**PRISMA ORM SUMMARY NOTES.**

**What is Prisma?**

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.

**Prisma**simplifies **database**access by providing a type-safe and intuitive **ORM (Object-Relational Mapping)** layer.

**What are Benefits**

* Productivity: Reduces the amount of boilerplate code for database interactions.
* Portability: Makes it easier to switch databases without significant code changes.
* Security: Helps prevent SQL Injection by using parameterized queries.
* Maintenance: Simplifies the maintenance of database interactions and schema changes

**STEPS TO USE PRISMA.**

**First Step: installing Prisma.**

npm install prisma -D

**Second step: Set up Prisma.**

npx prisma init --datasource-provider DATABASE

**Third Step: Models**

model Product {

id Int @id @default(autoincrement())

First\_Name String @map("First Name")

Last\_Name String @map("Last\_Name")

Fee\_balance Float @map("Fee\_Balance")

@@map("STUDENTS")

}

**FOUTH STEP: Migrations**

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

npx prisma migrate dev --name MIGRATION-NAME

**FITH STEP : PRISMA CLIENT**

The Prisma Client is an auto-generated and type-safe database client that you use to interact with your database in a Node.js or TypeScript application.

npx prisma generate

OR

npm install @prisma/client

If you have not already installed.

**CRUD OPERATIONS.**

**Create.**

This operation is used to create records, you can create one record or Many Records.

Syntax for creating a single record

create({data:})

Syntax for Multiple records using **CreateMany()**

# **Read.**

Read operation is used to retrieve data from the database.

One can get all records using **findMany()**

You can find the first record record that matches a criteria using **findFirst()** and apply the filter using **Where**

**Update.**

Updating means modify existing records in a database.

## Update a single record using update()

## Update multiple records using updateMany()

**Delete.**

This operation removes records from databases.

## Delete a single record using delete()

## Delete multiple records using deleteMany()

**RELATIONSHIPS.**

Relationships define how different models (tables) in a database are connected to each other.

The types of relationships include:

* one-to-one relationship (1-1): means that one record in a table is associated to only one record in another table.

Example: Every Kenyan has one national id number and An Id number has only one person.

* one-to-many relationship (1-n): means that one record in a table can be associated with multiple other records in another table.

Example: One student has many units during his university education.

* many-to-many relationship (m-n): means that multiple records in one table can be associated to multiple records in another table.

Example: Many lectures teach many units.