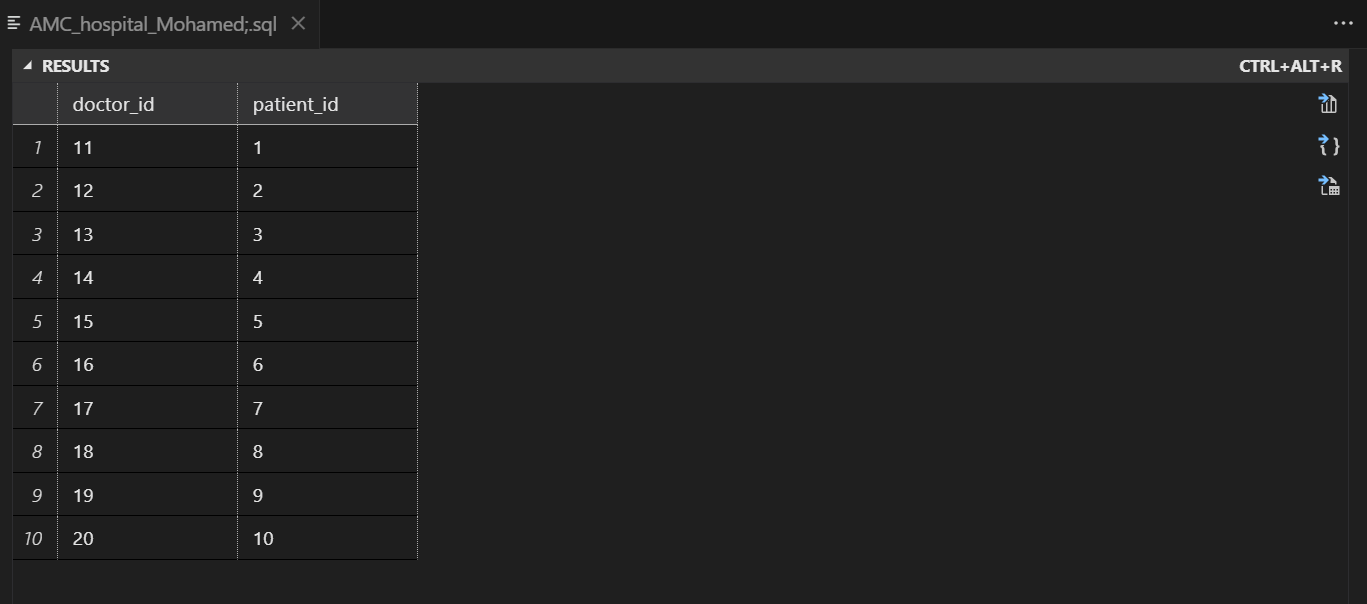
****

****

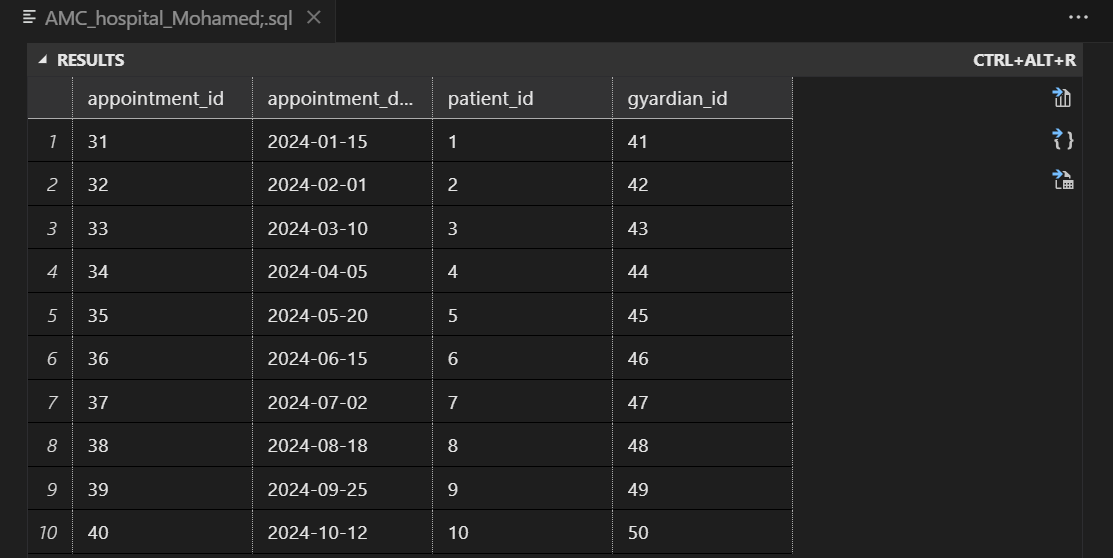
****

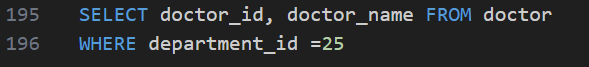
****

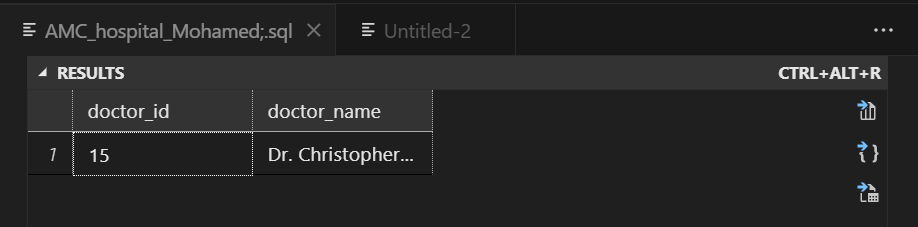
****

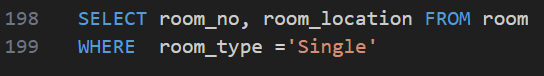
****

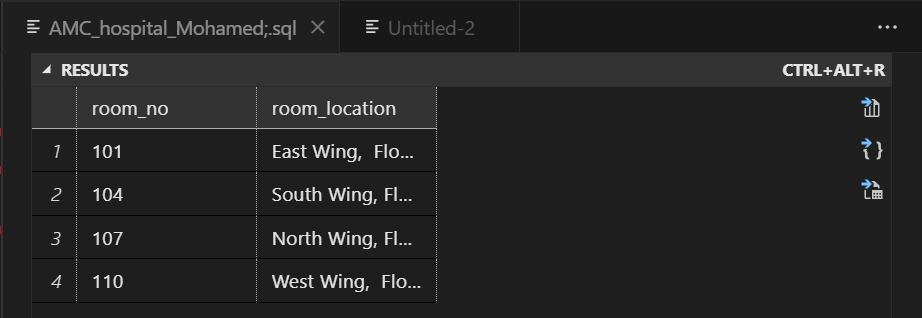
****

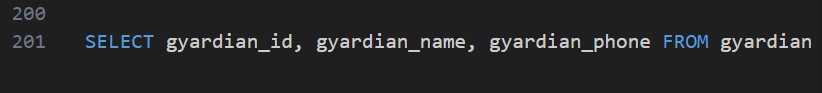
****

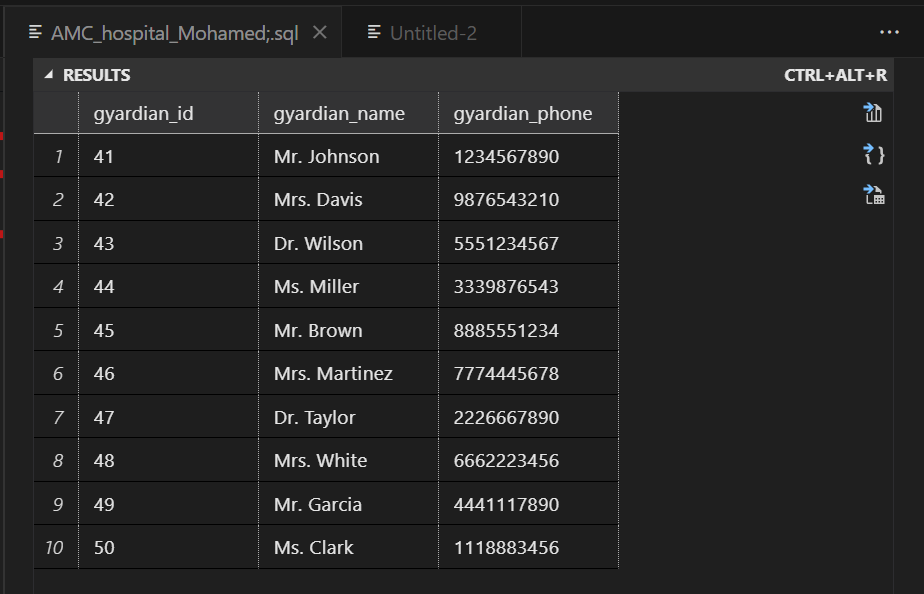
****

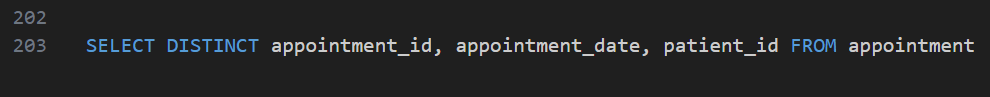
****

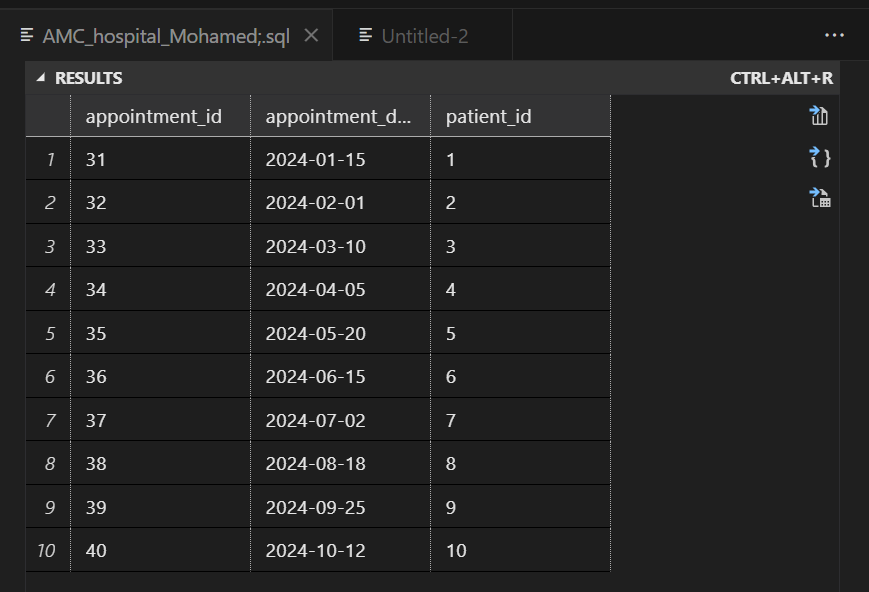
****

****

****

****

****

****

**Conclusion:**

SQL, with its DROP, ALTER, UPDATE, DELETE, and JOIN statements, provides a robust foundation for effective database design and development. Whether you're creating, modifying, or connecting data, a solid understanding of these SQL commands empowers you to build scalable and efficient database systems. As you navigate the intricate world of SQL, remember that each command serves a purpose in crafting data ecosystems that drive innovation and insight. Happy coding! [15]

[1] “Structured vs. Unstructured Data: A Complete Guide,” Talend - A Leader in Data Integration & Data Integrity. Accessed: Jan. 14, 2024. [Online]. Available: https://www.talend.com/resources/structured-vs-unstructured-data/

[2] “What is a database?” Accessed: Jan. 14, 2024. [Online]. Available: https://www.oracle.com/in/database/what-is-database/

[3] “Relational vs Non-Relational Databases.” Accessed: Jan. 14, 2024. [Online]. Available: https://www.pluralsight.com/blog/software-development/relational-vs-non-relational-databases

[4] N. Duffy, “Exploring User Requirements and Functional Requirements,” QRA. Accessed: Jan. 14, 2024. [Online]. Available: https://qracorp.com/exploring-user-requirements-and-functional-requirements/

[5] B. Johnston, “Unmasking SQL Server Dynamic Data Masking - Part 1 - Introduction,” Simple Talk. Accessed: Jan. 14, 2024. [Online]. Available: https://www.red-gate.com/simple-talk/blogs/unmasking-sql-server-dynamic-data-masking/

[6] “Why identification of stakeholders is important in project management?” Accessed: Jan. 14, 2024. [Online]. Available: https://www.linkedin.com/pulse/why-identification-stakeholders-important-project-mohammed-ajaz-pmp

[7] “Platform Requirements Gathering Techniques.” Accessed: Jan. 14, 2024. [Online]. Available: https://www.linkedin.com/pulse/platform-requirements-gathering-techniques-lalit-kale

[8] “What Is Requirement Analysis [Requirement Analysis Example/Sample Inside] | 2024,” Simplilearn.com. Accessed: Jan. 14, 2024. [Online]. Available: https://www.simplilearn.com/what-is-requirement-analysis-article

[9] “What Are System Requirements Specifications? | Inflectra.” Accessed: Jan. 14, 2024. [Online]. Available: https://www.inflectra.com/Ideas/Topic/Requirements-Definition.aspx

[10] “DBMS: Database Management Systems Explained,” BMC Blogs. Accessed: Jan. 14, 2024. [Online]. Available: https://www.bmc.com/blogs/dbms-database-management-systems/

[11] “What is Delta Live Tables?” Accessed: Jan. 14, 2024. [Online]. Available: https://docs.databricks.com

[12] “What is an Entity Relationship Diagram (ERD)?,” Lucidchart. Accessed: Jan. 14, 2024. [Online]. Available: https://www.lucidchart.com/pages/er-diagrams

[13] A. Watt, “Chapter 8 The Entity Relationship Data Model,” Oct. 2014, Accessed: Jan. 14, 2024. [Online]. Available: https://opentextbc.ca/dbdesign01/chapter/chapter-8-entity-relationship-model/

[14] “Introduction of Relational Algebra in DBMS,” GeeksforGeeks. Accessed: Jan. 14, 2024. [Online]. Available: https://www.geeksforgeeks.org/introduction-of-relational-algebra-in-dbms/

[15] “SQL ALTER TABLE - ADD, DROP, MODIFY,” GeeksforGeeks. Accessed: Jan. 14, 2024. [Online]. Available: https://www.geeksforgeeks.org/sql-alter-add-drop-modify/

­­­­­