```
File Edit Selection View Go Run Terminal Help
                                               Settings - project - code-server
                                                                                               OUTPUT TERMINAL
                   DEBUG CONSOLE
  PROBLEMS
              Cl1
                                                    1200.00
                                                              2020-09-10
                           P4
               Cl3
                                                              2020-09-10
                           P1
                                                     150.00
               Cl1
                                                              2020-09-12
                                                     500.00
                           P1
              C12
                                                              2020-09-13
         10
                           P2
                                                     100.00
  10 rows in set (0.00 sec)
  mysql> DELIMITER //
  mysql> CREATE FUNCTION GetTotalCost(Cost DECIMAL(5,2)) RETURNS DECIMAL(5,2) DETERMINISTIC BEGIN IF
  (Cost >= 100 AND Cost < 500) THEN SET Cost = Cost - (Cost * 0.1); ELSEIF (Cost >= 500) THEN SET Cos
  t = Cost - (Cost * 0.2); END IF; RETURN (Cost); END//
  Query OK, 0 rows affected (0.01 sec)
  mysql> DELIMITER ;
  mysql> SELECT GetTotalCost(500);
    GetTotalCost(500)
                400.00
  1 row in set (0.00 sec)
  mysql> DROP FUNCTION GetTotalCost;
```

Question

When creating complex stored procedures, you must change the delimiter from a semi-colon to another delimiter sign so that MySQL can compile your code in a BEGIN-END block as one compound statement.

- True
- False

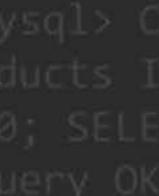
/

Correct

Correct! MySQL requires you to change the delimiter sign so that it can compile your code.

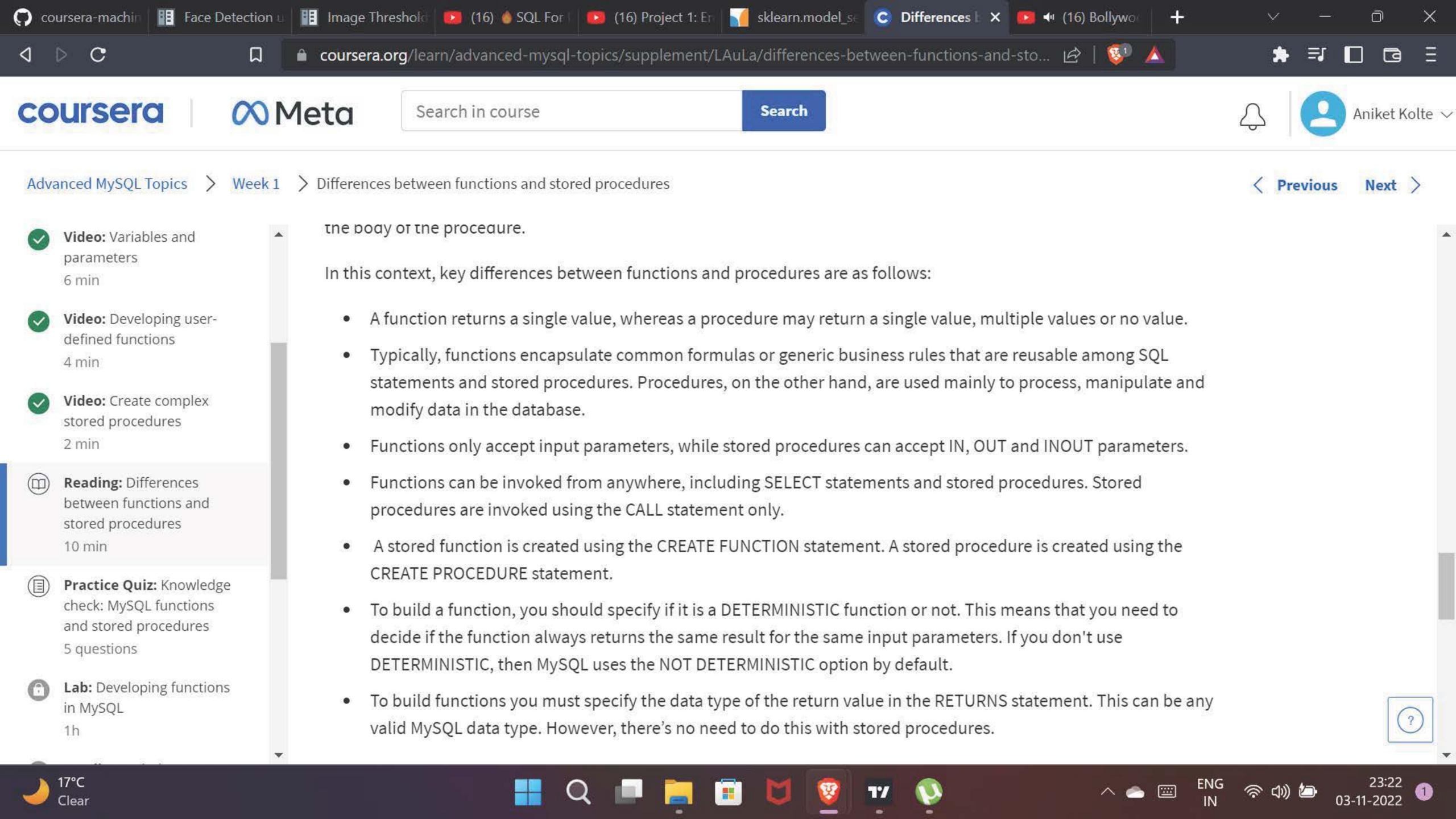
Skip

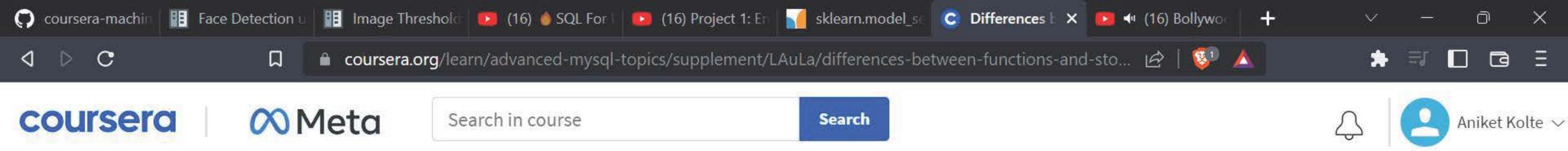
Continue



nysql> D

2:17 / 2:42







	Video: Variables and
	parameters
	6 min

- Video: Developing userdefined functions 4 min
- Video: Create complex stored procedures
 2 min
- Reading: Differences
 between functions and
 stored procedures
 10 min
- Practice Quiz: Knowledge check: MySQL functions and stored procedures 5 questions
- Lab: Developing functions in MySQL

 1h

The following table provides a summary of the key differences between stored procedures and stored functions.

	Functions	Procedures
1	Created using CREATE FUNCTION command	Created using the CREATE PROCEDURE command
2	Invoked using the SELECT statement	Invoked using the CALL statement
3	Must return a single value	Outputs values via the OUT parameter
4	Takes IN parameters only	Takes IN, OUT and INOUT parameters
5	Typically encapsulates common formulas or generic business rules	Typically used to process, manipulate and modify data in the database
6	Must specify the data type of the return value	User must specify the OUT paameter type

Conclusion

Functions and procedures are used to encapsulate code that can be executed to implement repetitive tasks such as equations, formulas or business rules.





















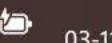






< Previous

Next >







Lucky Shrub and MySQL trigger types

```
CREATE TRIGGER AfterDeleteOrder
AFTER DELETE
ON Orders FOR EACH ROW
INSERT INTO Audits
VALUES('AFTER', CONCAT('Order',OLD.OrderID, 'was deleted at', CURRENT_TIME(),
'on', CURRENT_DATE()), 'DELETE');
```





2:33 / 3:12

□ ☆ ★

Lucky Shrub and one-time scheduled events syntax

```
CREATE EVENT GenerateRevenueReport

ON SCHEDULE AT CURRENT_TIMESTAMP + INTERVAL 12 HOUR

DO

BEGIN

INSERT INTO ReportData (OrderID, ClientID, ProductID, Quantity, Cost, Date)

SELECT *

FROM Orders

WHERE Date

BETWEEN '2022-08-01' AND '2022-08-31';

END
```





Lucky Shrub and recurring scheduled event syntax

```
CREATE EVENT DailyRestock
ON SCHEDULE
EVERY 1 DAY
DO
BEGIN
  IF Products.NumberOfItems < 50 THEN</pre>
  UPDATE Products SET NumberOfItems = 50;
  END IF;
END
```





```
File Edit Selection View Go Run Terminal Help
                               Settings - project - code-server
                                                             Using where
  1 row in set, 1 warning (0.00 sec)
出
  mysql> CREATE INDEX IdxFullName ON Clients(FullName);
  Query OK, 0 rows affected (0.06 sec)
  Records: 0 Duplicates: 0 Warnings: 0
  mysql> EXPLAIN SELECT ContactNumber FROM Clients WHERE FullName='Jane Delgado';
  ------
   id | select_type | table | partitions | type | possible_keys | key
                                                    | key_len | ref | rows |
  filtered Extra
  ------
                                          | IdxFullName | 803
   1 | SIMPLE | Clients | NULL | ref | IdxFullName
                                                         const 1
   100.00 | NULL
  1 row in set, 1 warning (0.00 sec)
  mysal>
    4:22 / 4:36

□ ☆ ★
```

Lucky Shrub and MySQL transactions

```
START TRANSACTION
INSERT INTO Orders (OrderID, ClientID, ProductID , Quantity, Cost, Date)
VALUES (22, "Cl1", "P1", 10, 500, "2022-09-01" );
UPDATE Products SET NumberOfItems = (NumberOfItems - 10) WHERE ProductID =
"P1";
SELECT Orders.OrderID, Orders.Quantity, Products.ProductID,
Products.NumberOfItems FROM Orders INNER JOIN Products ON (Orders.ProductID =
Products.ProductID) WHERE Orders.OrderID = 22;
```

Lucky Shrub and Common Table Expressions

```
WITH
Average_Sales_2020 AS (SELECT CONCAT(AVG(Cost), "in 2020") AS "Average Sale" FROM
Orders WHERE YEAR(Date) = 2020),
Average_Sales_2021 AS (SELECT CONCAT(AVG(Cost), "in 2021") FROM Orders WHERE
YEAR(Date) = 2021),
Average_Sales_2022 AS (SELECT CONCAT(AVG(Cost), "in 2022") FROM Orders WHERE
YEAR(Date) = 2022)
SELECT * FROM Average_Sales_2020
UNION
SELECT * FROM Average_Sales_2021
UNION
SELECT * FROM Average_Sales_2022;
```



FULL OUTER JOIN

A JOIN used to return all records from two tables, including those that don't have a match.

FULL OUTER JOIN with UNION ALL operator

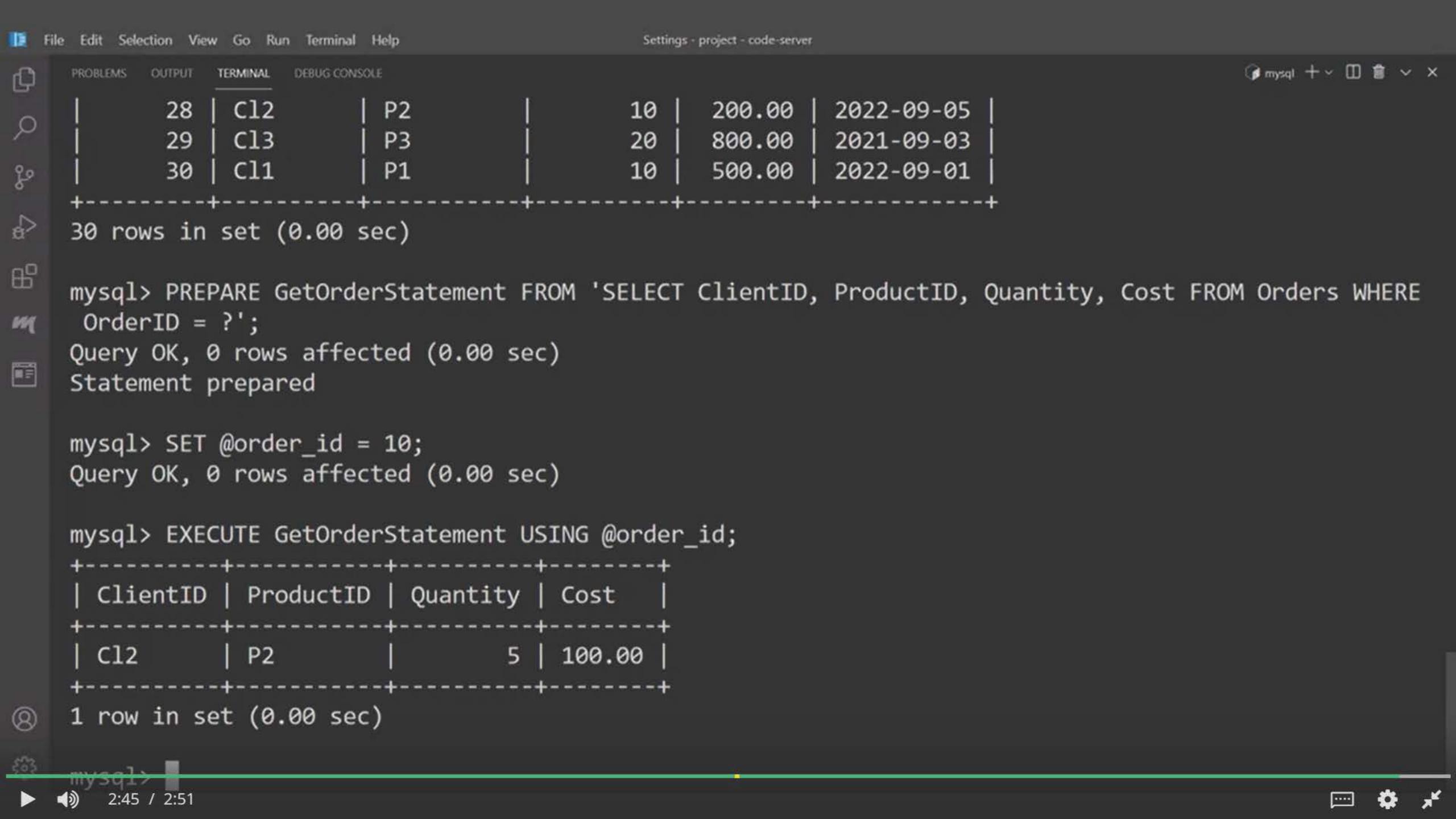
```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.CommonColumn = table2.CommonColumn
UNION ALL
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.CommonColumn = table CommonColumn.CommonColumn
```

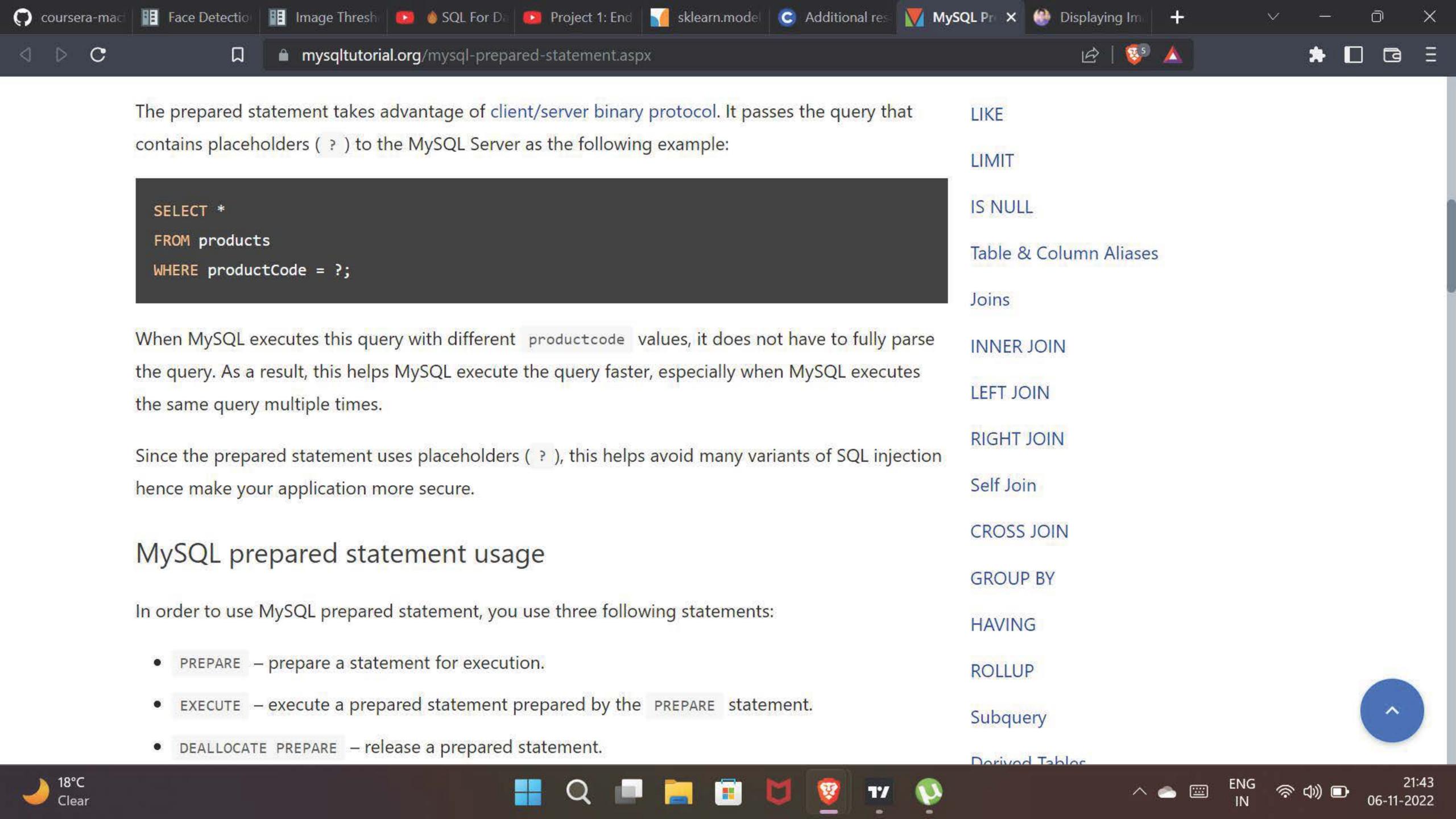




FULL OUTER JOIN with UNION operator

```
SELECT columns
FROM table1
LEFT JOIN table2
ON table1.CommonColumn = table2.CommonColumn
UNION
SELECT columns
FROM table1
RIGHT JOIN table2
ON table1.CommonColumn = table CommonColumn.CommonColumn
```





```
CREATE TRIGGER before employee update
   BEFORE UPDATE ON employees
   FOR EACH ROW
 INSERT INTO employees audit
SET action = 'update',
     employeeNumber = OLD.employeeNumber,
     lastname = OLD.lastname,
     changedat = NOW();
```