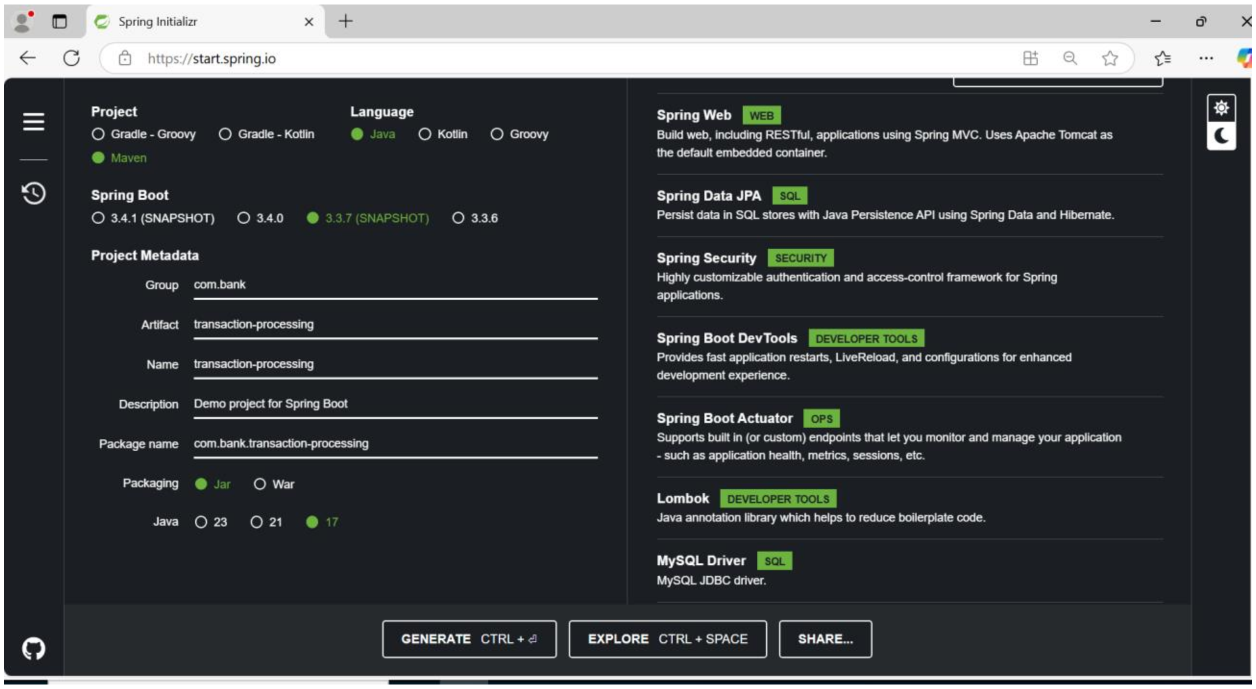
Banking Case Study: Real-Time Transaction Processing

Scenario: A large, multinational bank is seeking to modernize its legacy transaction processing systems to improve efficiency, scalability, and customer experience. The bank aims to implement a real-time transaction processing system capable of handling high transaction volumes while ensuring data integrity, security, and compliance with regulatory requirements. Additionally, the bank wants to implement a robust fraud detection system to identify and prevent fraudulent activities.

GitHub Link:

[harshapriyav6301/JavaCasestudy]()

SPRING INITILIZER



Project
Gradle - Groovy | Gradle - Kotlin | [✓ Maven]

Language
[✓ Java] | Kotlin | Groovy

Spring Boot
3.4.1 (SNAPSHOT) | 3.4.0 | [✓ 3.3.7 (SNAPSHOT)] | 3.3.6

Project Metadata

Group	com.bank
Artifact	transaction-processing
Name	transaction-processing
Description	Demo project for Spring Boot
Package name	com.bank.transaction-processing
Packaging	[✓ Jar] War
Java	23 21 [✓ 17]

Spring Web WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

Spring Data JPA SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

Spring Security SECURITY
Highly customizable authentication and access-control framework for Spring applications.

Spring Boot DevTools DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

Spring Boot Actuator OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc.

Lombok DEVELOPER TOOLS
Java annotation library which helps to reduce boilerplate code.

MySQL Driver SQL
MySQL JDBC driver.

INTELLIJ IDEA

1. TRANSACTION ENTITY:

```
package com.bank.transaction_processing;

import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;

@Entity
public class Transaction {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)  // Auto-generate the ID
    private Long id;
    private String transactionType;
    private Double amount;

    // Getters and Setters

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getTransactionType() {
        return transactionType;
    }

    public void setTransactionType(String transactionType) {
        this.transactionType = transactionType;
    }

    public Double getAmount() {
        return amount;
    }

    public void setAmount(Double amount) {
        this.amount = amount;
    }
}
```

2. TRANSACTION REPOSITORY

```
package com.bank.transaction_processing;

import org.springframework.data.jpa.repository.JpaRepository;

public interface TransactionRepository extends JpaRepository<Transaction, Long> {
}
```

3. TRANSACTION CONTROLLER

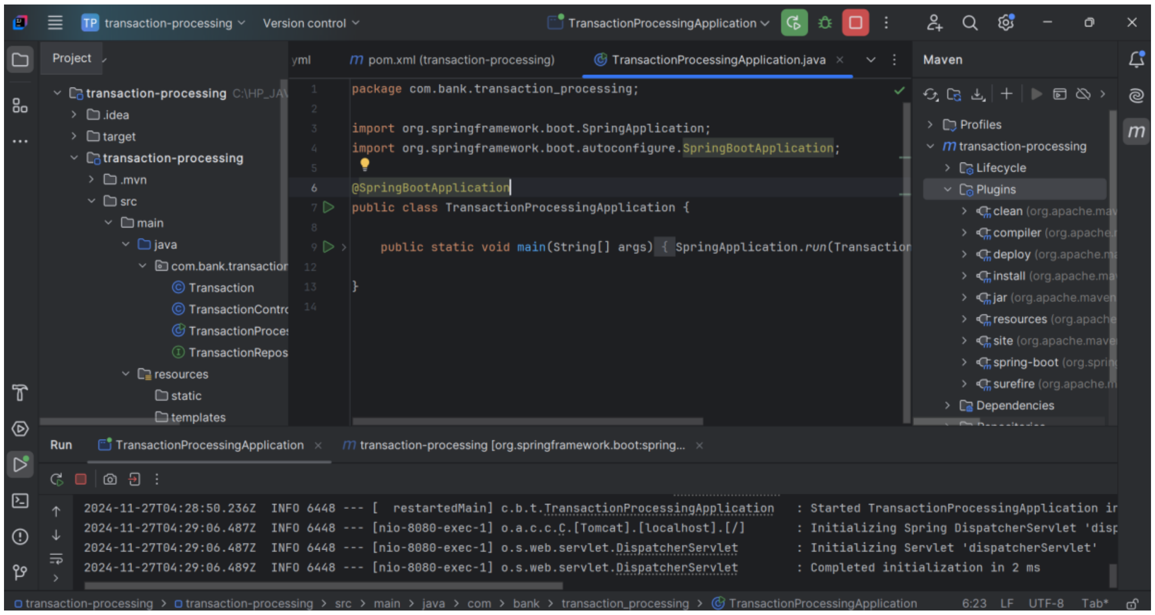
```
package com.bank.transaction_processing;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;
```

```
@RestController
@RequestMapping("/transactions")
public class TransactionController {

   @Autowired
   private TransactionRepository transactionRepository;

   // POST endpoint to create a new transaction
   @PostMapping
   public Transaction createTransaction(@RequestBody Transaction transaction) {
      return transactionRepository.save(transaction);
   }



   // GET endpoint to retrieve a transaction by its ID
   @GetMapping("/{id}")
   public Transaction getTransactionById(@PathVariable Long id) {
      return transactionRepository.findById(id).orElse(null);
   }

   // GET endpoint to retrieve a list of all transactions
   @GetMapping
   public List<Transaction> getAllTransactions() {
      return transactionRepository.findAll();
   }

   // PUT endpoint to update an existing transaction
   @PutMapping("/{id}")
   public Transaction updateTransaction(@PathVariable Long id, @RequestBody Transaction updatedTransaction) {
      return transactionRepository.findById(id)
            .map(transaction -> {
               // Update fields with the new values from updatedTransaction
               transaction.setTransactionType(updatedTransaction.getTransactionType());
               transaction.setAmount(updatedTransaction.getAmount());
               // Save the updated transaction
               return transactionRepository.save(transaction);
            })
            .orElse(null); // If not found, return null
   }

   // DELETE endpoint to delete a transaction by its ID
   @DeleteMapping("/{id}")
   public void deleteTransaction(@PathVariable Long id) {
      transactionRepository.deleteById(id);
   }
}
```

4. **APPLICATION.YML**

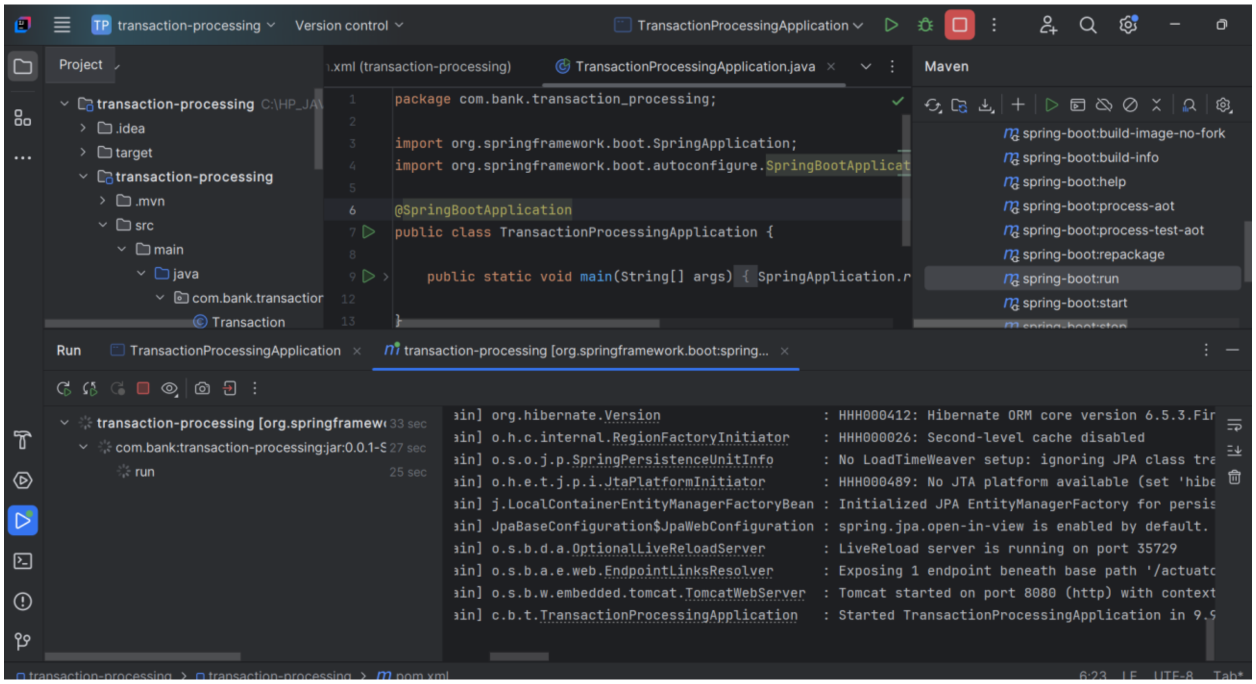
```
spring:
  datasource:
    url: jdbc:mysql://localhost:3306/bank_db
    username: root
    password: abcd
  jpa:
    hibernate:
      ddl-auto: update
```

OUTPUTS



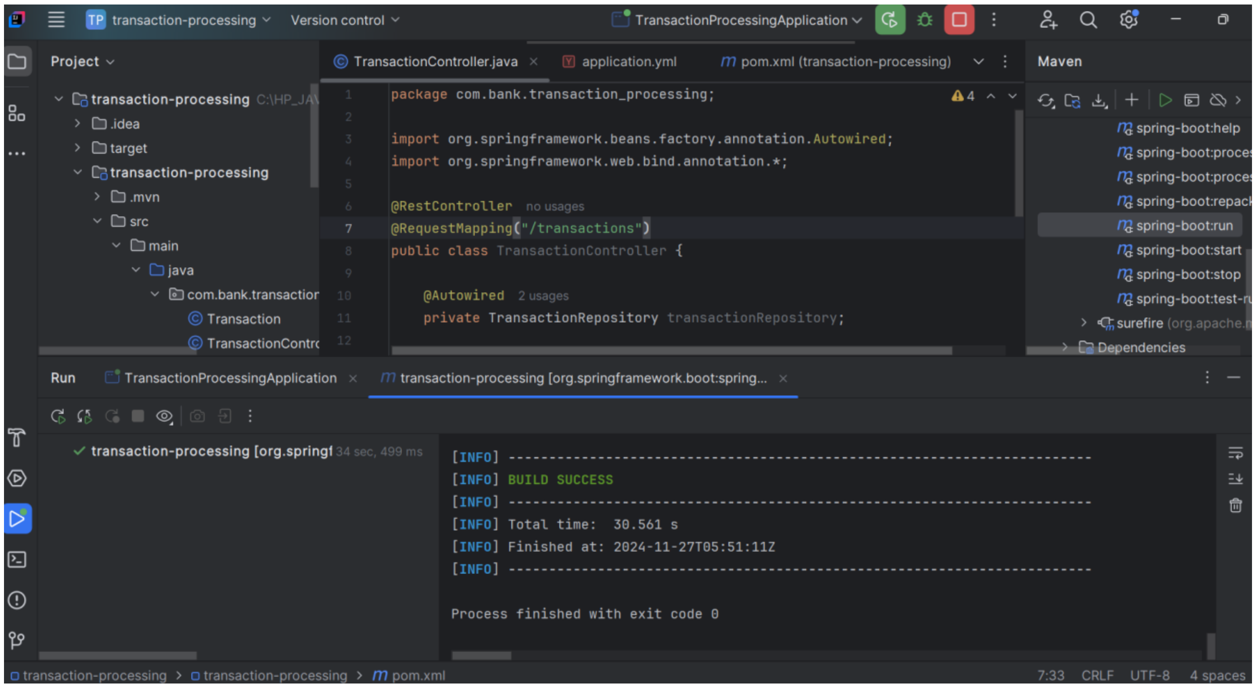
```
package com.bank.transaction_processing;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class TransactionProcessingApplication {

    public static void main(String[] args) { SpringApplication.run(Transactior

}
```

```
2024-11-27T04:28:50.236Z  INFO 6448 --- [  restartedMain] c.b.t.TransactionProcessingApplication   : Started TransactionProcessingApplication ir
2024-11-27T04:29:06.487Z  INFO 6448 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'disp
2024-11-27T04:29:06.487Z  INFO 6448 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
2024-11-27T04:29:06.489Z  INFO 6448 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 2 ms
```



```
package com.bank.transaction_processing;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplicat

@SpringBootApplication
public class TransactionProcessingApplication {

    public static void main(String[] args) { SpringApplication.r
```

```
ain] org.hibernate.Version                    : HHH000412: Hibernate ORM core version 6.5.3.Fir
ain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
ain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class tra
ain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibe
ain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persis
ain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default.
ain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
ain] o.s.b.a.e.web.EndpointLinksResolver      : Exposing 1 endpoint beneath base path '/actuatc
ain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context
ain] c.b.t.TransactionProcessingApplication   : Started TransactionProcessingApplication in 9.9
```

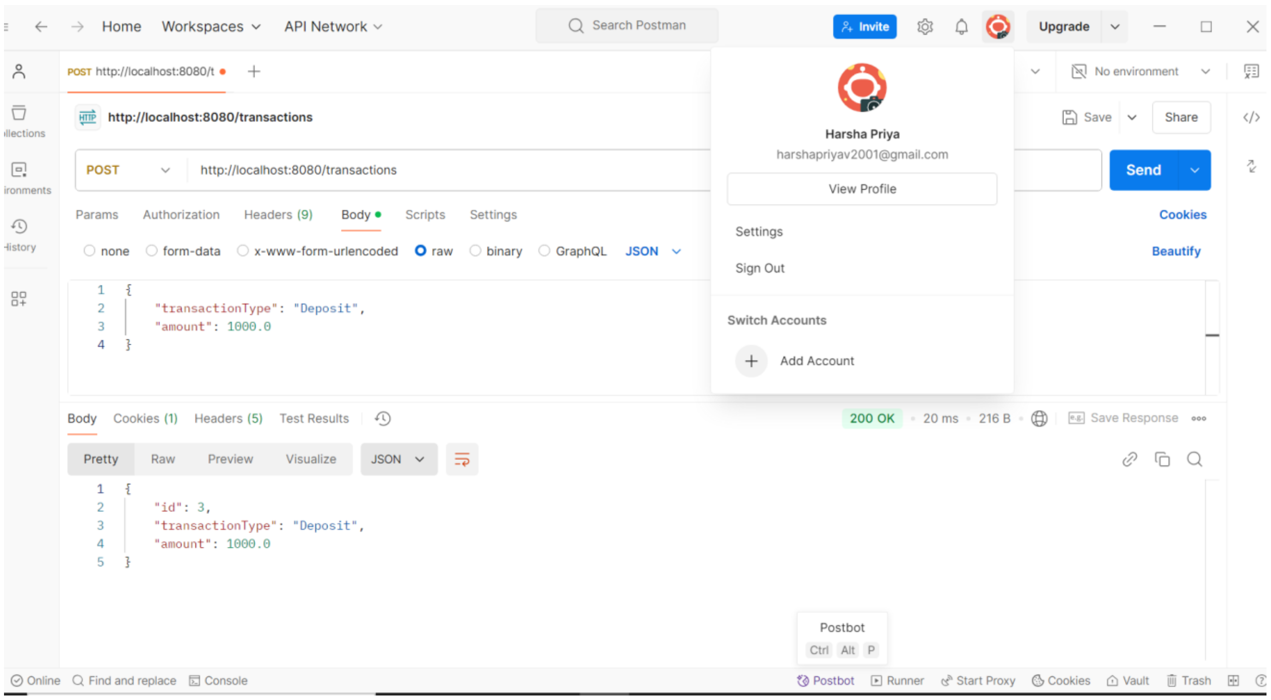

transaction-processing
Version control
TransactionProcessingApplication
Project
TransactionController.java
application.yml
pom.xml (transaction-processing)
Maven
spring-boot:help
spring-boot:run
spring-boot:start
spring-boot:stop
Dependencies
Run
TransactionProcessingApplication
transaction-processing [org.springframework.boot:spring...

```
package com.bank.transaction_processing;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController  no usages
@RequestMapping("/transactions")
public class TransactionController {

    @Autowired  2 usages
    private TransactionRepository transactionRepository;
```

transaction-processing [org.springf 34 sec, 499 ms

```
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  30.561 s
[INFO] Finished at: 2024-11-27T05:51:11Z
[INFO] ------------------------------------------------------------------------

Process finished with exit code 0
```

transaction-processing > transaction-processing > pom.xml   7:33 CRLF UTF-8 4 spaces

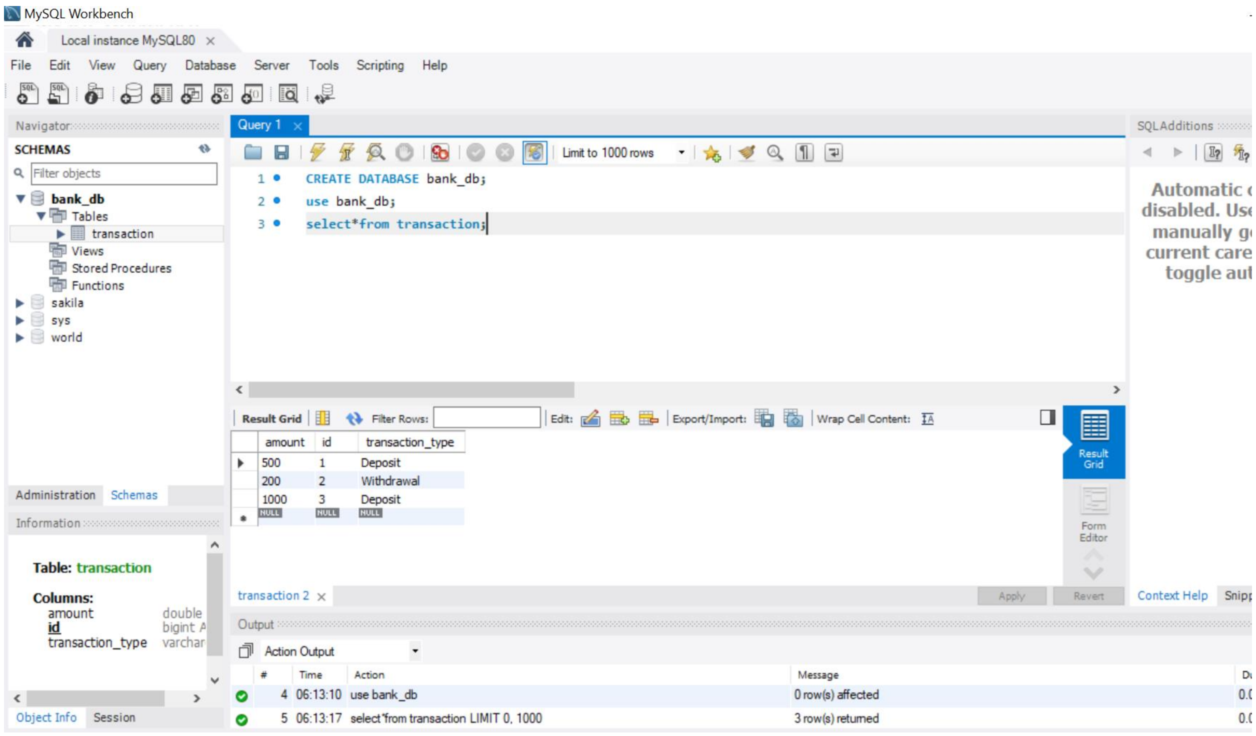


POST http://localhost:8080/transactions

```
{
    "transactionType": "Deposit",
    "amount": 1000.0
}
```

200 OK · 20 ms · 216 B

```
{
    "id": 3,
    "transactionType": "Deposit",
    "amount": 1000.0
}
```



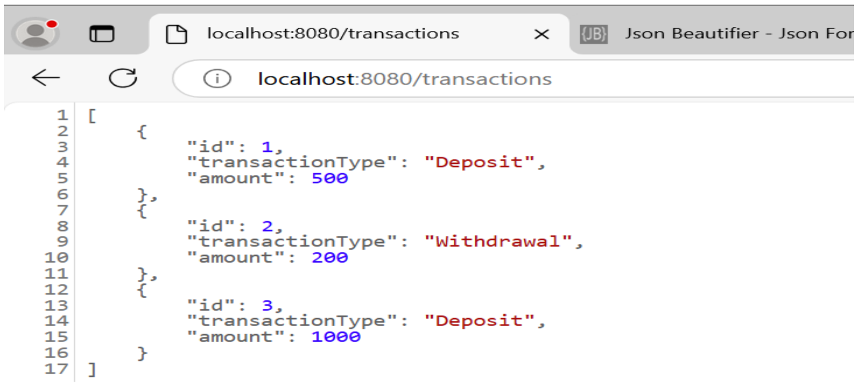
```
CREATE DATABASE bank_db;
use bank_db;
select*from transaction;
```

	amount	id	transaction_type
▶	500	1	Deposit
	200	2	Withdrawal
	1000	3	Deposit
*	NULL	NULL	NULL

Table: transaction

Columns:
amount double
id bigint A
transaction_type varchar

#	Time	Action	Message
4	06:13:10	use bank_db	0 row(s) affected
5	06:13:17	select*from transaction LIMIT 0, 1000	3 row(s) returned



localhost:8080/transactions

```
[
    {
        "id": 1,
        "transactionType": "Deposit",
        "amount": 500
    },
    {
        "id": 2,
        "transactionType": "Withdrawal",
        "amount": 200
    },
    {
        "id": 3,
        "transactionType": "Deposit",
        "amount": 1000
    }
]
```