**PRISMA ORM**

* An ORM (Object-Relational Mapping) is a technique or a tool that allows developers to interact with a database using an object-oriented approach, rather than writing SQL queries.
* It makes it easy for developers to reason about their database queries by providing a clean and type-safe API for submitting database queries which returns *plain old JavaScript objects*.

1. **How to install and set up Prisma ORM**

* This command ensures this ensures that it is installed during deployment.



* Setting up with our respective database of choice for this instance will be Postgres



* The following command does two important things:
* First it Creates a new directory called Prisma that contains the file schema. Prisma, which contains our database connection variables and models.
* Secondly it Creates a .env file at the root of the project which is used for defining the environment variables such as the connection string.



1. **Models**

Models represent the **entities** of the application domain Map to the **tables** (relational databases like PostgreSQL) or **collections** (MongoDB) in your database.

Models include:

1. Data Models: Define the tables and their columns (fields) in your database.
2. Field Types: Specify the data types for each field (e.g., String, Int, Boolean).
3. Relations: Describe how models relate to each other using relation fields.
4. Attributes: Add additional metadata to fields and models using attributes like @id, @default, @relation.

Attributes are stored in the schema file and includes:@id, @default,@unique,@updatedAt, @map,@relation when it has @@ it changes to model attributes.(it isn’t supported by MongoDB)

1. **Migrations**

migrations are a way to manage and apply changes to your database in a controlled and consistent way.

To run migrations this is the command we will use:



1. **Prisma client**

 Auto-generated and type-safe query builder for Node.js & TypeScript .In order to generate it we can run the command below and it generates the Prisma client within the node modules @prisma/client



1. **CRUD OPERATIONS(CREATE,READ,UPDATE,DELETE)**
2. **CREATE**

([create()](https://www.prisma.io/docs/orm/reference/prisma-client-reference#create)) creates a single user with two fields

([CreateMany()](https://www.prisma.io/docs/orm/reference/prisma-client-reference" \l "createmany)query creates multiple users and skips any duplicates

1. **READ**

([findUnique()](https://www.prisma.io/docs/orm/reference/prisma-client-reference" \l "findunique))by unique identifier or ID and returns a single record

([findMany()](https://www.prisma.io/docs/orm/reference/prisma-client-reference" \l "findmany)) query returns *all* User records

([findFirst()](https://www.prisma.io/docs/orm/reference/prisma-client-reference" \l "findfirst)) query returns the record that specifies a certain criterion that applies the *where* filter

1. **UPDATE**

 ([update()](https://www.prisma.io/docs/orm/reference/prisma-client-reference#update)) to find and update a single User record

 ([updateMany()](https://www.prisma.io/docs/orm/reference/prisma-client-reference" \l "updatemany)) to update all User records

1. **DELETE**

uses ([delete()](https://www.prisma.io/docs/orm/reference/prisma-client-reference#delete)) to delete a single User record

uses ([deleteMany()](https://www.prisma.io/docs/orm/reference/prisma-client-reference" \l "deletemany)) to delete all User records

1. **RELATIONSHIPS**

 relationship is a connection between two models in the Prisma schema

types include:

* [One-to-one](https://www.prisma.io/docs/orm/prisma-schema/data-model/relations/one-to-one-relations) (also called 1-1 relations)
* [One-to-many](https://www.prisma.io/docs/orm/prisma-schema/data-model/relations/one-to-many-relations) (also called 1-n relations)
* [Many-to-many](https://www.prisma.io/docs/orm/prisma-schema/data-model/relations/many-to-many-relations) (also called m-n relations)

**One-to-one relationship(1-1)**

A one-to-one relationship occurs when one record in a table is associated with exactly one record in another table. User ↔ Profile

Example: When a user can have more than 1 profile but each profile is linked to one user

[**One-to-many**](https://www.prisma.io/docs/orm/prisma-schema/data-model/relations/one-to-many-relations)**relationship(1-n )**

A one-to-many relationship exists when a single record in one table can be associated with multiple records in another table. User ↔ Post

Example: when a user can write many posts but each post belongs to a user and it must the @unique attribute.

[**Many-to-many**](https://www.prisma.io/docs/orm/prisma-schema/data-model/relations/one-to-many-relations)**relationship(m-n )**

Many-to-many relationships allow multiple records in one table to be associated with multiple records in another table.

Example: when an author can have multiple books and books can have various authors and therefore you want to store the role of each author with each book .The @@id attribute specifies a composite primary key, ensuring uniqueness for each combination chosen.