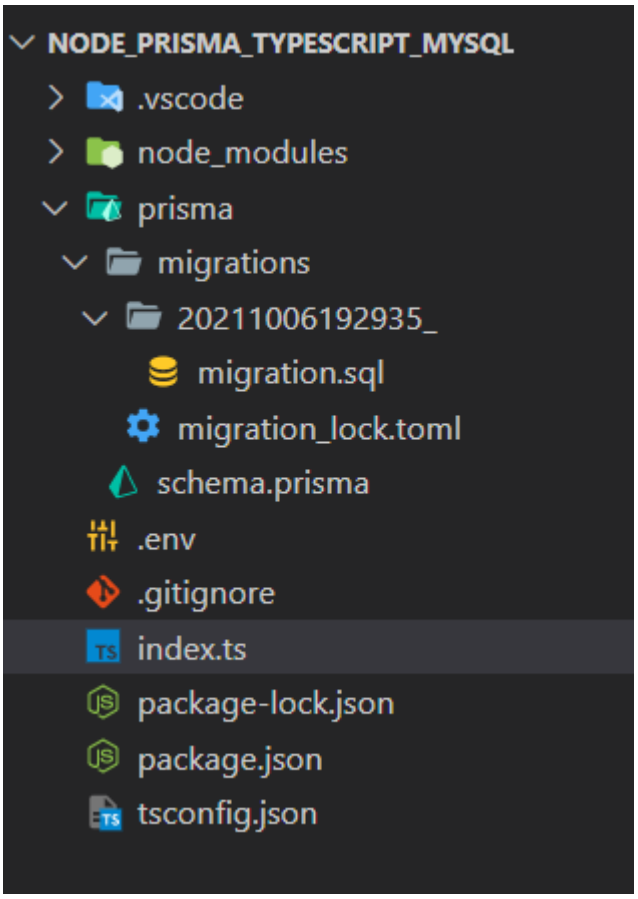
Practical-9

Aim: Use any one ORM from Sequelize ORM, TypeORM and Prisma ORM and implement following task.

1. Create Connection
2. Create Database
3. Create Table
4. Insert Record
5. Display Record
6. Association (any two)-> only for those who use Sequelize ORM

File- Structure: -



Code:-

1. index.ts

```
import express,{Request,Response} from 'express';
const { exec } = require('child_process');
require('dotenv').config();
import {PrismaClient} from '@prisma/client';
import execa from 'execa';

const PORT=process.env.PORT||3000
const DB=process.env.DB
const app=express()
app.use(express.json())

const prisma=new PrismaClient();

app.get("/db-create",async (req:Request,res:Response)=>{
  try {
         prisma.$connect()
        console.log("Database Connected Successfully!");
        res.send("<h1>Database Connected and Created Successfully</h1>")

    } catch (error) {
        console.log(error);
        res.send("<h1>Database Not Connected Successfully</h1>")
    }

})

app.get("/db-create-table",async (req:Request,res:Response)=>{
    try {

       const tb= await execa.command(`prisma migrate dev --preview-feature`)
        console.log("Table and Migration Folder Created");
        res.send(tb);


    } catch (error) {
        console.log(error);
        res.send("<h1>Database Not Connected Successfully</h1>")
    }

})

//Crud using Prisma
//Insert data
app.get("/db-insert",async (req:Request,res:Response)=>{
```

```
    const
data=[{fname:"Jahanvi",lname:"soni",email:"jahanvi212@mail.com",phoneNo:"2345678
"}]
    try {
        const user=await prisma.tblRegistration.createMany({
            data:data,
        });

        res.json(user);

    } catch (error) {
        console.log(error);
        res.send("Error Occured")

    }

});

//Display data
app.get("/db-display",async (req:Request,res:Response)=>{



    try {
        const students= await prisma.tblRegistration.findMany();
        res.json(students);

    } catch (error) {
        console.log(error);
        res.send("Error Occured")

    }

});

//Update data use put
app.get("/db-update",async (req:Request,res:Response)=>{



    try {
        const updatedUser=await prisma.tblRegistration.update({
            where:{
                id:4
            },
            data:{
                fname:"Amar"
            }
        });
```

```
        res.json(updatedUser)

    } catch (error) {
        console.log(error);
        res.send("Error Occured")

    }
});

// Delete data use delete

app.get("/db-delete/:id",async (req:Request,res:Response)=>{

    const id=req.params.id

    try {
        const deletedUser= await prisma.tblRegistration.delete({
            where:{
                id:Number(id),
            },
        });

        res.json(deletedUser)

    } catch (error) {
        console.log(error);
        res.send("Error Occured")

    }

});

app.listen(PORT,()=>{
   console.log(`Server is running on Port ${PORT}`);
})
```

2. schema.prisma

```
datasource db {
  provider = "mysql"
  url      = env("DATABASE_URL")
}

generator client {
  provider = "prisma-client-js"
}
```

```
model tblRegistration{
  id Int @default(autoincrement()) @id
  fname String @db.VarChar(50)
  lname String @db.VarChar(50)
  email String @unique
  phoneNo String @db.VarChar(10)

}
```

3. .env

DATABASE_URL="mysql://root:@localhost:3306/dbUniversity?schema=public"

DB=dbUniversity

PORT=5000

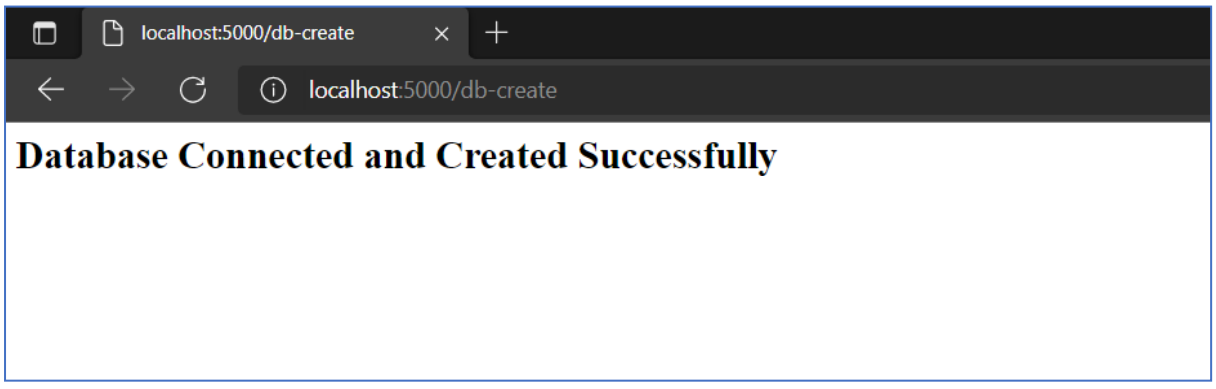
Output:-

1. Starting commands:

```
npm install prisma typescript ts-node @types/node --save-dev

npm install express @types/express nodemon execa

npx prisma

npx prisma init
```

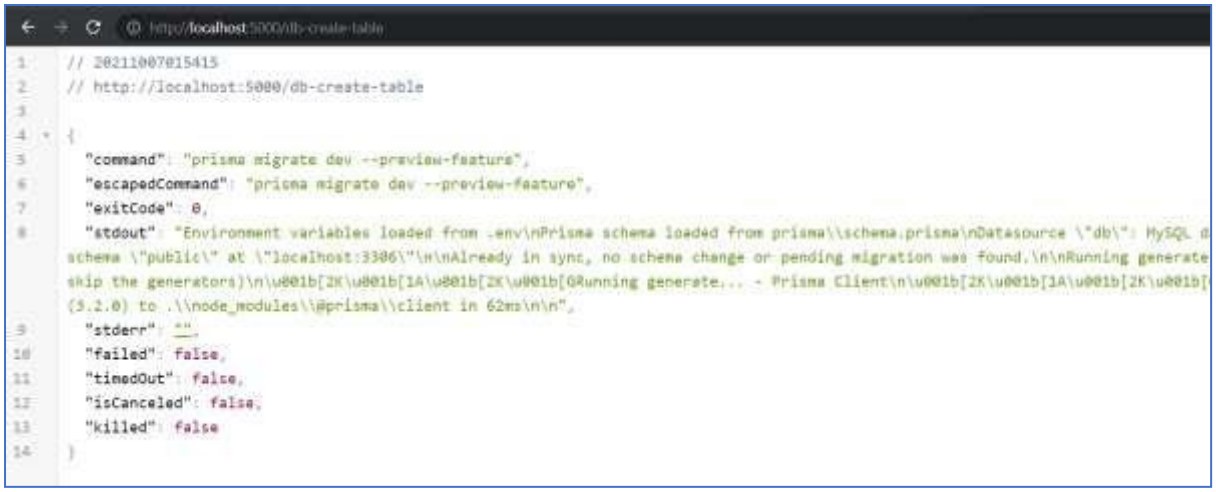
```
> nodemon index.ts

[nodemon] 2.0.12
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: ts,json
[nodemon] starting `ts-node index.ts`
Server is running on Port 5000
```

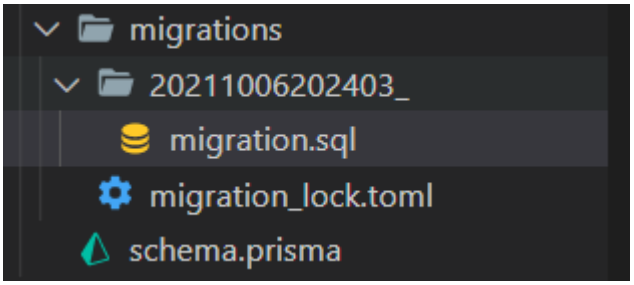
2. localhost:5000/db-create:




```
Server is running on Port 5000
Database Connected Successfully!
```

3. localhost:5000/db-create-table:





4. Database: