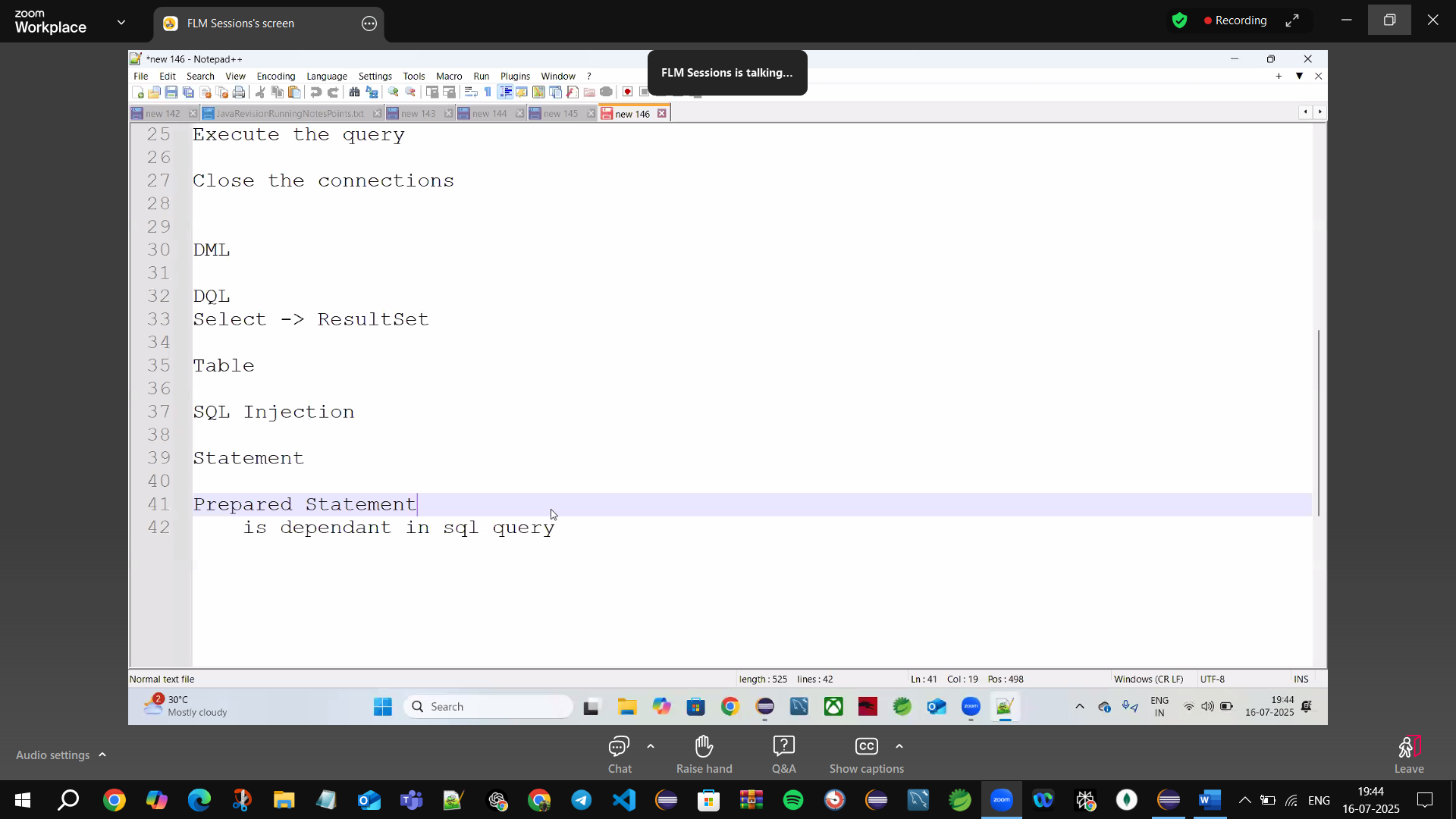
**3.MetaData\_TCL\_Callable**



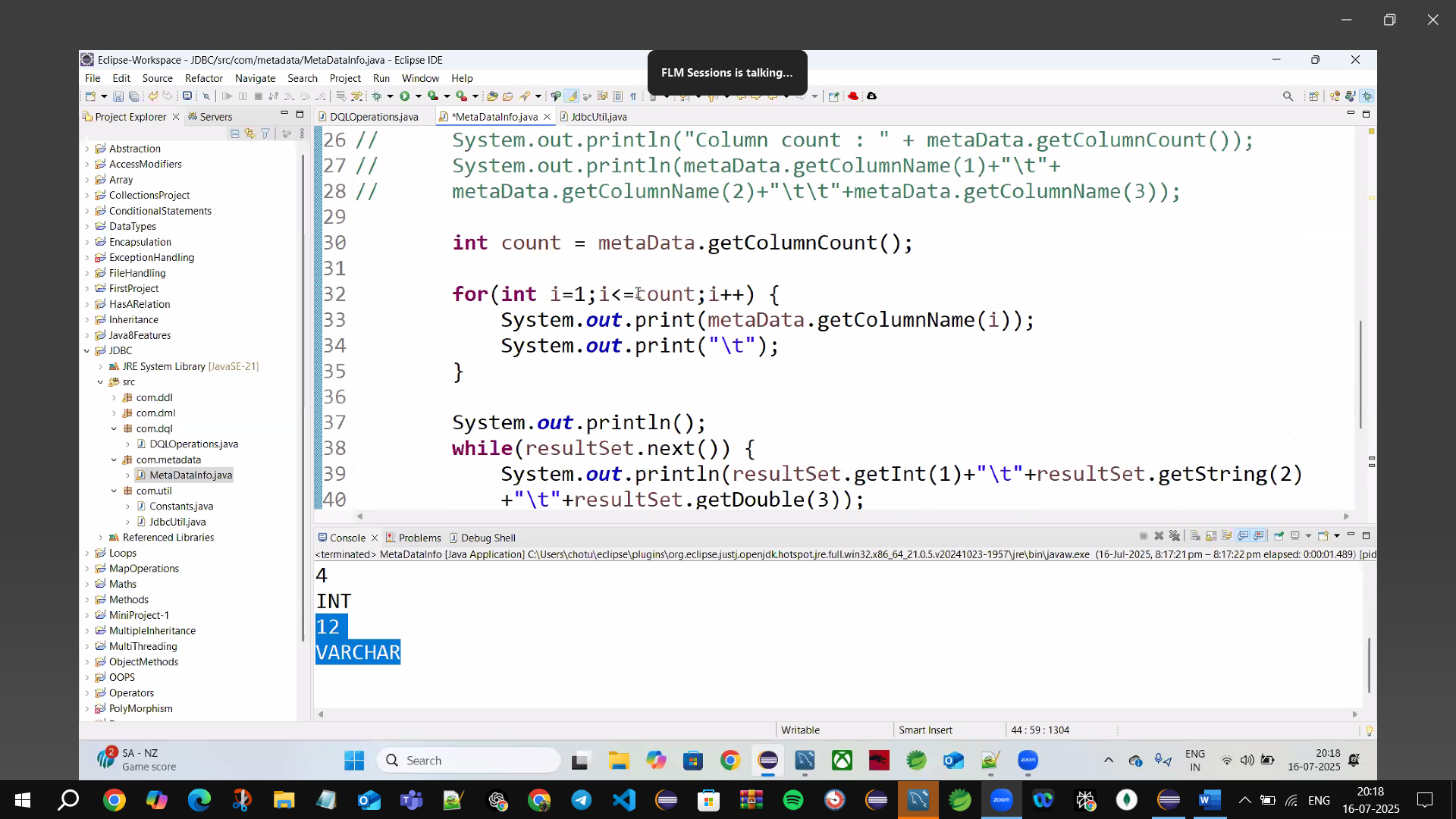
**Spend:** 2+ months (on Core Java)

🡪Bcz of Concatenation SQL injection Happens

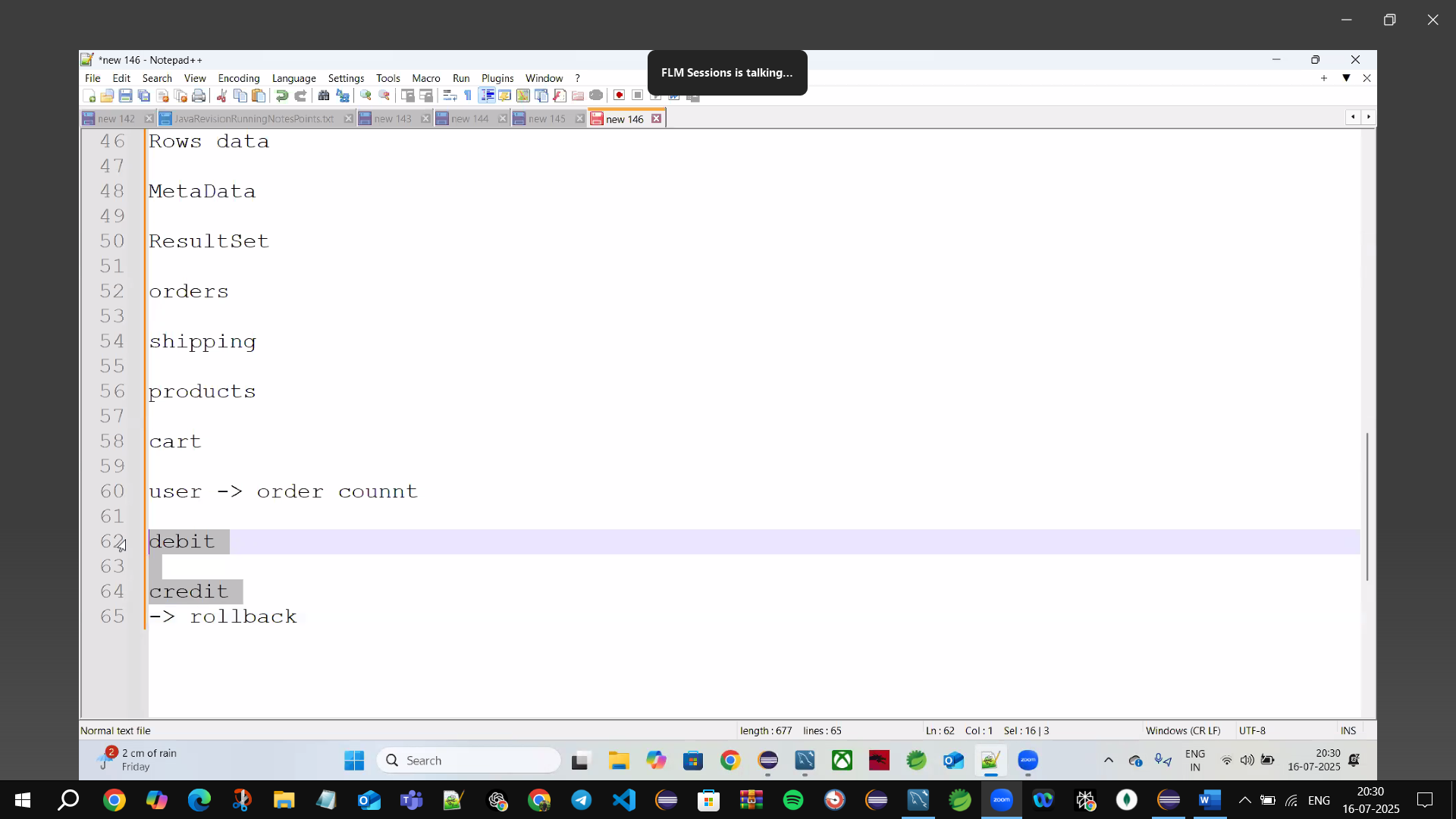
**Q) Why do we use Prepared Statement?**  
ans: Bcz There is a chance of SQL injection

**Q) Why SQL Injection happens?**

ans: Bcz of Concatenation (+‘<space> --<space>)

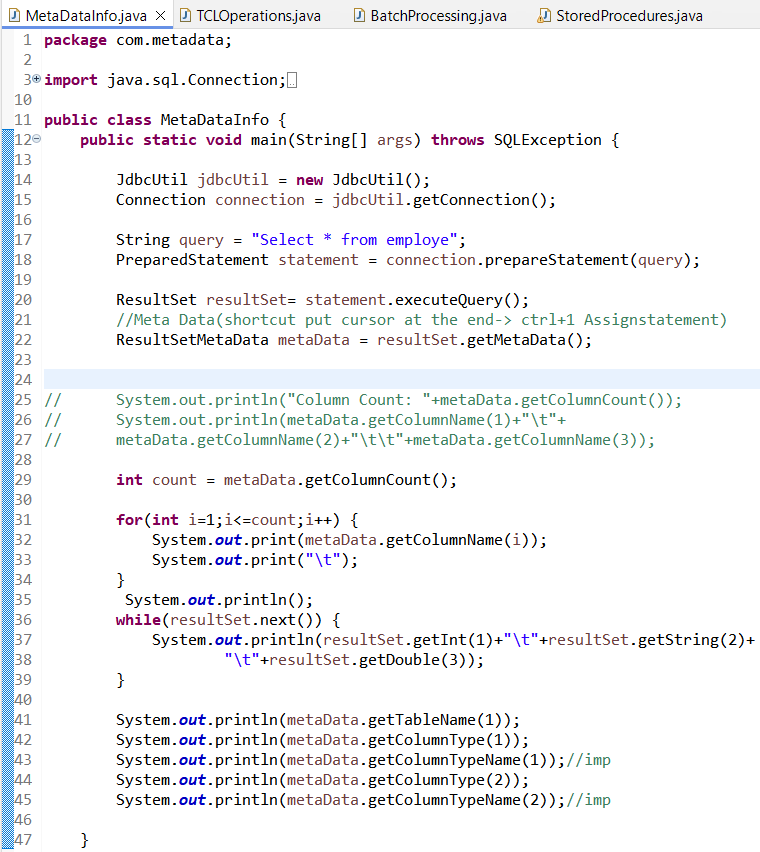
🡪In Prepared Statement we use place holders (?->(1,1))1st->index,1column

shortcut put cursor at the end-> ctrl+1 Assignstatement



**Metadata🡪** Data about Data(Table information)





**TCL Operations:** Transaction control Language

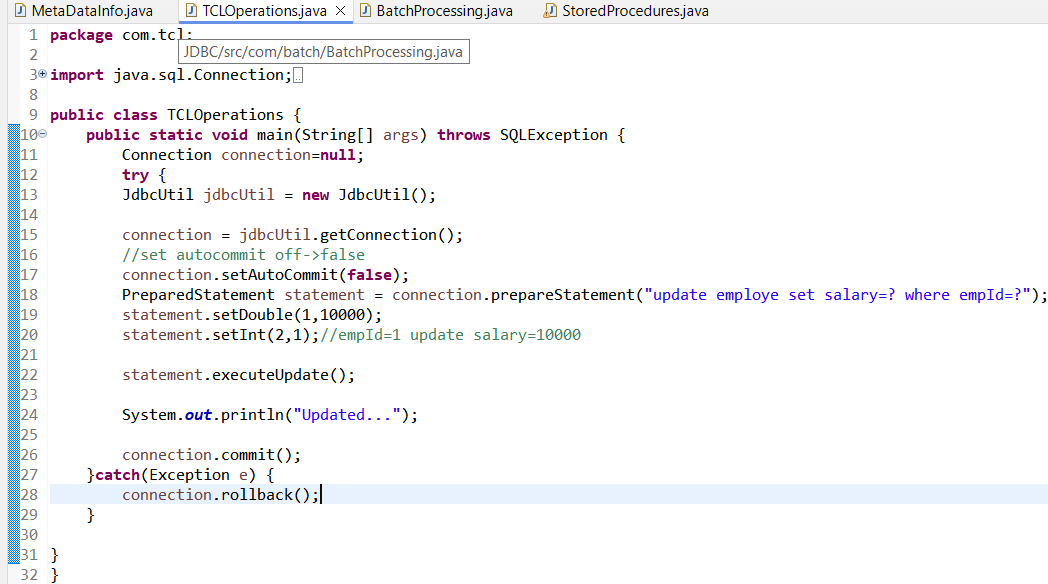
🡪setAutoCommit(false)

Commit()

Rollback()

🡪if autocommit(true) can’t rollback()



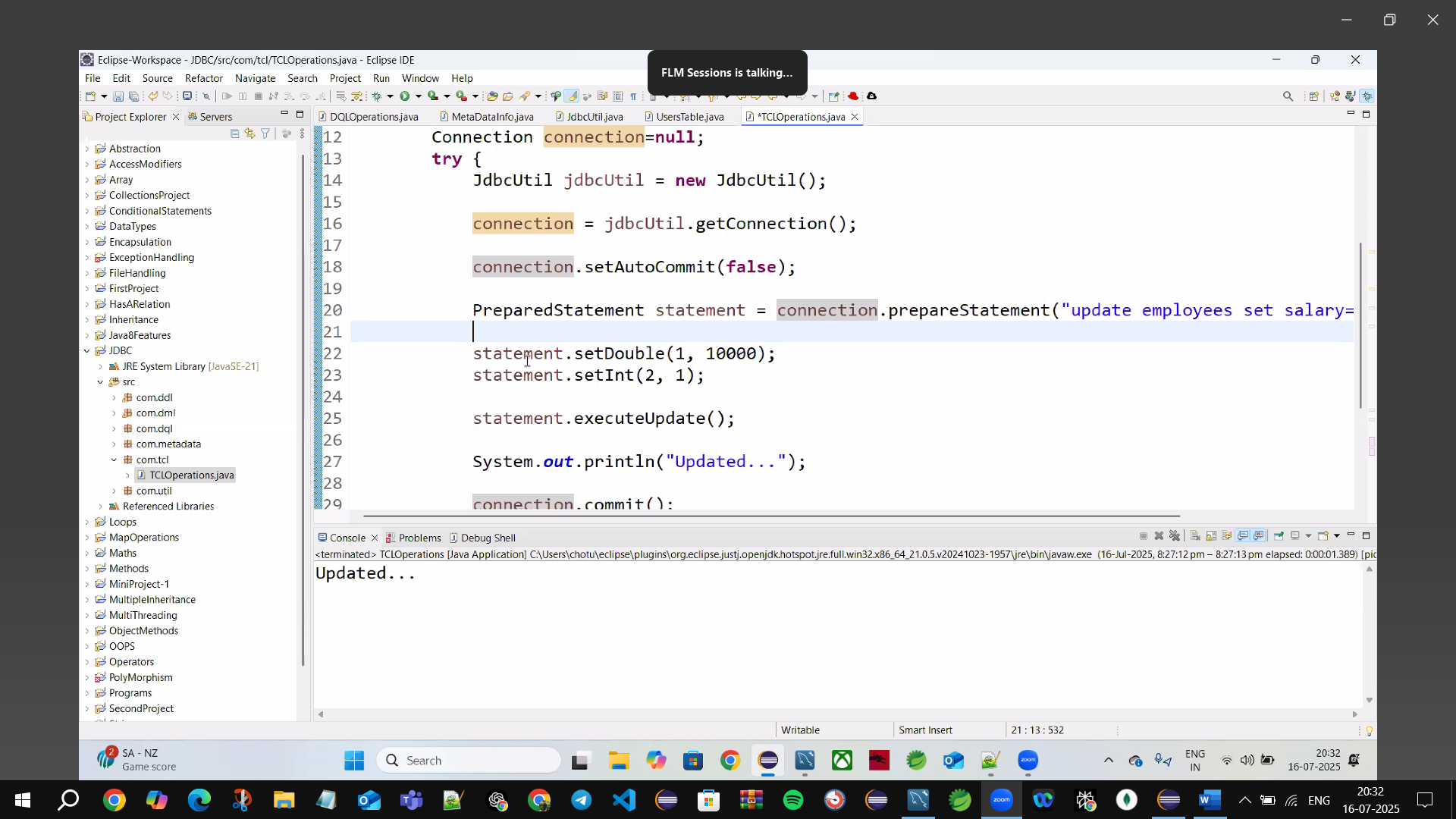


Debit()

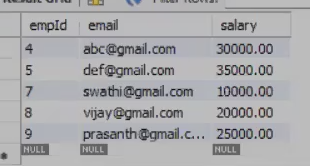
Roll back()

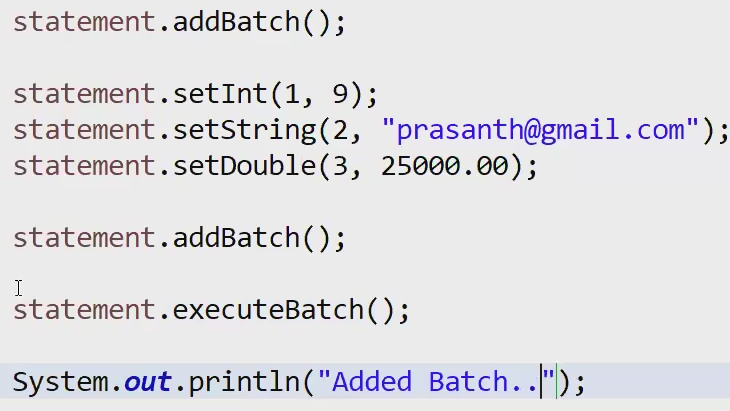
Credit()

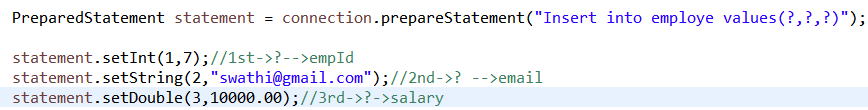
🡪if money debited in account & not credited in account they rollback the transaction then the debited money will be rollbacked & Refund the amount back



**Batch Processing🡪**Insert multiple Rows of data @ a time into table use Batching



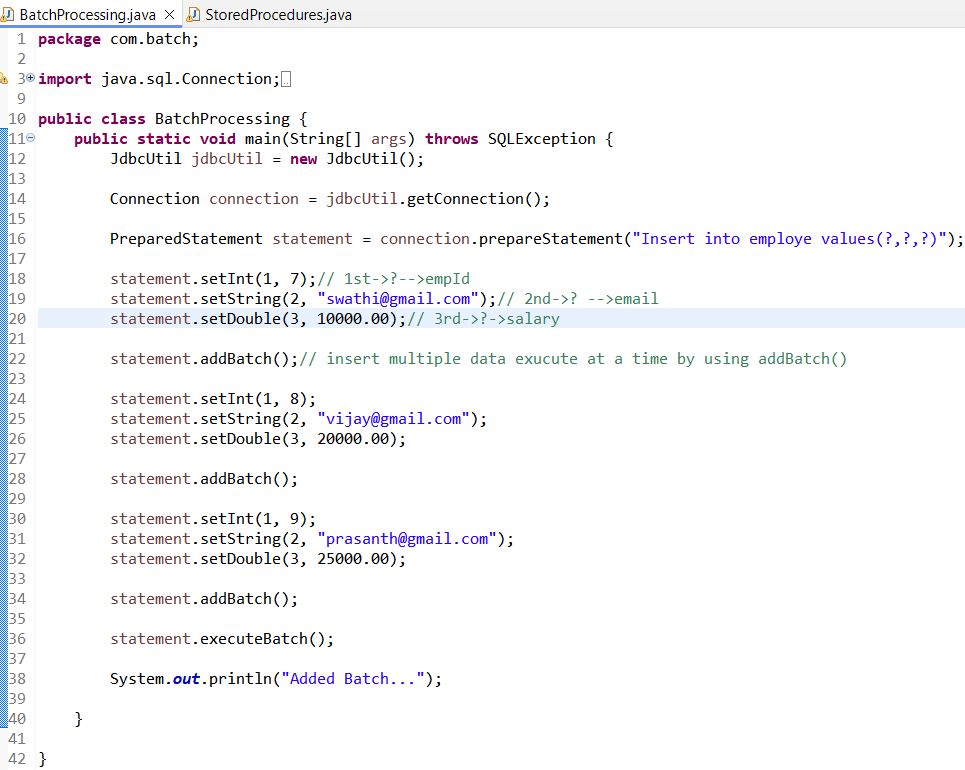




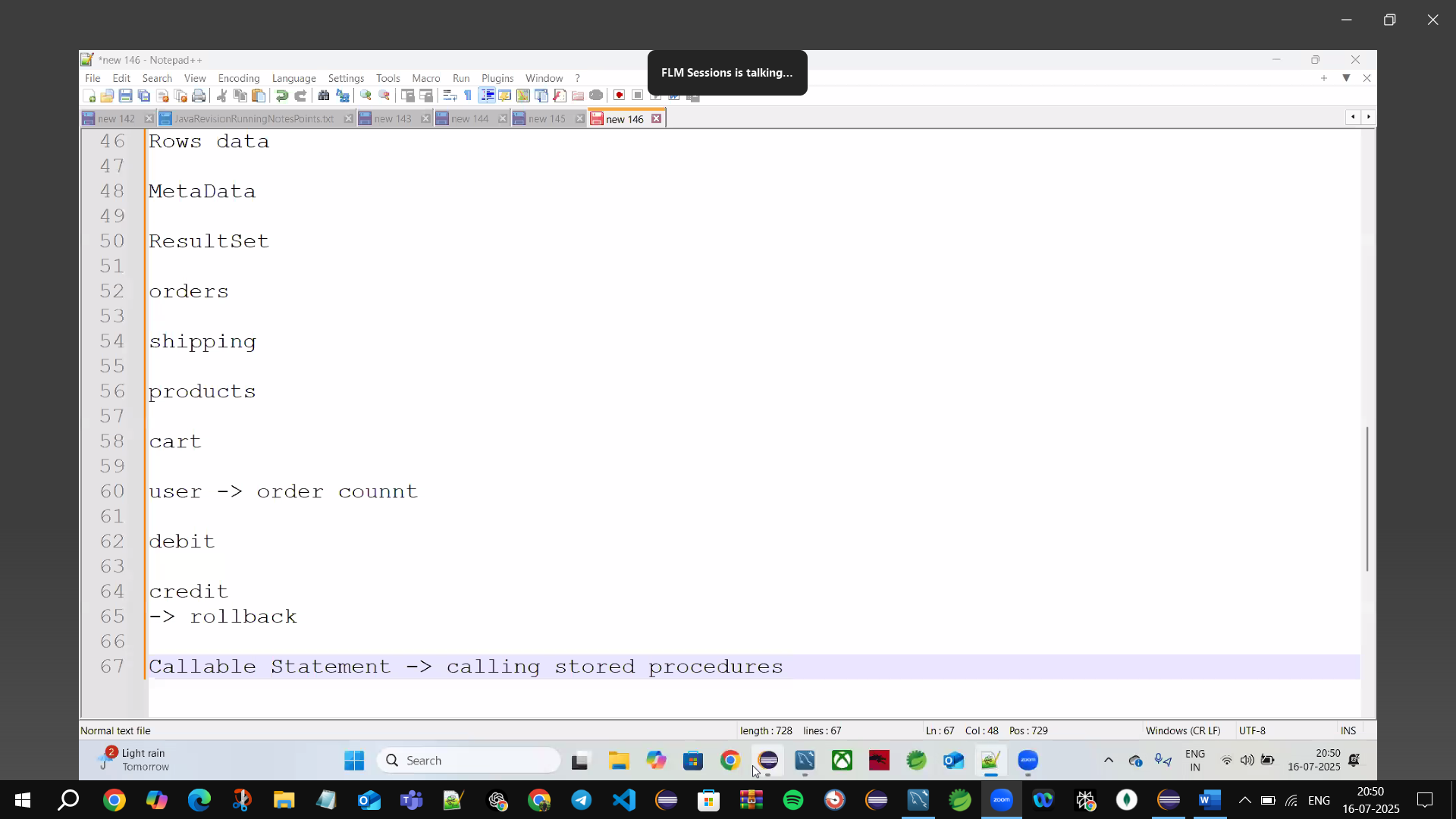
Details (emp data)

addBatch()

executeBatch();





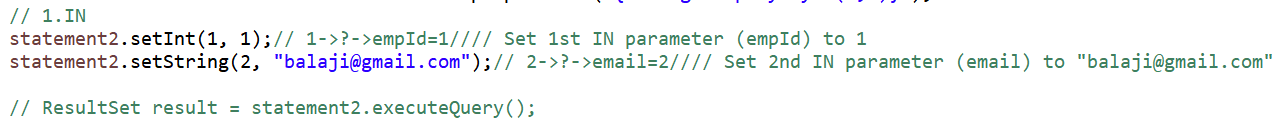


**Callable Statement🡪**calling Stored Procedures

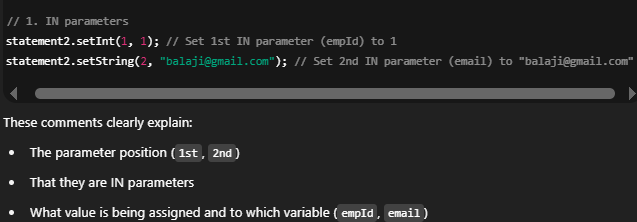
{}🡪preparedStatement in java

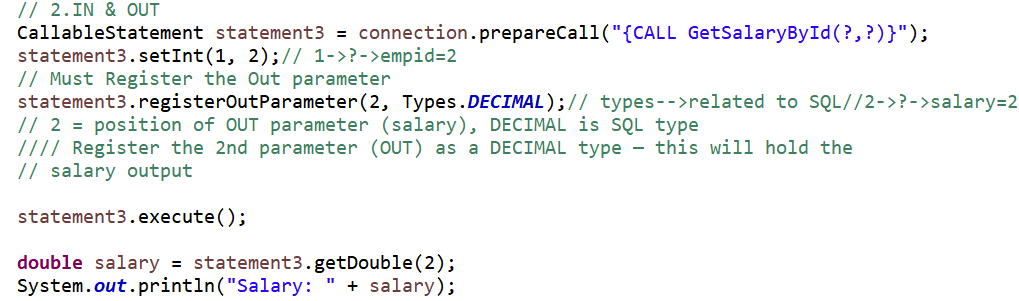
“{CALL getAllEmploye()}”

Stored Procedures🡪types🡪IN, OUT, INOUT



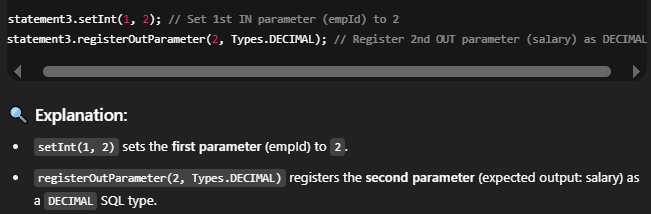
**✅ IN**

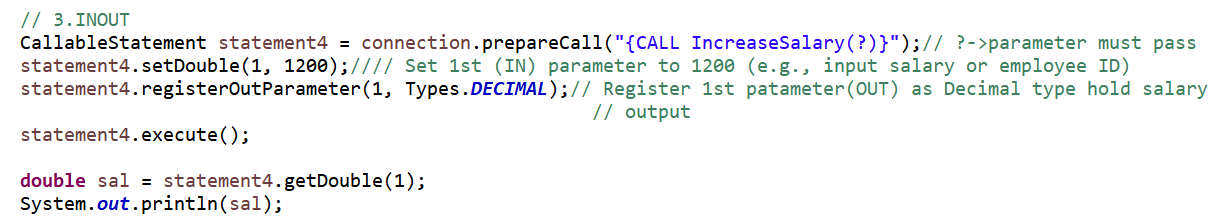
**1st parameter (?)** → IN: empId = 1 (value)  
**2nd parameter (?)** → IN: email = 'balaji@gmail.com' (value)



**✅ IN & OUT**

**1st parameter (?) → IN: empId = 2 (value)  
2nd parameter (?) → OUT: salary as DECIMAL**





**✅ INOUT Parameter Explanation (in your format):**

* **1st parameter (?)** → setDouble(1, 1200) → sets IN value: salary = 1200
* **1st parameter (?)** → registerOutParameter(1, Types.DECIMAL) → registers OUT type: DECIMAL
* **1st parameter (?)** → getDouble(1) → retrieves modified salary value from OUT





**✅ Why must we register the OUT parameter in JDBC?**

You **must register an OUT (or INOUT) parameter** in JDBC because:

**🔍 JDBC needs to know:**

1. **Which parameter** will return a value (OUT or INOUT), and
2. **What type** of value to expect (e.g., INTEGER, DECIMAL, VARCHAR)

**💡 Reason:**

When you call a stored procedure that includes OUT or INOUT parameters, **JDBC doesn't automatically know** which ? placeholders will be returning data.

So, calling:



is like telling JDBC:

"Hey, when this procedure finishes, expect the **2nd parameter** to return a **DECIMAL** value."

Without this line, JDBC won't:

* Capture the returned value
* Allocate memory to store it
* Allow you to retrieve it with getXXX(index)

**🚫 What happens if you don’t register it?**

* You’ll likely get an exception when you try to retrieve the OUT value (e.g., getDouble(2)), such as:



**✅ Summary:**

| **Step** | **Purpose** |
| --- | --- |
| registerOutParameter(...) | Tell JDBC which parameter will return a value, and what type it is |
| execute() | Run the stored procedure |
| getXXX(index) | Read the returned value |



