Bank account withdrawal code improvement exercise:

Improve on the provided code snippet, which outlines a basic banking operation involving an account balance withdrawal and an event notification, which is part of the core domain for the organization. You have full carte blanche to suggest improvements across any and all aspects of the implementation. Your recommendations should aim to enhance the ***code's structure***, ***efficiency***, ***throughput***, ***maintainability***, ***flexibility***, ***consistency***, ***fault tolerance***, ***testability***, ***dependency management***, ***observability***, ***auditability***, ***portability***, ***correctness***, ***cost efficiency***, ***data governance***, ***interoperability, architecture*** and ***overall quality*** and ***more***, while preserving the existing business functionality.

Feel free to use a programming language you feel comfortable with.

Out of scope:

- Security is not part of this exercise.
- The code snippet does not need to compile and run.
- You can omit unit/integration testing for this particular piece of code.

Please submit:

- An outline of your approach, ensuring that the fundamental business capability remains unchanged.
- Elaboration on any implementation choices.
- The fixed code snippet.
- Document any unclear library usage.

Estimated time we'd expect you to spend on this exercise (but we will not time it): ~ 1.5 to 2.5 hours.

```java
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sns.SnsClient;
import software.amazon.awssdk.services.sns.model.PublishRequest;
import software.amazon.awssdk.services.sns.model.PublishResponse;

import java.math.BigDecimal;

@RestController
@RequestMapping("/bank")
public class BankAccountController {

    @Autowired
    private JdbcTemplate jdbcTemplate;

    private SnsClient snsClient;

    public BankAccountController() {
        this.snsClient = SnsClient.builder()
                                  .region(Region.YOUR_REGION) // Specify your region
                                  .build();
```

```
     }

    @PostMapping("/withdraw")
    public String withdraw(@RequestParam("accountId") Long accountId, @RequestParam("amount") BigDecimal amount) {
        // Check current balance
        String sql = "SELECT balance FROM accounts WHERE id = ?";
        BigDecimal currentBalance = jdbcTemplate.queryForObject(sql, new Object[]{accountId}, BigDecimal.class);

        if (currentBalance != null && currentBalance.compareTo(amount) >= 0) {
            // Update balance
            sql = "UPDATE accounts SET balance = balance - ? WHERE id = ?";
            int rowsAffected = jdbcTemplate.update(sql, amount, accountId);
            if (rowsAffected > 0) {
                return "Withdrawal successful";
            } else {
                // In case the update fails for reasons other than a balance check
                return "Withdrawal failed";
            }
        } else {
            // Insufficient funds
            return "Insufficient funds for withdrawal";
        }

        // After a successful withdrawal, publish a withdrawal event to SNS
        WithdrawalEvent event = new WithdrawalEvent(amount, accountId, "SUCCESSFUL");
        String eventJson = event.toJson(); // Convert event to JSON
        String snsTopicArn = "arn:aws:sns:YOUR_REGION:YOUR_ACCOUNT_ID:YOUR_TOPIC_NAME";

        PublishRequest publishRequest = PublishRequest.builder()
                                                       .message(eventJson)
                                                       .topicArn(snsTopicArn)
                                                       .build();
        PublishResponse publishResponse = snsClient.publish(publishRequest);

        return "Withdrawal successful";
    }
}

-------

public class WithdrawalEvent {
    private BigDecimal amount;
    private Long accountId;
    private String status;

    public WithdrawalEvent(BigDecimal amount, Long accountId, String status) {
        this.amount = amount;
        this.accountId = accountId;
        this.status = status;
    }

    public BigDecimal getAmount() {
        return amount;
    }

    public Long getAccountId() {
        return accountId;
    }

    public String getStatus() {
        return status;
    }

    // Convert to JSON String
    public String toJson() {
        return String.format("{\"amount\":\"%s\",\"accountId\":%d,\"status\":\"%s\"}", amount, accountId, status);
    }
}
```