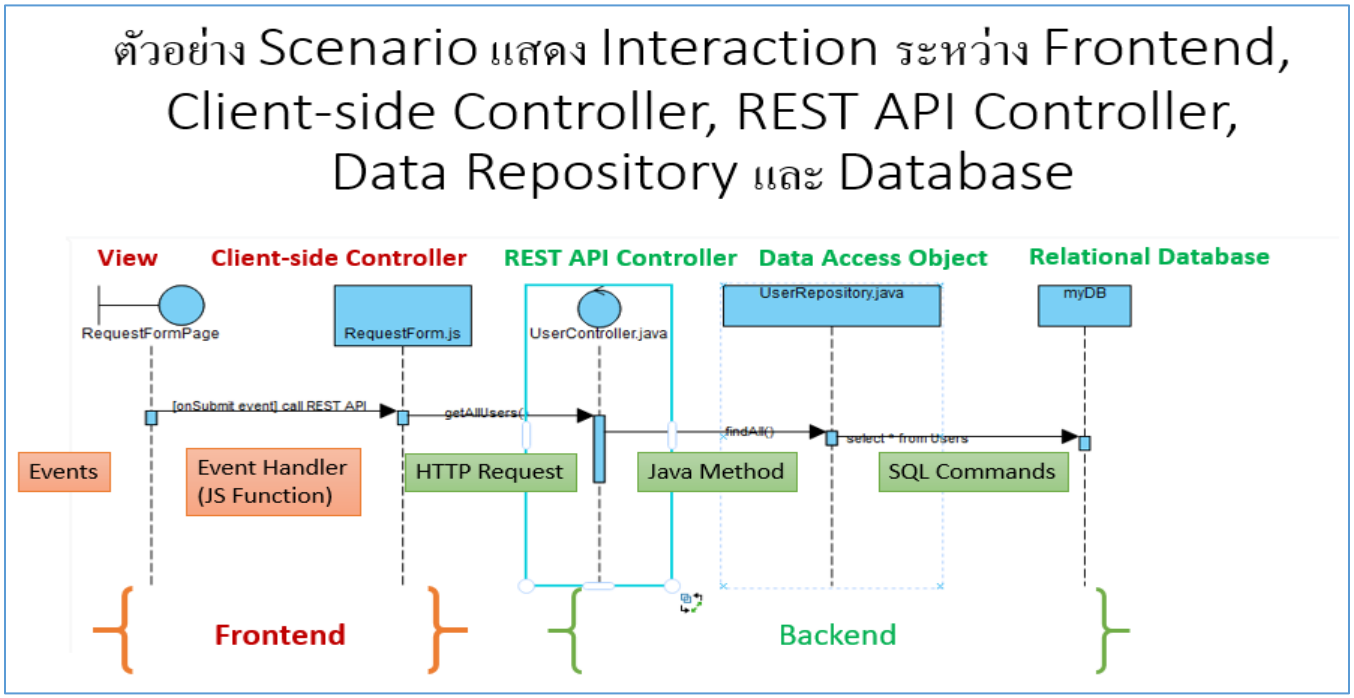
Workshop การเขียนโค้ดเพื่อสร้าง REST API ให้บริการประเภท CRUD จัดการข้อมูล โดยเชื่อมต่อกับฐานข้อมูลประเภท Relational Database

อ้างอิงโค้ด

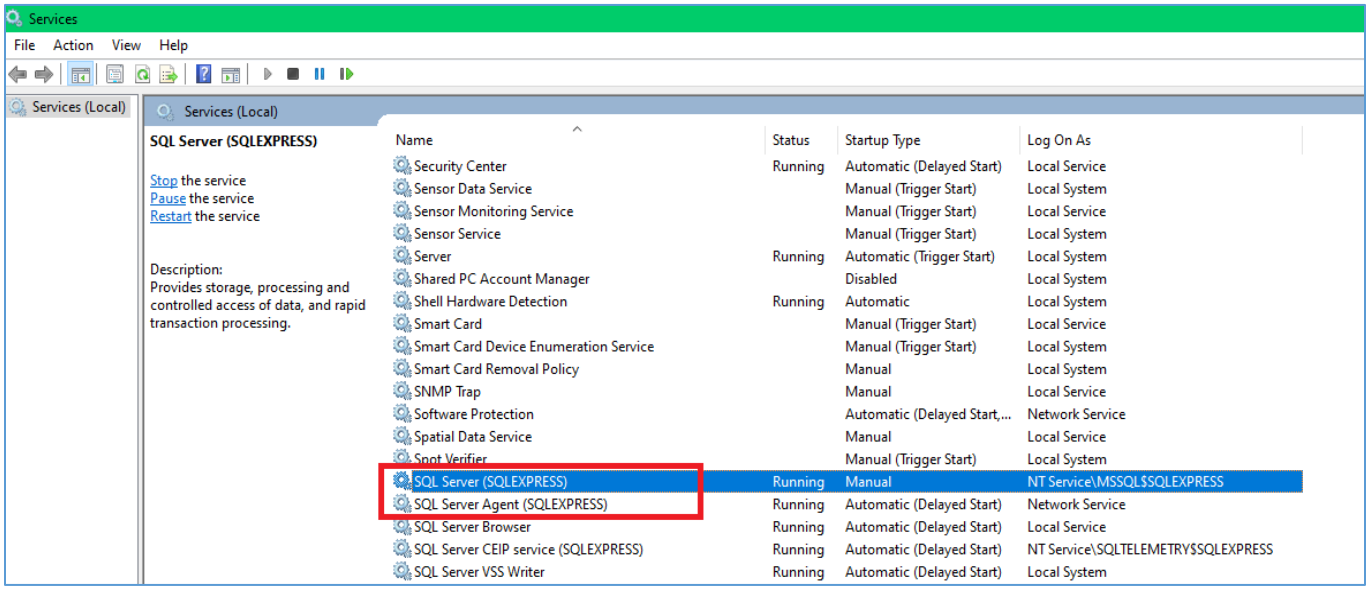
D:\cs264_rangsit\REST_Controller_All_HTTP_Methods_Repo_SQL\demo\src\main\java\com\example\demo



รูปที่ 1 แผนภาพ sequence แสดงลำดับการทำงานร่วมกันระหว่าง components เพื่อเปิดบริการ API จัดการข้อมูล User ในฐานข้อมูล

จากรูปที่ *1* แผนภาพ sequence แสดงลำดับการทำงานร่วมกันระหว่าง components เพื่อเปิดบริการ API จัดการข้อมูล User ในฐานข้อมูล ให้นักศึกษาพัฒนา components จาก ระดับ database ขึ้นมา โดยจะทำการสร้าง Table ตารางสำหรับจัดเก็บข้อมูลจากการสร้างคอมโพเน้นต์ประเภท Model (Entity Class) ที่ทำการกำหนดโครงสร้างข้อมูล ที่จะ Map ไปเป็น Data Table ในฐานข้อมูลให้อัตโนมัติ หลังจากนั้นจะทำการสร้าง Component ประเภท Data Access Object หรือ DAO ที่ทำหน้าที่ให้บริการจัดการข้อมูล โดย จะสืบค้นข้อมูลและดึงออกมาเป็นผลลัพธ์ในรูปแบบ Object ที่ Components ต่างๆสามารถนำไปใช้ประโยชน์ได้เช่น REST API สามารถนำ Object ที่ค้นคืนได้จาก Database ส่งคืนกลับไปให้ View แสดงผลได้

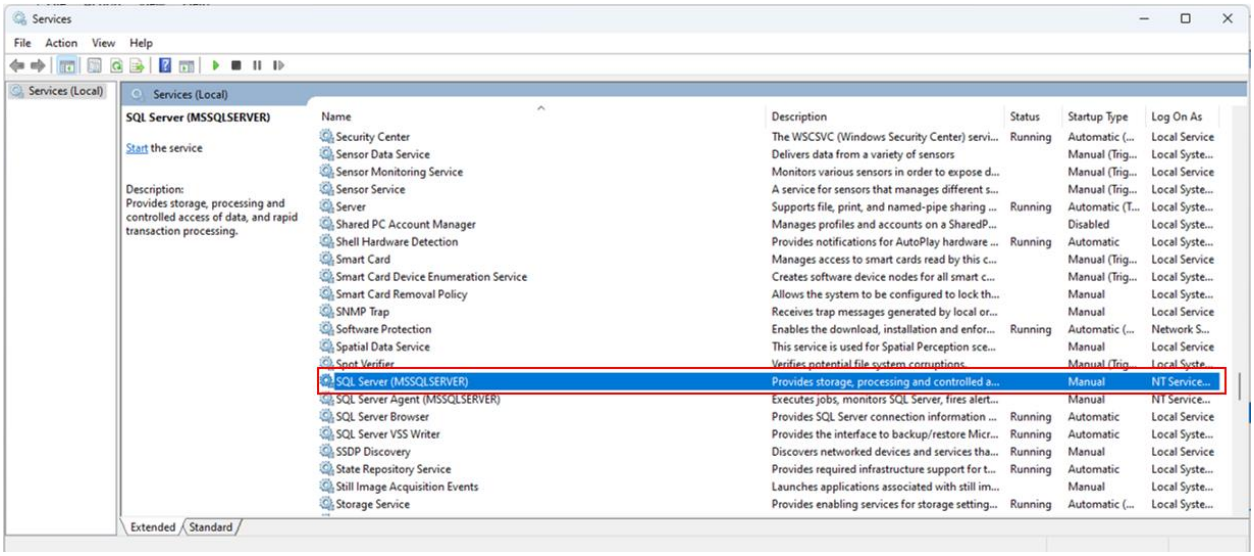
ขั้นตอนที่ 0 รูปแบบที่ 1 เปิด Services ที่เกี่ยวข้องกับ SQL Database ก่อน โดยการเปิดหน้าต่าง Services แล้วสั่ง start services ที่เกี่ยวข้องกับ SQL ให้มีสถานะเป็น running ดังรูปที่ 2



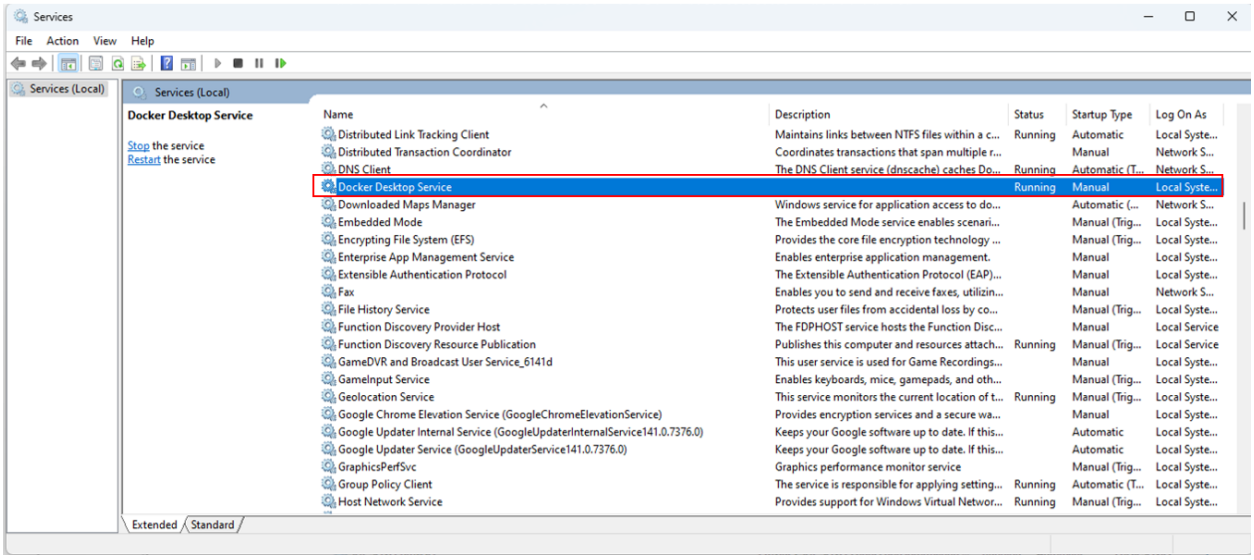
รูปที่ 2 การเปิด Service "SQL Database" ให้ทำงาน

ขั้นตอนที่ 0 รูปแบบที่ 2 ใช้ Docker

- เปิด Services ที่เกี่ยวข้องกับ SQL Database ก่อน โดยการเปิดหน้าต่าง Services แล้วสั่ง stop services ที่เกี่ยวข้องกับ SQL ให้มีสถานะเป็น running

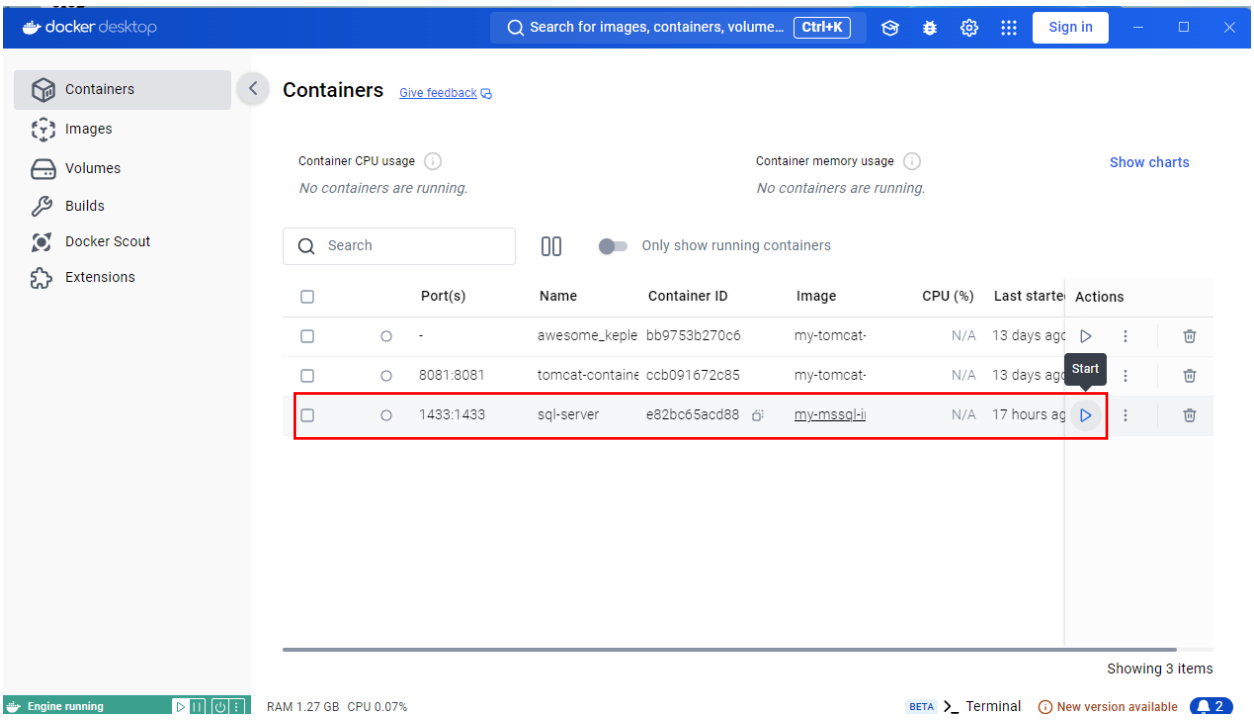


การปิด Service "SQL Database" ไม่ให้ทำงาน

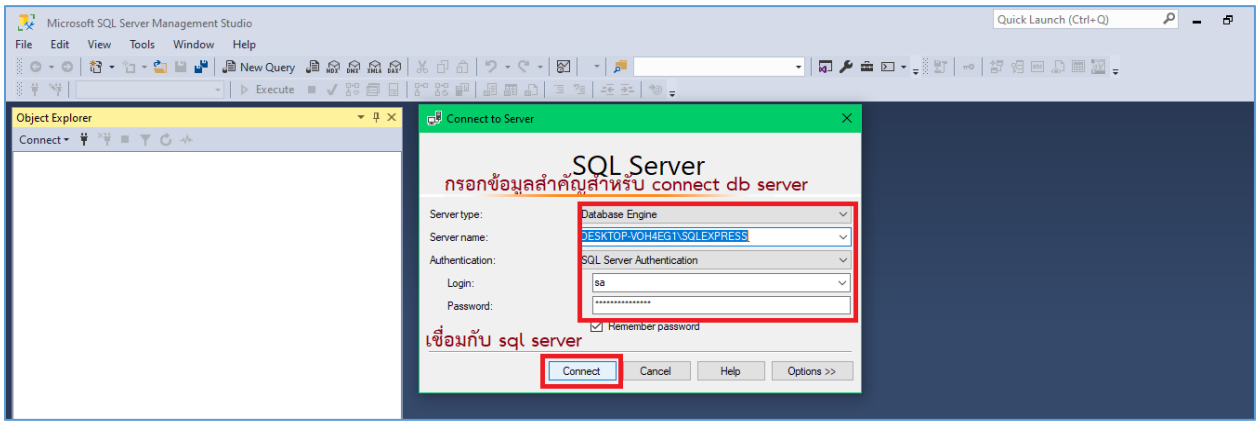


การเปิด Service " Docker Destop Service " ให้ทำงาน

- จากนั้นขั้นตอนที่ 0 รูปแบบที่ 2 ให้เปิด Docker Desktop แล้ว start contrainer sql server ดังรูป ** ห้ามปิด Docker Desktop ในระหว่างทำงาน**

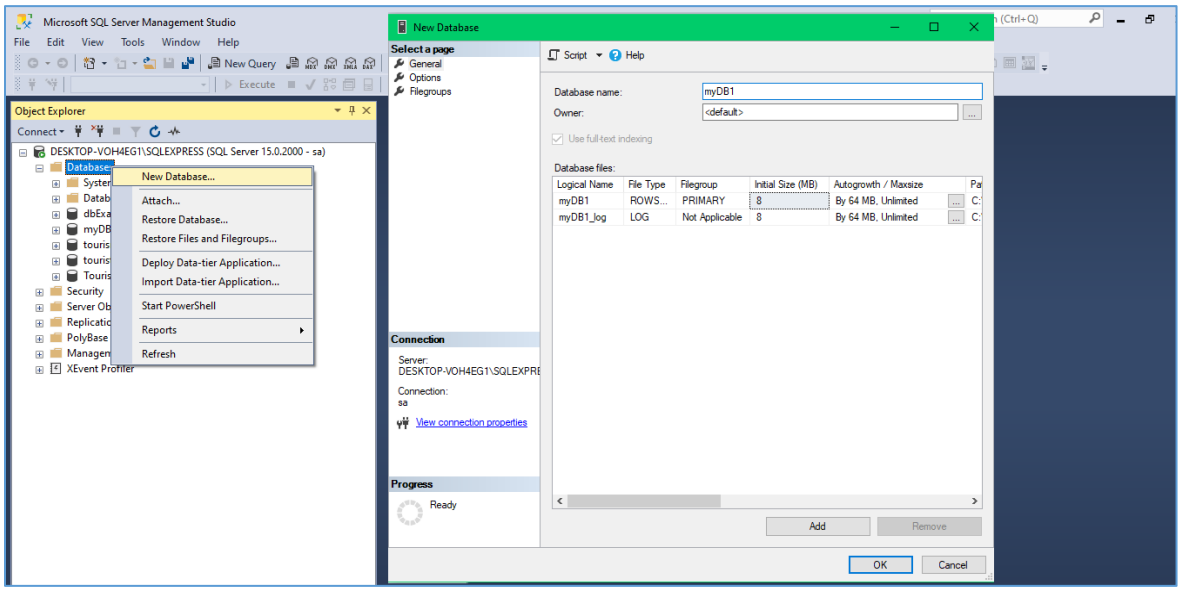


จากนั้นทั้ง 2 รูปแบบ ให้ใช้เครื่องมือประเภท SQL Client ในการเชื่อมต่อเข้าไปที่ SQL Server (ในที่นี้บน Windows จะใช้ SSMS เป็นเครื่องมือ SQL Client เชื่อมต่อฐานข้อมูล) เพื่อสร้างฐานข้อมูลชื่อ myDB1 เป็นฐานข้อมูลเปล่า ยังไม่มีตารางข้อมูลใดใด ให้นักศึกษาจำว่าใช้ User Account และ Password ใดในการเชื่อมต่อกับ SQL Server สำเร็จ (**ขั้นตอนที่ 0 รูปแบบที่ 1** ใช้ user เป็น “sa” และ password เป็น “รหัสผ่านที่นักศึกษาตั้งตอนลงโปรแกรม Sql Server” **ขั้นตอนที่ 0 รูปแบบที่ 2** ใช้ Docker เครื่องในห้อง Lab ใช้ user เป็น “sa” และ password เป็น “YourStrong@Passw0rd”) รูปแสดงหน้าจอการ connect database และรูปที่ 4 แสดงการสร้าง Database เปล่า ให้นักศึกษากรอกชื่อ database เช่น “myDB1”



รูปที่ 3 ใช้เครื่องมือชื่อ SSMS เป็น SQL Client ทดสอบการ connect กับ SQL database server

หมายเหตุ กรณีลืมรหัสผ่าน SQL ให้ดูวิธีแก้ไขหน้า 18



รูปที่ 4 สร้าง Database ชื่อ myDB1 เป็นฐานข้อมูลเริ่มต้นเปล่าๆ

เพิ่มการประกาศค่า config สำหรับเชื่อมต่อกับ database **ในไฟล์ application.properties** ตามโค้ดด้านล่าง สิ่งที่ประกาศเพิ่มเติม ได้แก่ URL ของ Database Server ชื่อฐานข้อมูล “myDB1” user name และ password สำหรับ connect กับ database

```
# application.properties
spring.application.name=demo
server.port=8080

# Microsoft SQL Server configuration
spring.datasource.driverClassName=com.microsoft.sqlserver.jdbc.SQLServerDriver
#spring.datasource.url=jdbc:sqlserver://<host>[:<port>][;instanceName=<instance>][;databaseName=<db>][;property=value]
#spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=myDB1;encrypt=true;trustServerCertificate=true;
spring.datasource.url=jdbc:sqlserver://DESKTOP-VOH4EG1:1433;instanceName=SQLEXPRESS;databaseName=myDB1;encrypt=true;trustServerCertificate=true;

spring.datasource.username=sa
spring.datasource.password= ให้ใส่ Your MS SQL password

# Enable SQL logging for debugging (optional)
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql =true

## Hibernate Properties
# The SQL dialect makes Hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.SQLServer2012Dialect

# Hibernate ddl auto (create, create-drop, validate, update)
spring.jpa.hibernate.ddl-auto =update
```

<u>ขั้นตอนที่ 1</u> เพื่อสร้างตารางข้อมูลเก็บ User ซึ่งประกอบด้วยข้อมูลชื่อ นามสกุล email เพิ่มลงในฐานข้อมูล myDB1 นักศึกษาต้องใช้ Libraries หรือ Maven Dependencies เพิ่มเติมให้ทำการเพิ่ม dependencies ต่อไปนี้ ลงในไฟล์ pom.xml

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>

<dependency>
        <groupId>com.microsoft.sqlserver</groupId>
        <artifactId>mssql-jdbc</artifactId>
        <version>11.2.1.jre17</version>
</dependency>

<dependency>
    <groupId>javax.persistence</groupId>
    <artifactId>persistence-api</artifactId>
    <version>1.0.2</version>
 </dependency>

<dependency>
    <groupId>org.eclipse.persistence</groupId>
    <artifactId>javax.persistence</artifactId>
    <version>2.2.1</version>
 </dependency>
```

ให้สร้าง java class ชื่อ “User” เพื่อให้ Spring นำไปสร้างเป็น Data Table ใน SQL Server ให้เราอัตโนมัติ โดยชื่อAttributes ใน Java Class จะถูกนำไปสร้างเป็นคอลัมน์ใน Data Table โค้ด Java Class ชื่อ “User” เป็นดังนี้

```
package com.example.demo.model;

import jakarta.persistence.*;
import lombok.Data;

@Data
@Entity
@Table(name = "users")
public class User {
      @Id
      @GeneratedValue(strategy = GenerationType.IDENTITY)
      private Long id ;
      @Column(name = "first_name", nullable = false)
      private String firstName ;
      @Column(name = "last_name", nullable = false)
      private String lastName ;
      @Column(name = "email", nullable = false, unique=true)
      private String email ;
```

```
	public Long getId() {
		return id;
	}
	public void setId(Long id) {
		this.id = id;
	}
	public String getFirstName() {
		return firstName;
	}
	public void setFirstName(String firstName) {
		this.firstName = firstName;
	}
	public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
		this.lastName = lastName;
	}
	public String getEmail() {
		return email;
	}
	public void setEmail(String email) {
		this.email = email;
	}
	public User(Long id, String firstName, String lastName, String email) {
		super();
		this.id = id;
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
	}
	public User() {
		super();
		// TODO Auto-generated constructor stub
	}
	public User(String firstName, String lastName, String email) {
		super();
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
	}


}
```

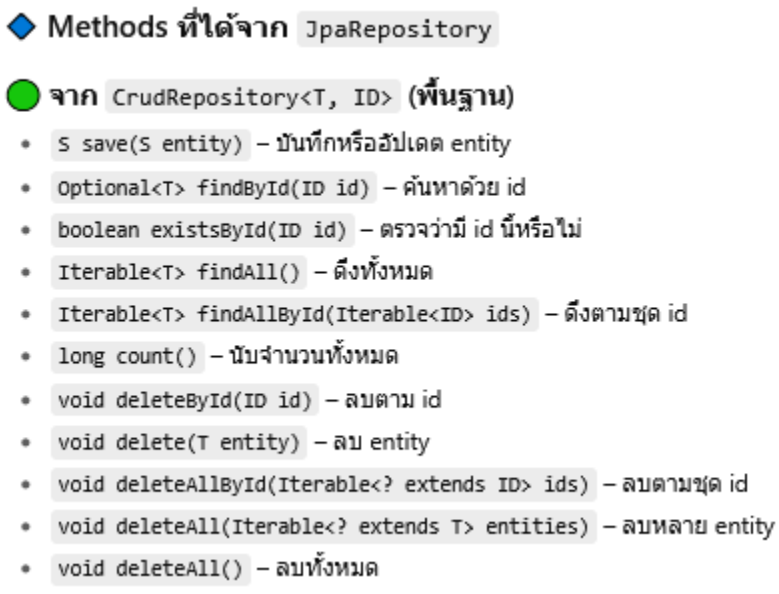
จากโค้ดตัวอย่าง เราใช้ @Data และ @Entity @Table ในการกำหนดให้คลาส map กับ table ตารางให้คลาสมีเมท็อด getter และ setter ถูกสร้างอัตโนมัติ ไม่ต้องประกาศเอง มีการกำหนด @Id เพื่อกำหนดคอลัมน์ข้อมูลที่เป็น Primary Key (ฟิลด์ที่ใช้ identify แยกข้อมูลที่ต่างกันด้วยฟิล์ด Primary Key ค่าห้ามซ้ำ

กัน) นอกจากนั้นมีการใช้ @Column กำหนดชื่อคอลัมน์และเงื่อนไขหรือ constraints ของคอลัมน์ได้แก่ห้ามเป็นค่า null ค่าห้ามซ้ำกับข้อมูล row อื่น

ขั้นตอนที่ 2 ให้สร้าง Component ประเภท Repository ที่ทำหน้าที่เป็น Data Access Object โดยประกาศเป็น Java Interface ดังโค้ดด้านล่าง เพียงเท่านี้ Component DAO มี methods สำหรับจัดการข้อมูลครอบคลุมทั้งสืบค้น บันทึกข้อมูลใหม่ ลบข้อมูล แก้ไขเปลี่ยนแปลงค่าข้อมูล มาให้เรียกใช้แบบอัตโนมัติ โดยไม่ต้องเขียนโค้ดใดใด

```
package com.example.demo.repo;
import com.example.demo.model.User;
import java.util.List;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
	List<User> findByFirstName(String firstName);
	List<User> findByEmailContaining(String keyword);
}
```

ได้แก่



🟢 จาก `PagingAndSortingRepository<T, ID>`

- `Iterable<T> findAll(Sort sort)` – ดึงทั้งหมดพร้อมเรียงลำดับ
- `Page<T> findAll(Pageable pageable)` – ดึงทั้งหมดแบบแบ่งหน้า

🟢 จาก `JpaRepository<T, ID>`

- `List<T> findAll()` – คืน `List` แทน `Iterable`
- `List<T> findAll(Sort sort)`
- `List<T> findAllById(Iterable<ID> ids)`
- `void flush()` – flush persistence context
- `S saveAndFlush(S entity)` – save แล้ว flush ทันที
- `List<S> saveAllAndFlush(Iterable<S> entities)`
- `void deleteAllInBatch()`
- `void deleteAllInBatch(Iterable<T> entities)`
- `T getOne(ID id)` *(deprecated → ใช้ `getById`)*
- `T getById(ID id)` – lazy load proxy
- `<S extends T> List<S> findAll(Example<S> ex e)` – query by Example
- `<S extends T> List<S> findAll(Example<S> example, Sort sort)`

นอกจากนี้ Spring Data JPA ยัง **generate method อัตโนมัติ** ตามชื่อ method ที่เราประกาศเพิ่ม เช่น:

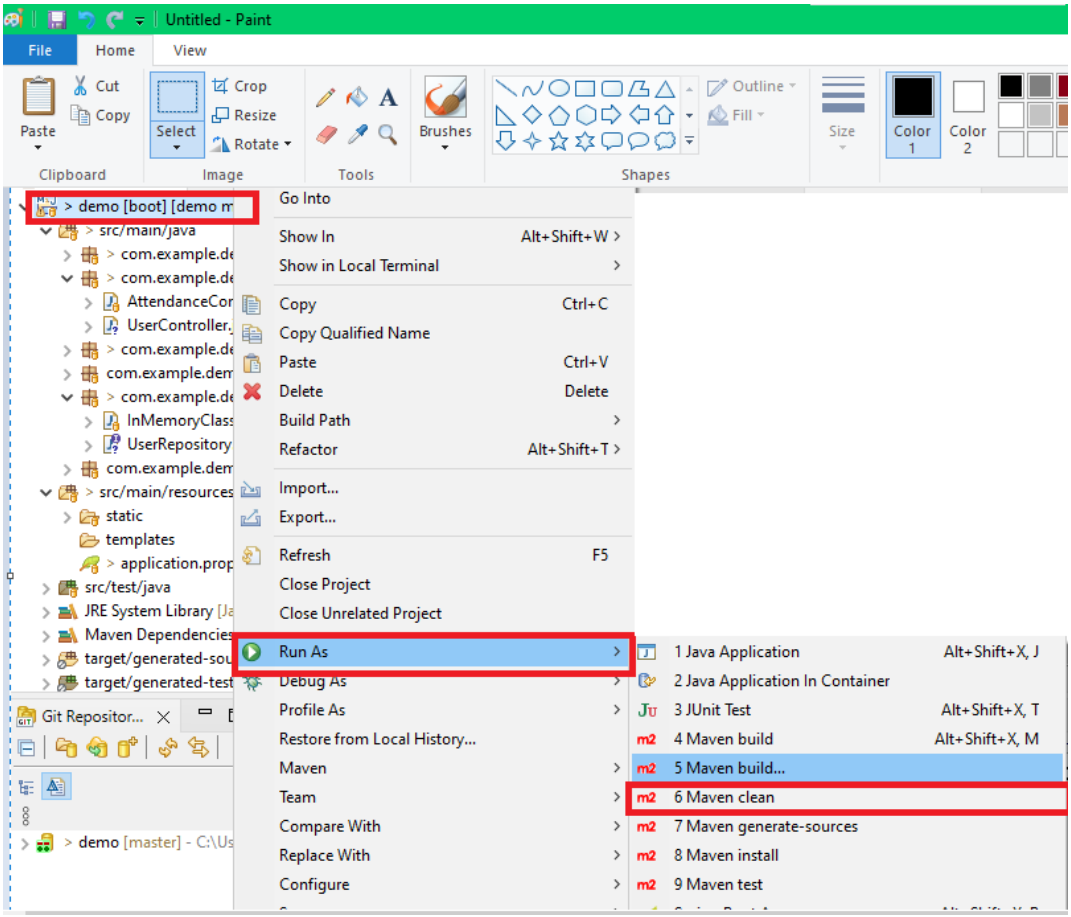
```
List<User> findByFirstName(String firtsName);

List<User> findByEmailContaining(String keyword);
```

หมายเหตุ ชื่อพารามิเตอร์ของ method ต้องสอดคล้องกับชื่อ attribute ของคลาส User เช่น พารามิเตอร์ “firstName” ชื่อ attribute ของคลาส User ต้องมีชื่อ “fisrtName” ด้วยเช่นกัน

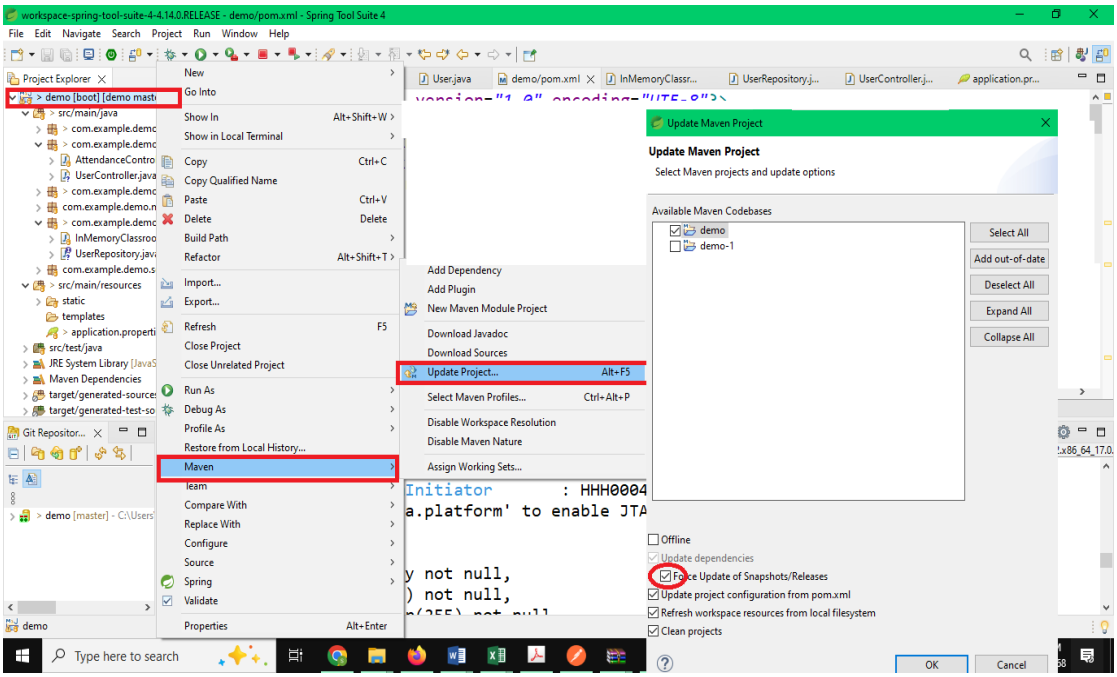
ขั้นตอนที่ 3 ให้สร้าง Component ประเภท REST Controller ที่เรียกใช้ Repository ในการจัดการ Data ประเภท User ให้ ตัว Controller ทำหน้าที่เป็น API ให้ แต่ตัวที่จะต่อ Database สร้างคำสั่ง SQL Query เพื่อจัดการข้อมูลในฐานข้อมูลคือ Repository

```
package com.example.demo.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import com.example.demo.model.User;
import com.example.demo.repo.UserRepository;
import java.util.List;

@RestController
@RequestMapping("/api/users")
public class UserController {

 @Autowired
 private UserRepository userRepository;

 @GetMapping
 public List<User> getAllUsers() {
  return userRepository.findAll();
    }

 @PostMapping
 public User createUser(@RequestBody User user) {
  return userRepository.save(user);
    }
}
```

ขั้นตอนที่ 4 ให้ clean โดยคลิกเม้าส์ขวาที่ชื่อ project เลือกใช้เมนู Run As > maven clean (ดูรูปที่ 5) จากนั้นให้ทำการ force update project อีกครั้ง โดยคลิกเม้าส์ขวาที่ชื่อ project เลือกใช้เมนู Maven > Update Project จากนั้นให้ check box ที่ force update เพื่อบังคับให้ maven download dependencies มาใหม่ (ดู รูปที่ 6) จากนั้น สั่งรัน Spring Boot App อีกครั้ง โดย คลิกเม้าส์ขวาที่ชื่อ Project เลือกเมนู Run As > Spring Boot App สังเกตผลการรัน Spring Boot App ในหน้า console รูปที่ 7

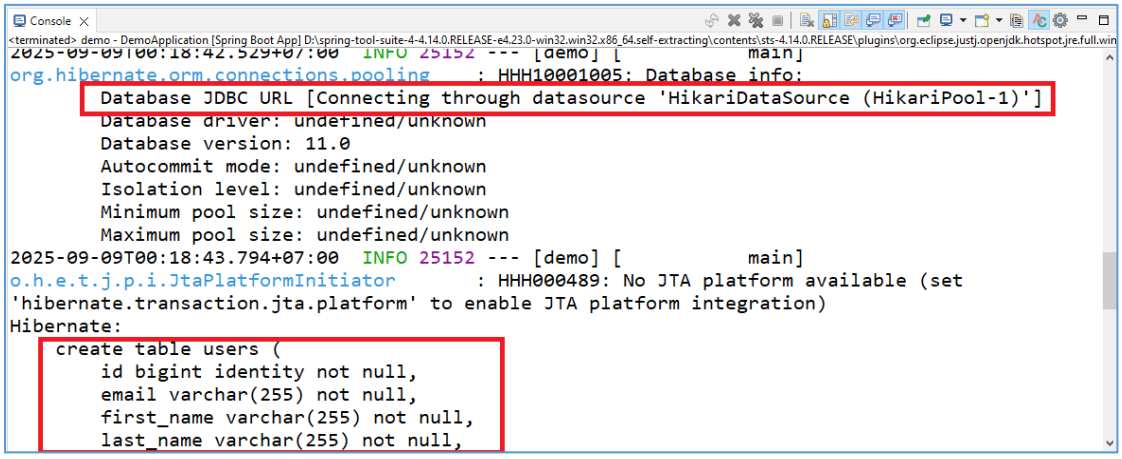


รูปที่ 5 การรัน maven clean เพื่อล้างสิ่งที่ได้ compiled ก่อนหน้า



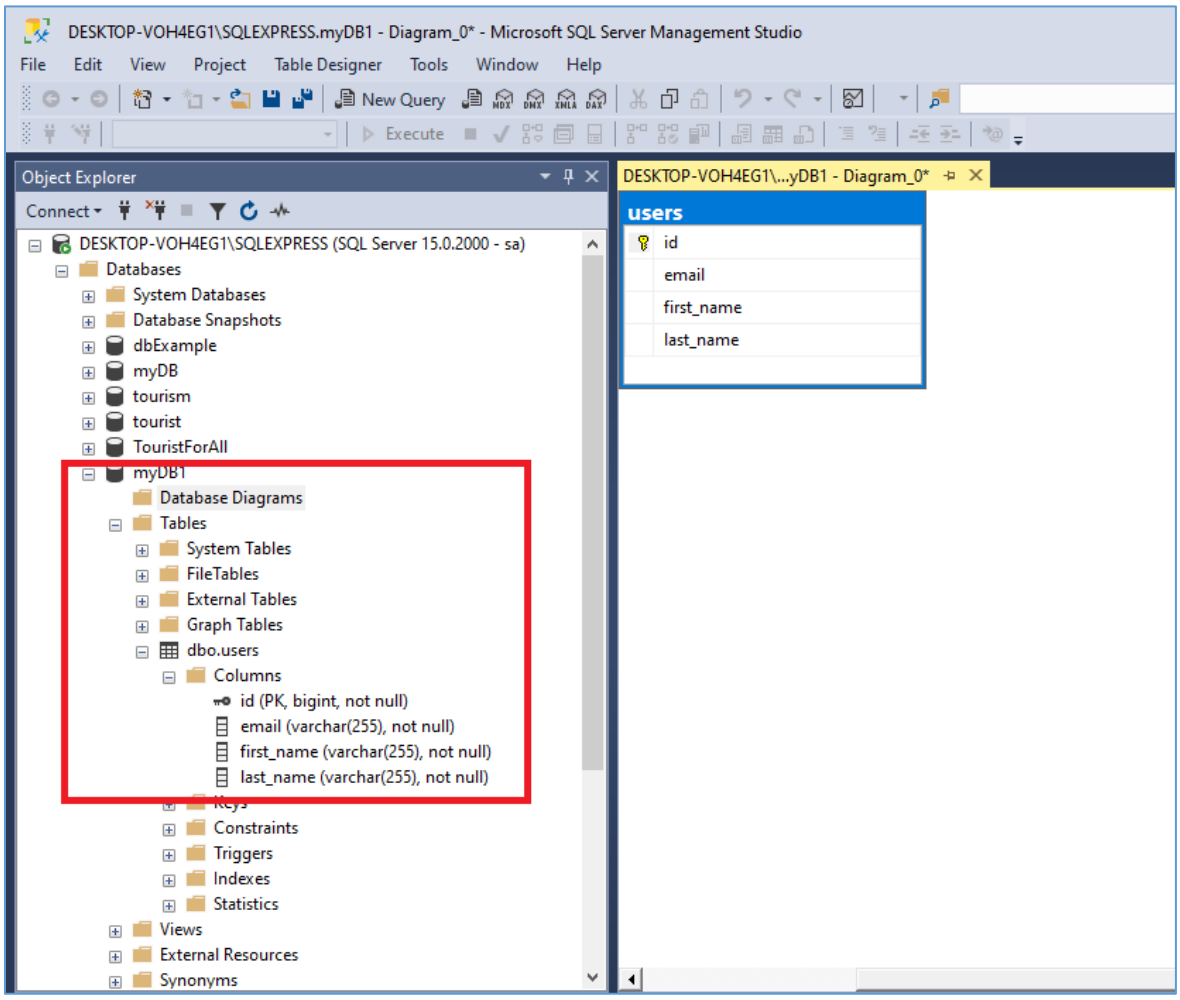
รูปที่ 6 รันคำสั่ง force update project ให้ reload dependencies ใหม่

ในหน้า console เห็นข้อความแสดงให้เห็นคำสั่งการสร้าง Data Table ให้อัตโนมัติตามโครงสร้างของ Model Class “User” ดังรูปที่ 7



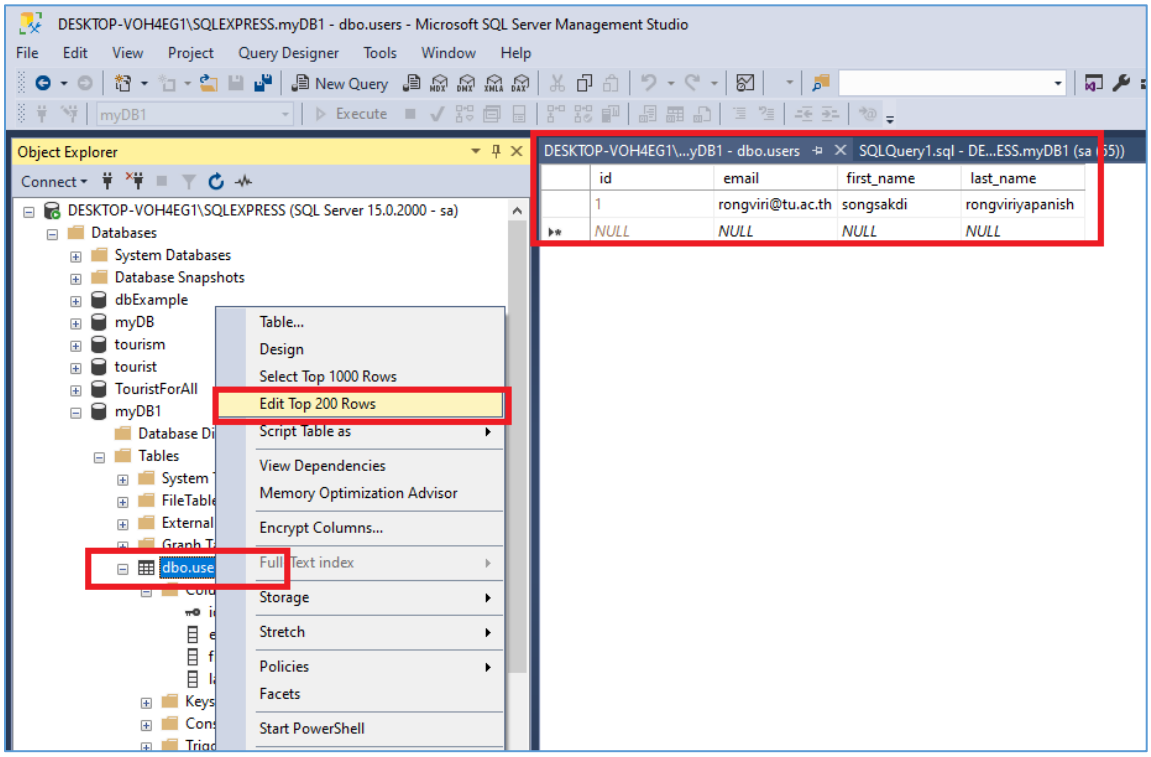
รูปที่ 7 หน้า console แสดงให้เห็น Spring Boot App สั่งให้สร้าง Data Table อัตโนมัติจาก Model Class

ขั้นตอนที่ 5 ให้นักศึกษาเช็คใน Database “myDB1” จะพบว่ามีตาราง “Users” ถูกสร้างขึ้นมาพร้อม columns ต่างๆ ได้แก่ id, first_name, last_name, email ดังรูปที่ *8* เป็นตารางเปล่าที่ไม่มีข้อมูล



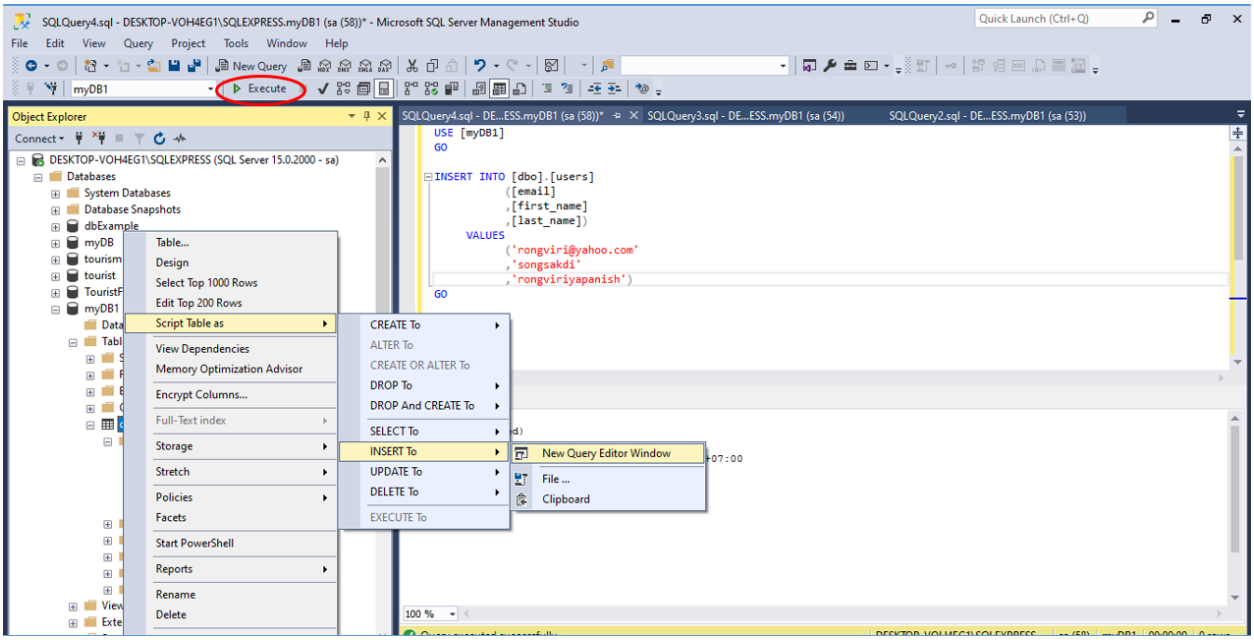
รูปที่ 8 เกิดตารางข้อมูล "users" ถูกสร้างอัตโนมัติใน Database "myDB1" จากการรัน Spring Boot App

ขั้นตอนที่ 6 ให้นักศึกษาเพิ่มข้อมูลตนเอง ได้แก่ ชื่อ นามสกุล email ของ tu ของนักศึกษาลงไปในตารางโดยการคลิกเม้าส์ขวาที่ชื่อตาราง “Users” เลือกเมนู edit top xxxx rows จะมีตารางให้กรอกรายการข้อมูลใหม่เพิ่ม ให้กรอกทุกคอลัมน์ยกเว้น Id ระบบจะ generate รหัสให้อัตโนมัติหลังกด enter เมื่อกรอกข้อมูลเสร็จ แล้วกด Enter ดังรูปที่ *9*



รูปที่ 9 ทดสอบคีย์ข้อมูล User ใหม่ใส่ตาราง users

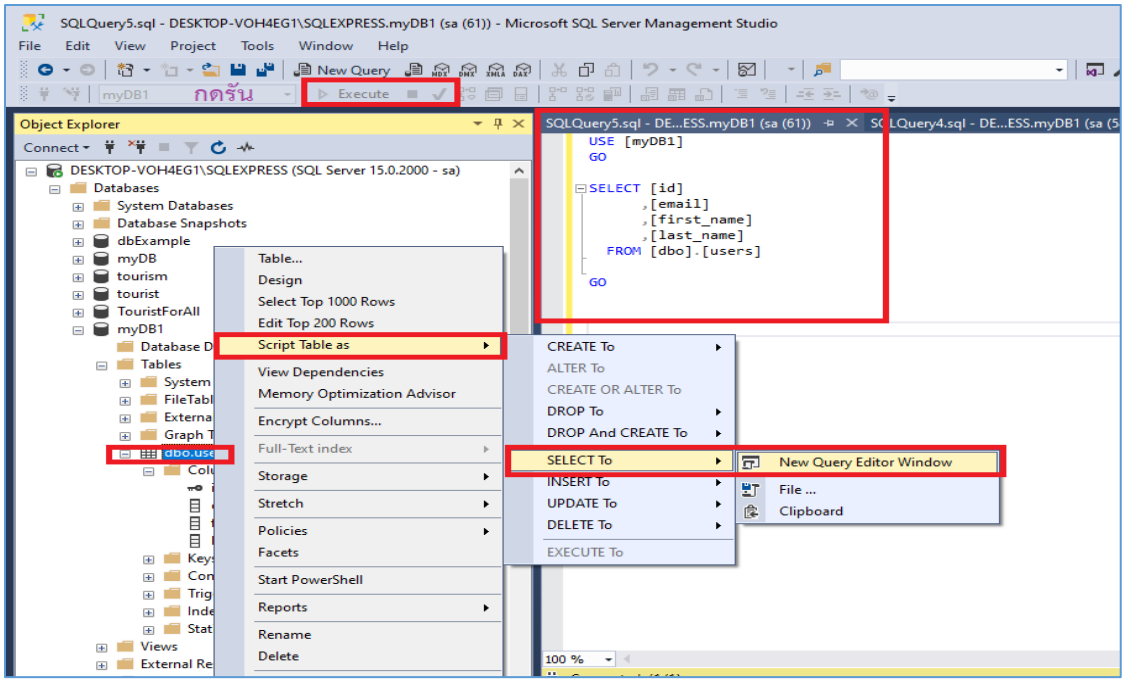
ทดลอง insert ข้อมูลรายการใหม่ด้วยคำสั่ง SQL โดยคลิกเม้าส์ขวาที่ชื่อตาราง User จากนั้นเลือกเมนู Script Table as > Insert To > New Query Editor Window เพื่อพิมพ์คำสั่งเพิ่มรายการข้อมูลใหม่ แล้วกดปุ่ม Execute ดังรูปที่ *10*



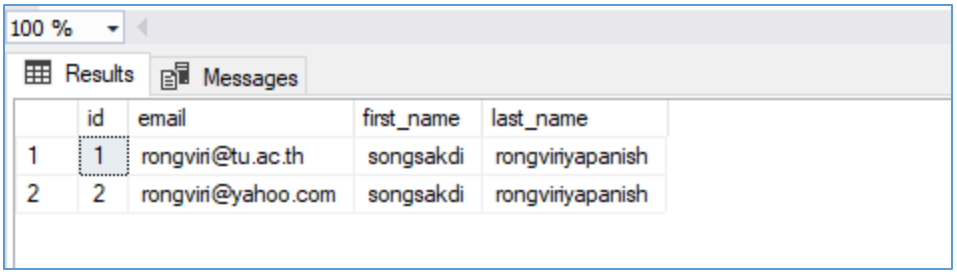
รูปที่ 10 ทดสอบเพิ่มข้อมูล user ใหม่ด้วยคำสั่ง SQL INSERT

ทดสอบสืบค้นข้อมูลทั้งหมดจากตารางโดยกดเลือก เมนู Scrip Table as > Select to > New Query Editor Window ตามรูปที่ *11*

ในหน้าต่างมีคำสั่ง select ที่ค้นหาข้อมูลจากตาราง users ตามเงื่อนไขที่เรากำหนด ให้กด “Execute” ได้ผลลัพธ์ดังรูปที่ *12*

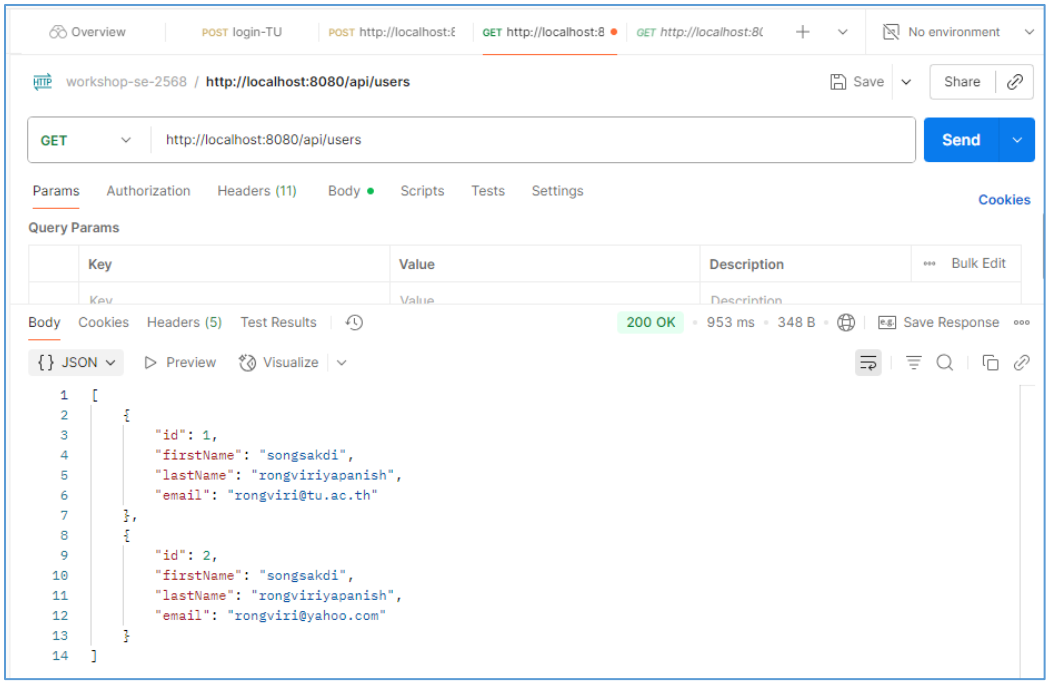


รูปที่ 11ทดสอบดึงข้อมูลทั้งหมดจากตาราง users ด้วยคำสั่ง SQL SELECT



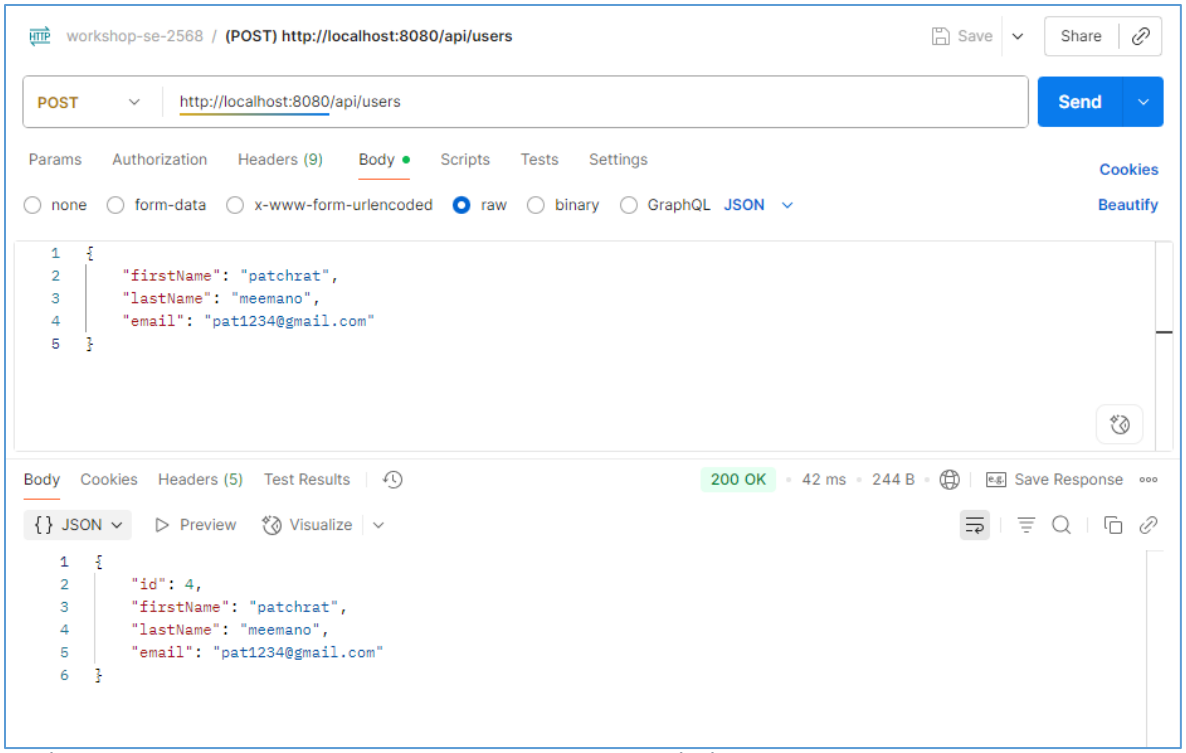
รูปที่ 12 ผลลัพธ์แสดงรายการข้อมูล user ทั้งหมดที่ได้จากการ query ด้วย SELECT

<u>ขั้นตอนที่ 7</u> ให้ลองเรียกใช้ REST API เพื่อดึงข้อมูลทั้งหมดจากตาราง Users โดยใช้ API ประเภท GET Method ที่ประกาศในคลาส UserController โดยใช้เครื่องมือ POSTMAN (ดูรูปที่ *13*)

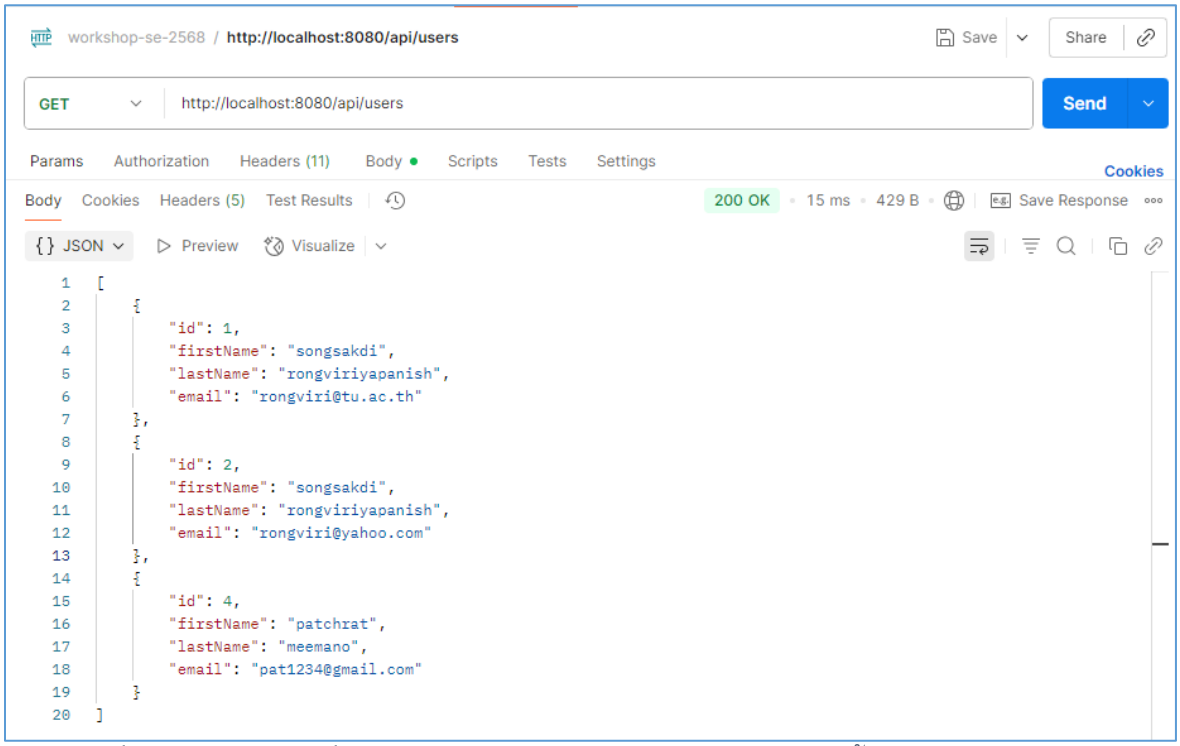


รูปที่ 13 ทดสอบ REST API ประเภท GET เพื่อดึงข้อมูล Users จากตาราง โดยใช้ POSTMAN

ขั้นตอนที่ 8 (รูปที่ *14*) ให้ลองเรียกใช้ REST API เพื่อเพิ่มข้อมูล User รายการใหม่เพิ่มลงในตาราง Users โดยใช้ API ประเภท POST Method ที่ประกาศในคลาส UserController โดยใช้เครื่องมือ POSTMAN เพิ่มข้อมูล User ใหม่ที่ส่งให้ POST METHOD ในรูปแบบ json แล้วเรียก API ประเภท GET METHOD อีกครั้งเพื่อดึงรายการข้อมูล user ทั้งหมดอีกครั้ง (ดูรูปที่ *15*)



รูปที่ 14 ทดสอบ REST API ประเภท POST METHOD ที่เพิ่มรายการข้อมูล user ใหม่ โดยใช้ POSTMAN



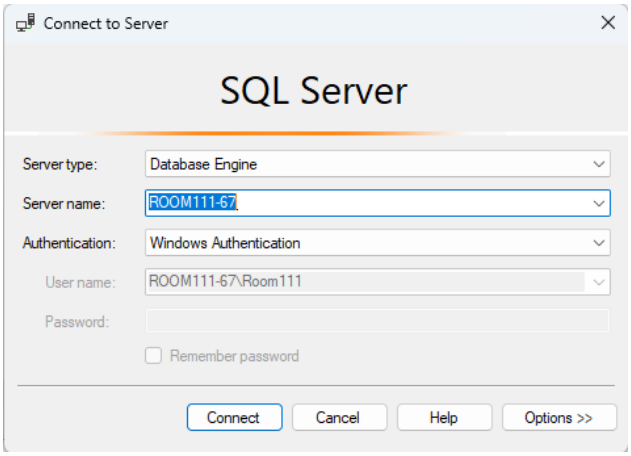
รูปที่ 15 ผลหลังการเพิ่มรายการ user ใหม่ แล้วดึงรายการ users ทั้งหมดด้วย GET METHOD

วิธีแก้ไขกรณีลืมรหัสผ่าน SQL

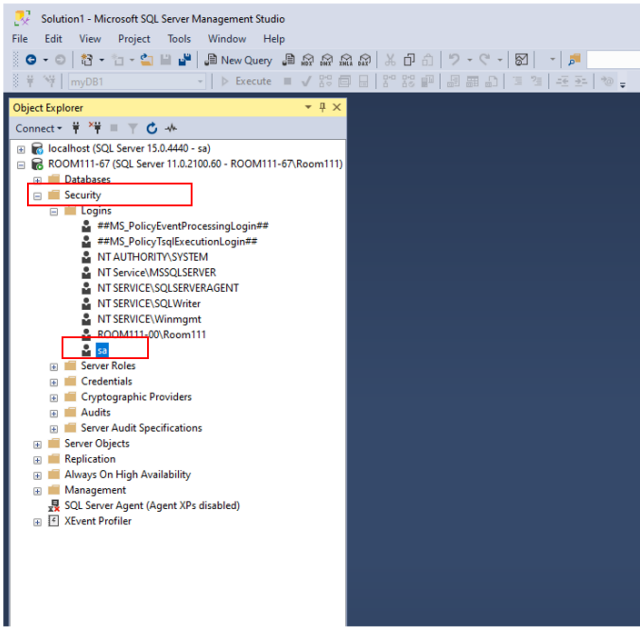
กรณีลืมรหัสผ่าน SQL สามารถแก้ไขได้ตามขั้นตอนต่อไปนี้

1.เปิด SSMS เป็นเครื่องมือ SQL Client เชื่อมต่อฐานข้อมูล

2.หลังจากที่คลิกแล้วจะเจอหน้าต่างConnect to Serverให้กำหนด “Authentication” > “Windows Authentication > กด Connect



3.เลือกที่ Folder “Security” แล้วเลือกคลิกที่ “sa”



4.หลังจากนั้นระบบจะแสดงหน้า Login: sa ให้ผู้ใช้ระบุ Password ที่ต้องการ set และ Confirm password ให้ตรงกันเสร็จแล้วกดปุ่ม OK จะได้รหัสผ่านใหม่เพื่อนำไปในงานเข้าสู่ระบบสำหรับ user ‘sa’