Prisma Commands and Their Uses

A Practical Guide for Efficient Database Management

jsupskills.dev

What is Prisma?

- A modern ORM for Node.js & TypeScript
- Simplifies database access
- Works with PostgreSQL, MySQL, SQLite, MongoDB, etc.

jsupskills.dev

Prisma CLI Overview

- **npx prisma** → Run Prisma CLI commands
- **prisma --help** → Lists available commands

jsupskills.dev

Project Initialization

- **Command:** npx prisma init
- **Use:** Initializes a Prisma project by creating a prisma directory, schema.prisma, and .env file.

Generating Prisma Client

- **Command:** npx prisma generate
- **Use:** Generates Prisma Client, which is needed to interact with the database in your application.

jsupskills.dev

Formatting Schema

- **Command:** npx prisma format
- **Use:** Formats the schema.prisma file for better readability.

Database Migrations (Development)

- **Command:** `npx prisma migrate dev --name <migration_name>`
- **Use:** Creates and applies a migration in a development environment.

Resetting the Database

- **Command:** npx prisma migrate reset
- **Use:** Resets the database by dropping all tables and reapplying migrations. Useful for development.

Deploying Migrations (Production)

- **Command:** npx prisma migrate deploy
- **Use:** Applies all pending migrations in a production database.

Checking Migration Status

- **Command:** npx prisma migrate status
- **Use:** Checks the current state of database migrations.

Pushing Schema Changes (No Migrations)

- **Command:** npx prisma db push
- **Use:** Updates the database schema without creating a migration. Ideal for prototyping.

Seeding the Database

- **Command:** npx prisma db seed
- **Use:** Runs a seed script to populate the database with initial data.

Pulling Database Schema

- **Command:** **npx prisma db pull**
- **Use:** Introspects the database and updates schema.prisma accordingly.

Executing Raw SQL

- **Command:** `npx prisma db execute --file <file.sql>`
- **Use:** Runs raw SQL queries from a specified file.

Querying the Database

- **prisma.<model>.findMany()** ➜ Fetch multiple records.
- **prisma.<model>.findUnique({ where: { id: 1 } })** ➜ Fetch a single record.
- **prisma.<model>.create({ data: {...} })** ➜ Insert a new record.
- **prisma.<model>.update({ where: {...}, data: {...} })** ➜ Update a record.
- **prisma.<model>.delete({ where: {...} })** ➜ Delete a record.

Running Raw SQL Queries

- **Command:** prisma.$queryRaw
- **Use:** Runs raw SQL queries and retrieves results.

Executing Raw SQL Commands

- **Command:** prisma.$executeRaw
- **Use:** Executes raw SQL commands without returning results.

jsupskills.dev

Using Prisma Studio (GUI Tool)

- **Command:** npx prisma studio
- **Use:** Opens Prisma Studio, a web-based GUI to visualize and manage database records.

Debugging & Logs

- **export DEBUG="prisma:*"**

  (Linux/macOS)
- **set DEBUG=prisma:*** (Windows)
- **Use:** Enables debugging logs for

  Prisma CLI.

Best Practices

- Always use migrations for schema changes.
- Validate queries before execution.
- Optimize queries for performance.
- Keep Prisma Client instance global.

Conclusion & Next Steps

- Recap key Prisma commands.
- Explore advanced topics like middleware, transactions, and caching.
- Visit prisma.io for documentation.
- Learn more at jsupskills.dev for advanced JavaScript resources.

jsupskills.dev