■ Prisma ORM Database Guide

1. Database Migration

Database migration in Prisma allows you to sync your Prisma schema with the actual database. Migrations create or update database tables, columns, and relations.

```
// Step 1: Define your schema in prisma/schema.prisma
model User {
  id    Int     @id @default(autoincrement())
  name  String
  email String  @unique
}

// Step 2: Run migration
npx prisma migrate dev --name init

// Step 3: Generated migration file will be in prisma/migrations/
```

2. Database Seeder

A seeder is used to populate your database with initial or test data. You can use Prisma Client inside a script for seeding.

```
// prisma/seed.js
const { PrismaClient } = require('@prisma/client');
const prisma = new PrismaClient();

async function main() {
  await prisma.user.createMany({
    data: [
      { name: "Ashish", email: "ashish@example.com" },
      { name: "Kumar", email: "kumar@example.com" }
    ]
  });
}

main().finally(() => prisma.$disconnect());
```

3. Database Factory / Data Generator

Factories help generate fake or random data for testing. You can use libraries like Faker.js with Prisma.

```
// prisma/factory.js
const { PrismaClient } = require('@prisma/client');
const { faker } = require('@faker-js/faker');
const prisma = new PrismaClient();

async function main() {
  for (let i = 0; i < 10; i++) {
    await prisma.user.create({
      data: {
        name: faker.person.fullName(),
        email: faker.internet.email()
      }
    });
  }
}

main().finally(() => prisma.$disconnect());
```

4. Database Relationships

Prisma supports different relationships between models (tables). Relations are defined in the Prisma schema and reflected in queries.

■ One-to-Many (User → Post)

```
model User {
  id    Int     @id @default(autoincrement())
  name  String
  posts Post[]
}

model Post {
  id       Int    @id @default(autoincrement())
  title    String
  author   User   @relation(fields: [authorId], references: [id])
  authorId Int
}
```

■ Many-to-Many (User ↔ Group)

```
model User {
  id     Int     @id @default(autoincrement())
  name   String
  groups Group[]
}

model Group {
  id     Int     @id @default(autoincrement())
  name   String
  users  User[]
}
```

■ One-to-One (User → Profile)

```
model User {
  id      Int     @id @default(autoincrement())
  name    String
  profile Profile?
}

model Profile {
  id     Int   @id @default(autoincrement())
  bio    String
  user   User  @relation(fields: [userId], references: [id])
  userId Int   @unique
}
```

■ Example Queries

```
// Fetch all users with posts
const users = await prisma.user.findMany({
  include: { posts: true }
});

// Create post with user relation
await prisma.post.create({
  data: {
    title: "Hello World",
    author: { connect: { id: 1 } }
  }
});
```