**Running Your Spring Boot Application with SQL Server**

This document outlines how to run your "demo" Spring Boot application, detailing the database setup and application configuration based on the SQL script and application.properties you provided.

**Prerequisites**

Before you start, make sure you have these installed:

* **Java Development Kit (JDK) 21 or higher:** Your application is built with Java 21.
* **Maven:** This project uses Maven for dependency management and building.
* **SQL Server Database:** The application connects to a SQL Server instance. Ensure your SQL Server is running and accessible (typically on localhost:1433).
* **An IDE (Optional but Recommended):** Integrated Development Environments like IntelliJ IDEA, Eclipse, or Visual Studio Code can make this easier.

**Database Setup**

Your application connects to a SQL Server database named ProductManagement. Follow these steps to set up your database:

1. **Execute the SQL Script:** Open SQL Server Management Studio (SSMS) or any SQL Server client and execute the following script to create your database and table:

SQL

CREATE DATABASE ProductManagement;

GO

USE ProductManagement;

GO

CREATE TABLE tbl\_Product(

id INT PRIMARY KEY IDENTITY(1, 1),

p\_name NVARCHAR(255),

price FLOAT,

quantity INT,

p\_description NVARCHAR(255)

);

This script creates the ProductManagement database and the tbl\_Product table with columns id, p\_name, price, quantity, and p\_description.

1. **Verify Database Credentials:** Your application.properties file specifies the connection details. Ensure your SQL Server instance is configured with the following credentials:
   * **Server:** localhost:1433
   * **Database Name:** ProductManagement
   * **Username:** sa
   * **Password:** 123456

Here's your application.properties for reference:

Properties

# Server port configuration (default is 8080)

server.port=8080

# SQL Server connection configuration

spring.datasource.url=jdbc:sqlserver://localhost:1433;databaseName=ProductManagement;encrypt=true;trustServerCertificate=true;

spring.datasource.username=sa

spring.datasource.password=123456

spring.datasource.driver-class-name=com.microsoft.sqlserver.jdbc.SQLServerDriver

# Show SQL queries for debugging

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.format\_sql=true

# Retain table and column names as in the entity (do not convert to snake\_case)

spring.jpa.hibernate.naming.physical-strategy=org.hibernate.boot.model.naming.PhysicalNamingStrategyStandardImpl

**Important Note:** The price column in your tbl\_Product table is defined as FLOAT. If your ProductStatistics DTO expects Double for maxPrice and minPrice, you'll still encounter the Missing constructor error. To resolve this, you either need to change price in Product.java to Double (and migrate your DB column type) OR change maxPrice and minPrice in ProductStatistics.java's constructor and fields to Float.

**Running the Application**

Follow these steps to run your application:

1. **Navigate to the Project Directory:** Open your command prompt or terminal and change the directory to your project's root folder (C:\Users\Admin\Desktop\ktrajavaweb\demo).

PowerShell

cd C:\Users\Admin\Desktop\ktrajavaweb\demo

1. **Clean and Build the Project:** Execute the Maven clean and install command. This compiles your code, runs tests, and packages the application into a JAR file.

PowerShell

mvn clean install

This command will download dependencies, compile source code, and create a JAR file (e.g., demo-0.0.1-SNAPSHOT.jar) in the target directory.

1. **Run the Application:** Once the build is successful, you can run the Spring Boot application using the java -jar command:

PowerShell

java -jar target\demo-0.0.1-SNAPSHOT.jar

*(If the JAR file name is different, replace demo-0.0.1-SNAPSHOT.jar with the actual name of your generated JAR file.)*

**Verifying the Application Status**

When your application starts successfully, you'll see logs in your console similar to this, indicating that Tomcat is initialized on port 8080 and your JPA entities are being managed:

2025-07-15T11:XX:XX.XXX+07:00 INFO XXXX --- [ main] o.s.boot.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)

...

2025-07-15T11:XX:XX.XXX+07:00 INFO XXXX --- [ main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

...

If you encounter an "Application run failed" error, especially related to a "Missing constructor," ensure you've resolved the data type mismatch between Product.price (which is FLOAT) and ProductStatistics's constructor arguments (which you might have set to Double for maxPrice and minPrice). A mvn clean install is crucial after any code changes.

Once the application starts without errors, it will be accessible via your web browser or API client at http://localhost:8080.