PostgreSQL and TypeORM

Course Code: CSC 4182 Course Title: Advanced Programming In Web Technologies

Dept. of Computer Science
Faculty of Science and Technology

Lecture No:	**1**	**Week No:**	**04**	**Semester:**	
Lecturer:	*Sazzad Hossain; sazzad@aiub.edu*				

Lecture Outline

- ✓ Object-Relational Mapping (ORM)
- ✓ TypeORM
- ✓ PostgreSQL
- ✓ TypeORM and PostgreSQL Setup

Object-Relational Mapping (ORM)

ORM is a technique **allows** developers to interact with relational **databases** using **object-oriented paradigms**. ORM frameworks **provide** a way to **map** database **tables to classes and objects** to perform **database operations using familiar object-oriented syntax.**

Some characteristics of ORM:

- Abstraction of Database Complexity
- Object-Relational Mapping
- Database Agnostic
- Data Validation and Type Safety
- Query Building and Optimization
- Relationships and Associations

Object-Relational Mapping (ORM)

Following is an example of SQL query without using ORM

```
var sql = "SELECT id, first_name, last_name, phone, birth_date, sex, age FROM persons WHERE id = 10";
var result = context.Persons.FromSqlRaw(sql).ToList();
var name = result[0]["first_name"];
```

Example of SQL query using ORM

```
var person = repository.GetPerson(10);
var firstName = person.GetFirstName();
```

Object-Relational Mapping (ORM)

Popular ORM libraries;

JavaScript/TypeScript:

- Sequelize (JavaScript/TypeScript)
- **TypeORM (TypeScript)**
- Prisma (TypeScript)
- MikroORM (TypeScript)

Python:

- Django ORM (Python)

PHP

- Laravel ORM Eloquent(PHP)

Ruby:

- ActiveRecord (Ruby)

Java:

- Hibernate (Java)
- EclipseLink (Java)

C#:

- Entity Framework (C#)

TypeORM

TypeORM is an ORM used with TypeScript and JavaScript (ES5, ES6, ES7, ES8). It is designed to simplify database access and management.

TypeORM supports multiple database systems, including MySQL, **PostgreSQL**, SQLite, Microsoft SQL Server, Oracle, and MongoDB.

Some Features of ORM:

- Supports both DataMapper and ActiveRecord.
- Entities and columns.
- Database-specific column types.
- Entity manager.
- Repositories and custom repositories.
- Clean object relational model.
- Associations (relations).

PostgreSQL

PostgreSQL/Postgres, is a powerful open-source relational database management system (RDBMS).

Some features of PostgreSQL

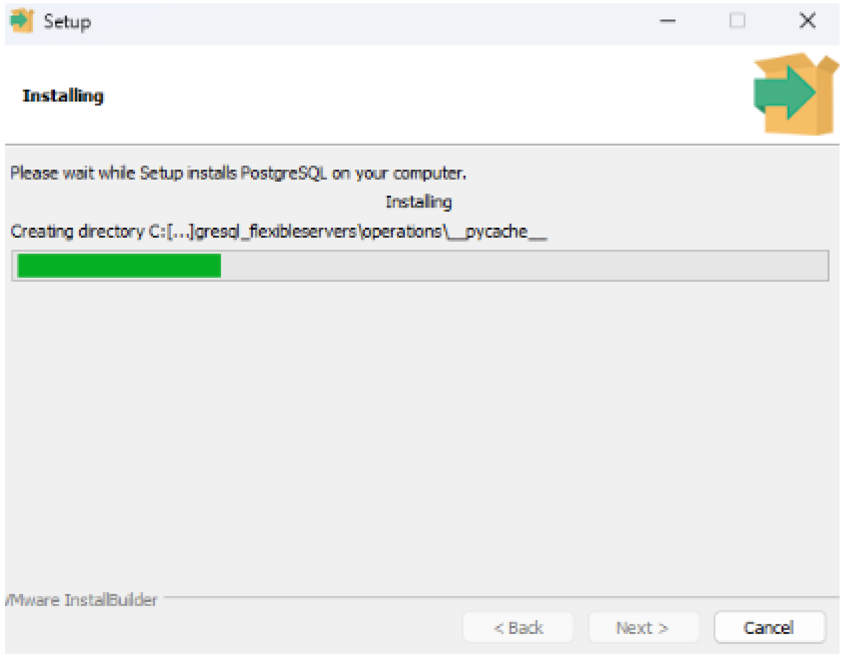
- Relational Database
- ACID Compliance
- Replication and High Availability
- Security
- Cross-Platform Compatibility
- Active Community and Ecosystem

PostgreSQL

Organizations uses PostgreSQL:
Apple: Apple uses PostgreSQL extensively in various applications and services, including iCloud and iTunes.
Cisco: Cisco, a multinational technology conglomerate, utilizes PostgreSQL for data management and analytics purposes.
Fujitsu: Fujitsu, a global IT services and solutions provider, has adopted PostgreSQL for its enterprise systems and solutions.
Red Hat: Red Hat, a leading provider of open-source solutions, includes PostgreSQL in its product portfolio and uses it for internal applications.
Uber: Uber, the ride-hailing and food delivery giant, relies on PostgreSQL for storing and managing vast amounts of data related to their operations.

TypeORM and PostgreSQL Setup

Download PostgreSQL from below link
https://www.enterprisedb.com/downloads/postgres-postgresql-downloads
then install it



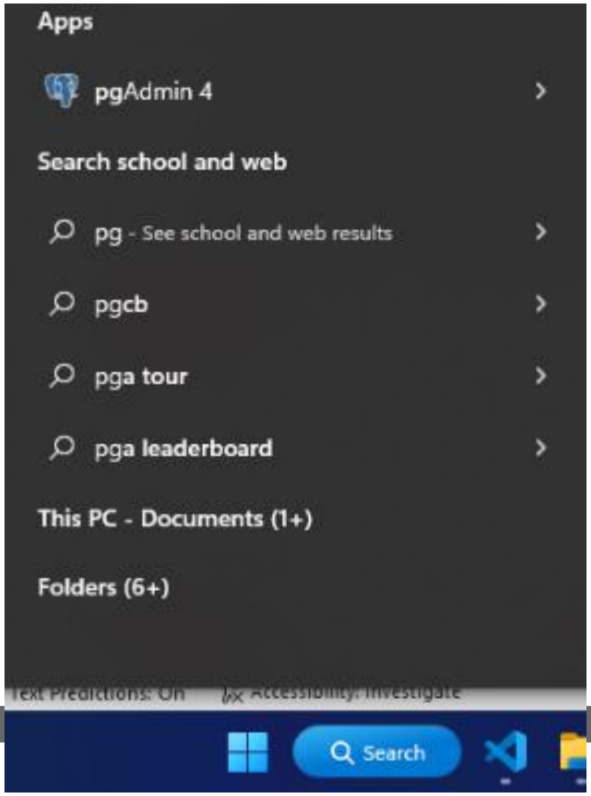
Please wait while Setup installs PostgreSQL on your computer.

Installing

Creating directory C:[...]gresql_flexibleservers\operations\__pycache__

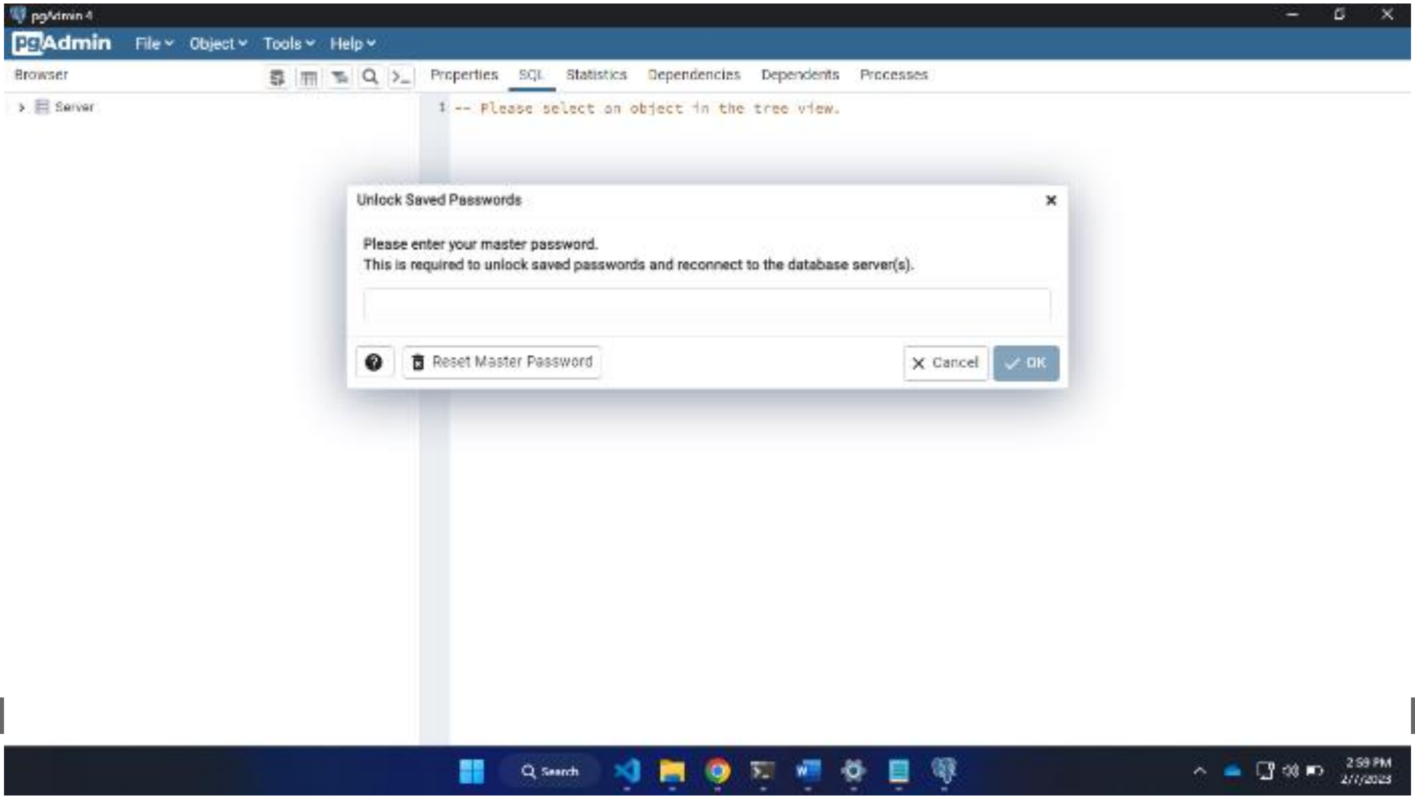
TypeORM and PostgreSQL Setup

After installation is completed the go to windows search bar and type **pgadmin** and select the app to open



TypeORM and PostgreSQL Setup

It will ask you to set or reset password. Insert the password what you have given during PostgreSQL installation.



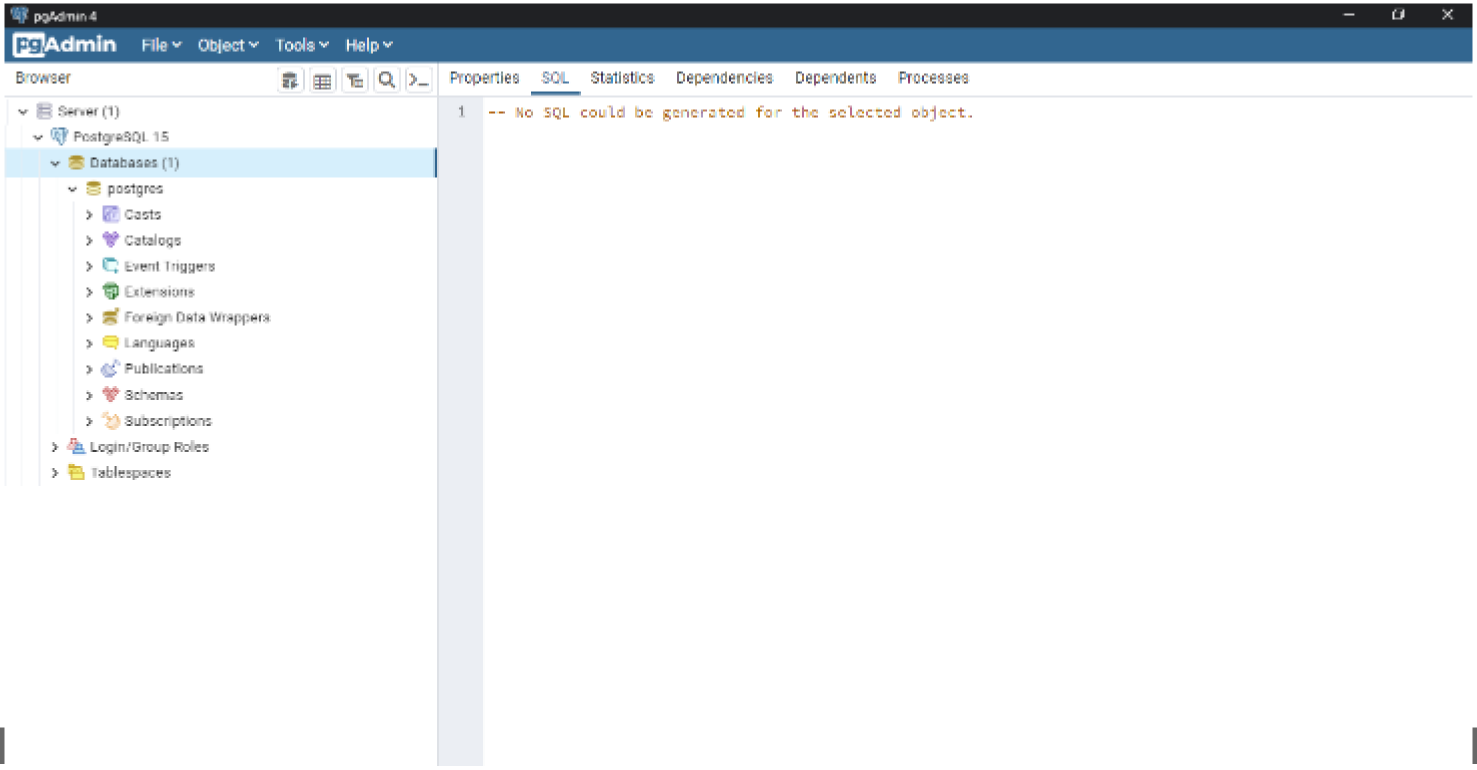
```
1 -- Please select an object in the tree view.
```

Unlock Saved Passwords

Please enter your master password.
This is required to unlock saved passwords and reconnect to the database server(s).

TypeORM and PostgreSQL Setup

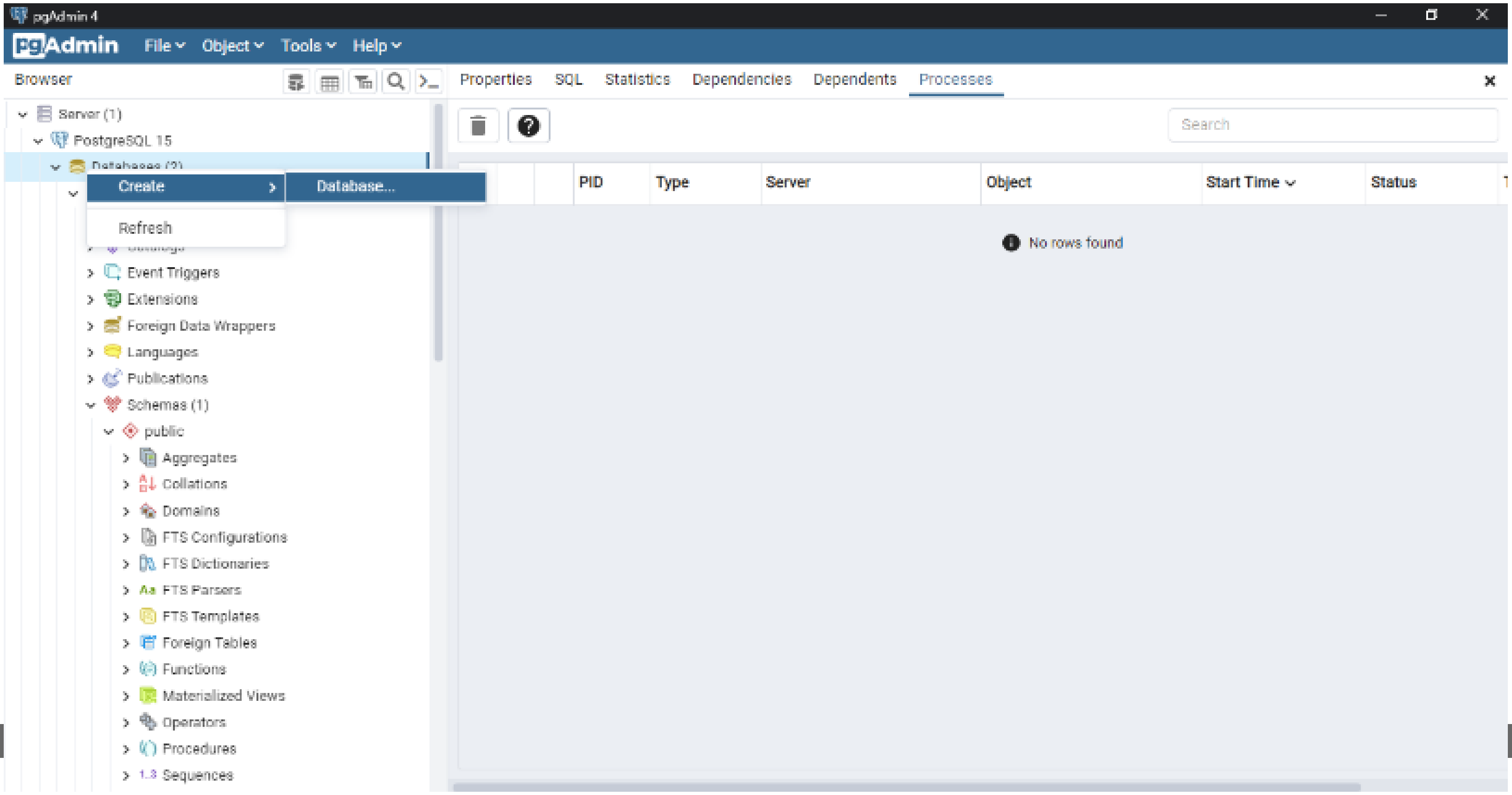
After that you will find below dash board



```
-- No SQL could be generated for the selected object.
```

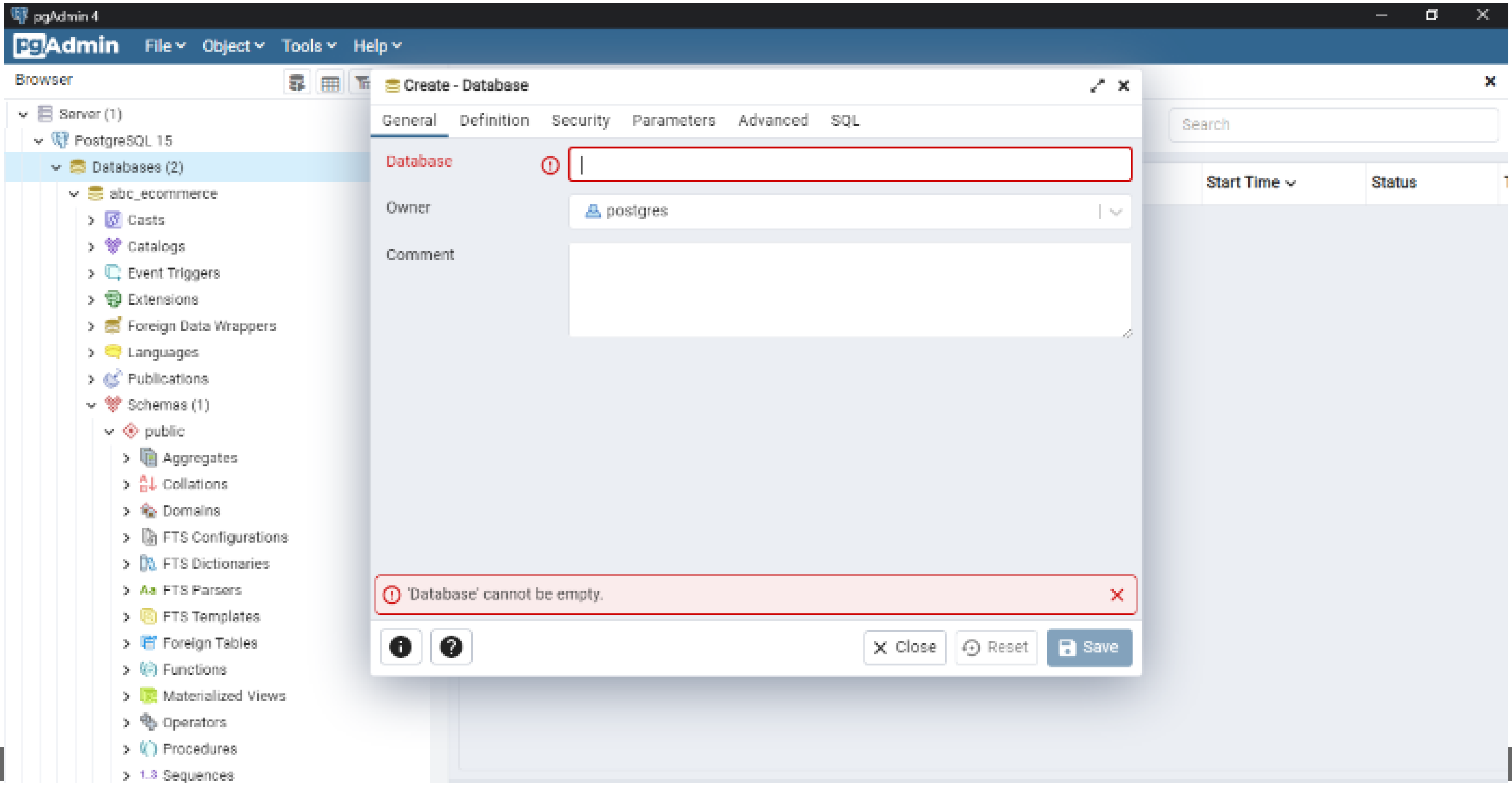
TypeORM and PostgreSQL Setup

Right click on database and create database



TypeORM and PostgreSQL Setup

Name the database based on your project.



Database:

Owner: postgres

Comment:

'Database' cannot be empty.

TypeORM and PostgreSQL Setup

Go to your nestjs **project folder** and write below command to install PostgreSQL driver for nestjs

```
npm i @nestjs/config @nestjs/typeorm typeorm pg
```

TypeORM and PostgreSQL Setup

then go to your project’s **app.module.ts** file and add database connection.

```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { AdminModule } from './admin/adminmodule.module';
import { ManagerModule } from './manager/manager.module';
@Module({
  imports: [AdminModule, ManagerModule, TypeOrmModule.forRoot(
   { type: 'postgres',
    host: 'localhost',
    port: 5432,
    username: 'postgres',
    password: 'root',
    database: 'abc_ecommerce',//Change to your database name
    autoLoadEntities: true,
    synchronize: true,
  }  ),
],
  controllers: [],
  providers: [],
})
export class AppModule {}
```

TypeORM and PostgreSQL Setup

Then, go to your **user-specific** folder, for example Admin and create an entity file for your user , for example **admin.entity.ts** and write down the below code;

```
import { Entity, Column, PrimaryGeneratedColumn} from 'typeorm’;

@Entity("admin")
export class AdminEntity{
  @PrimaryGeneratedColumn()
  id: number;
  @Column()
  name: string;
  @Column()
  email: string;
  @Column()
  password: string;
}
```

TypeORM and PostgreSQL Setup

Then, go to your **user-specific** folder, for example Admin and create an entity file for your user , for example **admin.entity.ts** and write down the below code;

```
import { Entity, Column, PrimaryGeneratedColumn} from 'typeorm’;

@Entity("admin")
export class AdminEntity{
  @PrimaryGeneratedColumn()
  id: number;
  @Column()
  name: string;
  @Column()
  email: string;
  @Column()
  password: string;
}
```

TypeORM and PostgreSQL Setup

Then update the admin.module.ts file to add the Entity class.

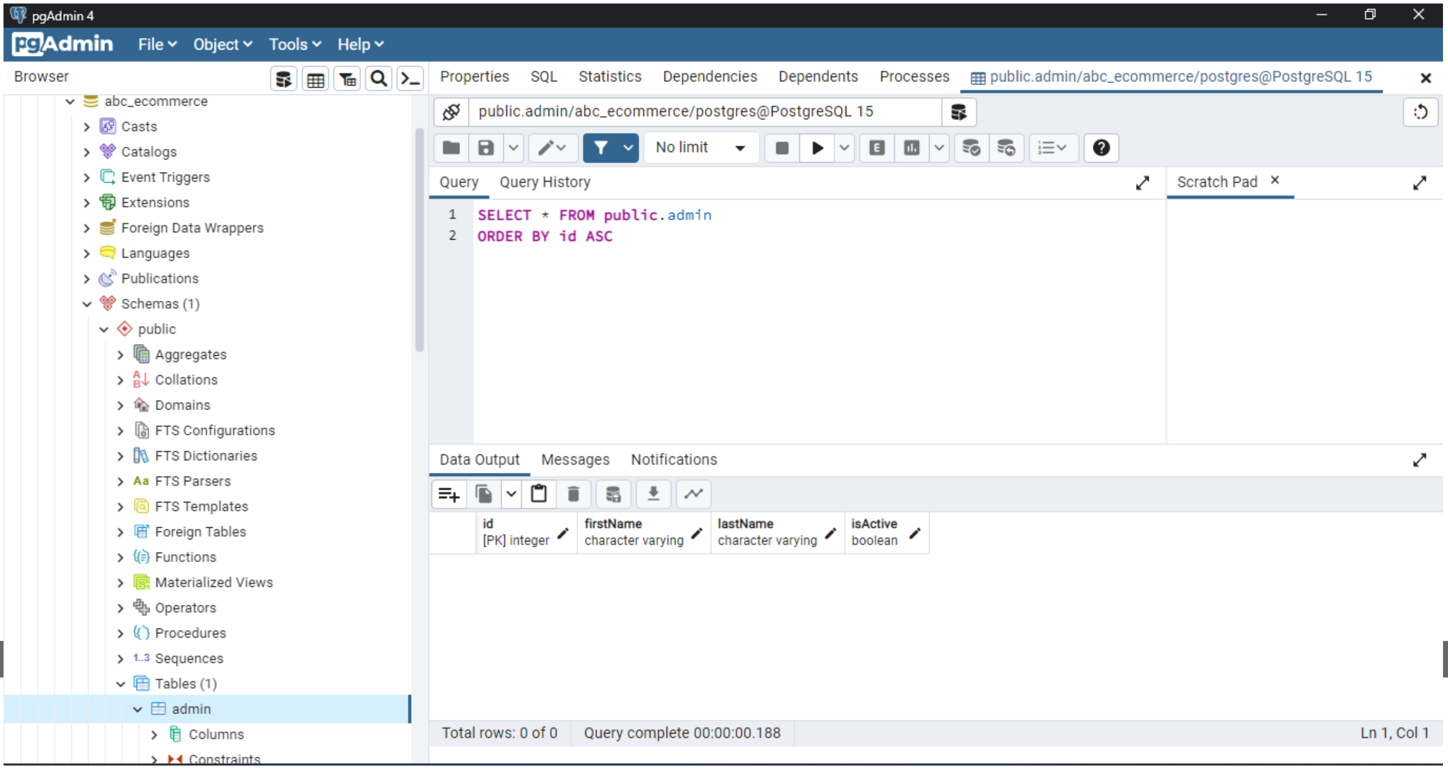
```
import { Module } from "@nestjs/common";
import { AdminController } from "./admin.controller";
import { AdminService } from "./admin.service";
import { AdminEntity } from "./admin.entity";
import { TypeOrmModule } from "@nestjs/typeorm";

@Module({
    imports: [ TypeOrmModule.forFeature([AdminEntity]),],
    controllers: [AdminController],
    providers: [AdminService],
  })
  export class AdminModule {}
```

TypeORM and PostgreSQL Setup

Now run your project

```
npm run start:dev
```

You suppose to see your table name admin is created.



public.admin/abc_ecommerce/postgres@PostgreSQL 15

```
SELECT * FROM public.admin
ORDER BY id ASC
```

id [PK] integer	firstName character varying	lastName character varying	isActive boolean

Total rows: 0 of 0   Query complete 00:00:00.188   Ln 1, Col 1

References

1. W3Schools Online Web Tutorials, URL: http://www.w3schools.com
2. Node.js, URL: https://nodejs.org/en/
3. Next.js, URL: https://nextjs.org/
4. TypeScript URL: https://www.typescriptlang.org/
5. MDN Web Docs URL: https://developer.mozilla.org/
6. TypeORM URL: https://typeorm.io/

Thank You!