### SQL Assignment

-- 1

CREATE database Student;

USE Student;

-- 2

CREATE TABLE StudentBasicInformation(

StudentName VARCHAR(30),

StudentSurname VARCHAR(30),

StudentRollNo INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

StudentAddress VARCHAR(255),

StudentMobile VARCHAR(10),

StudentEmail VARCHAR(30),

StudentAge INT

);

CREATE TABLE StudentAdmissionPaymentDetails(

PaymentId INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

StudentRollNo INT UNSIGNED,

AmountPaid INT,

AmountBalance INT,

PaymentDate DATE,

DueDate DATE,

AmountFine INT

);

CREATE TABLE StudentSubjectInformation(

SubjectOpted VARCHAR(30),

StudentRollNo INT UNSIGNED,

SubjectTotalMarks INT,

StudentObtainedMarks INT,

StudentMarksPercentage float,

PRIMARY KEY(StudentRollNo, SubjectOpted)

);

CREATE TABLE SubjectScholarshipInformation(

ScholarshipApplicationNo INT UNSIGNED AUTO\_INCREMENT PRIMARY KEY,

StudentRollNo INT UNSIGNED,

ScholarshipName VARCHAR(30),

ScholarshipDescription VARCHAR(255),

ScholarshipAmount INT,

ScholarshipCategory VARCHAR(30),

ScholarshipApproval BOOLEAN

);

-- 3

INSERT INTO StudentBasicInformation (StudentName, StudentSurname, StudentAddress, StudentMobile, StudentEmail, StudentAge) values

('name1','surname1','address1','mobile1','email1',20),

('name2','surname2','address2','mobile2','email2',20),

('name3','surname3','address3','mobile3','email3',20),

('name4','surname4','address4','mobile4','email4',20),

('name5','surname5','address5','mobile5','email5',20),

('name6','surname6','address6','mobile6','email6',20),

('name7','surname7','address7','mobile7','email7',20),

('name8','surname8','address8','mobile8','email8',20),

('name9','surname9','address9','mobile9','email8',20),

('name10','surname10','address10','mobile10','email10',20);

INSERT INTO StudentAdmissionPaymentDetails (StudentRollNo, AmountPaid, AmountBalance, PaymentDate, DueDate, AmountFine) values

(1,10000,10000,STR\_TO\_DATE('01-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(2,10000,10000,STR\_TO\_DATE('15-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(3,10000,10000,STR\_TO\_DATE('01-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(4,10000,10000,STR\_TO\_DATE('15-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(5,10000,10000,STR\_TO\_DATE('01-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(6,10000,10000,STR\_TO\_DATE('15-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(7,10000,10000,STR\_TO\_DATE('01-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(8,10000,10000,STR\_TO\_DATE('15-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(9,10000,10000,STR\_TO\_DATE('01-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0),

(10,10000,10000,STR\_TO\_DATE('15-02-2020', '%d-%m-%Y'),STR\_TO\_DATE('14-02-2020', '%d-%m-%Y'),0);

INSERT INTO StudentSubjectInformation (SubjectOpted, StudentRollNo, SubjectTotalMarks, StudentObtainedMarks) values

('subject1',1,100,74),

('subject2',1,80,75),

('subject3',1,90,60),

('subject1',2,100,94),

('subject2',2,80,67),

('subject3',2,90,84),

('subject1',3,100,77),

('subject2',3,80,78),

('subject3',3,90,79),

('subject1',4,100,77),

('subject2',4,80,78),

('subject3',4,90,79),

('subject1',5,100,74),

('subject2',5,80,78),

('subject3',5,90,79),

('subject1',6,100,92),

('subject2',6,80,67),

('subject3',6,90,82),

('subject1',7,100,74),

('subject2',7,80,75),

('subject3',7,90,60),

('subject1',8,100,74),

('subject2',8,80,78),

('subject3',8,90,79),

('subject1',9,100,94),

('subject2',9,80,79),

('subject3',9,90,84),

('subject1',10,100,74),

('subject2',10,80,75),

('subject3',10,90,60);

INSERT INTO SubjectScholarshipInformation (StudentRollNo, ScholarshipName, ScholarshipDescription, ScholarshipAmount, ScholarshipApproval) values

(1,'ssname','ssdescription',3000,false),

(2,'ssname','ssdescription',8000,true),

(3,'ssname','ssdescription',5000,false),

(4,'ssname','ssdescription',6000,true),

(5,'ssname','ssdescription',7000,false),

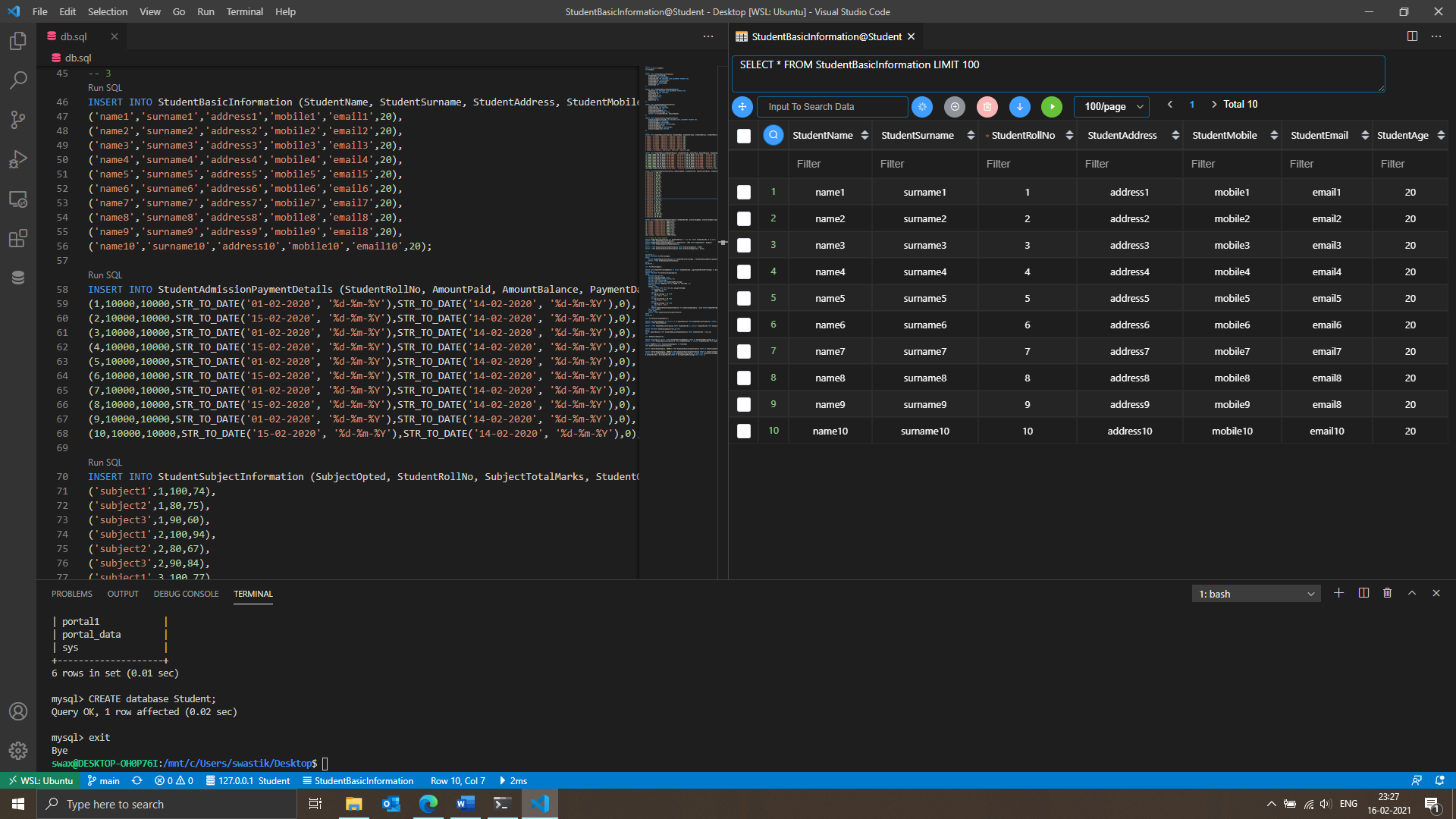
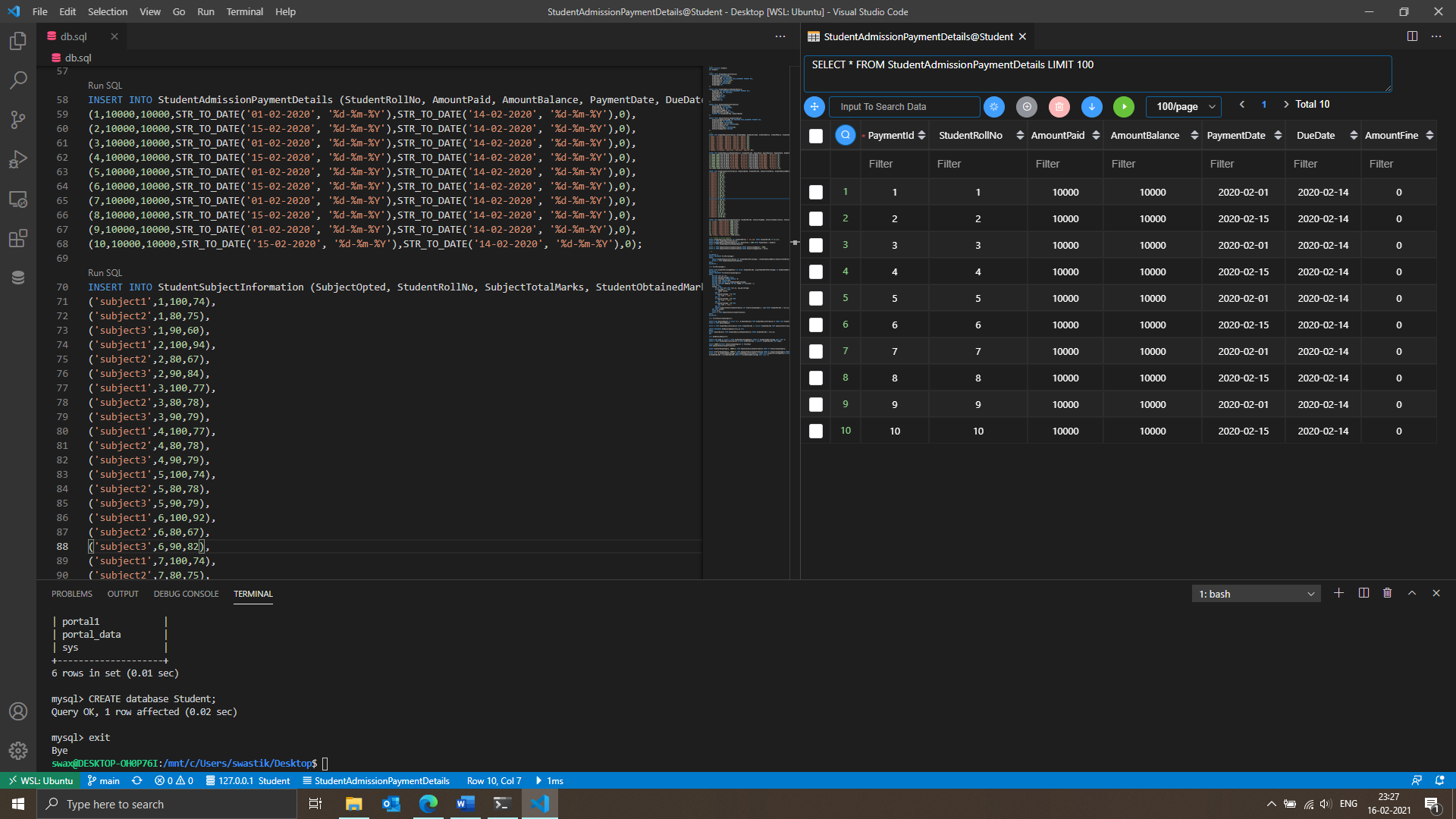
(6,'ssname','ssdescription',8000,true),

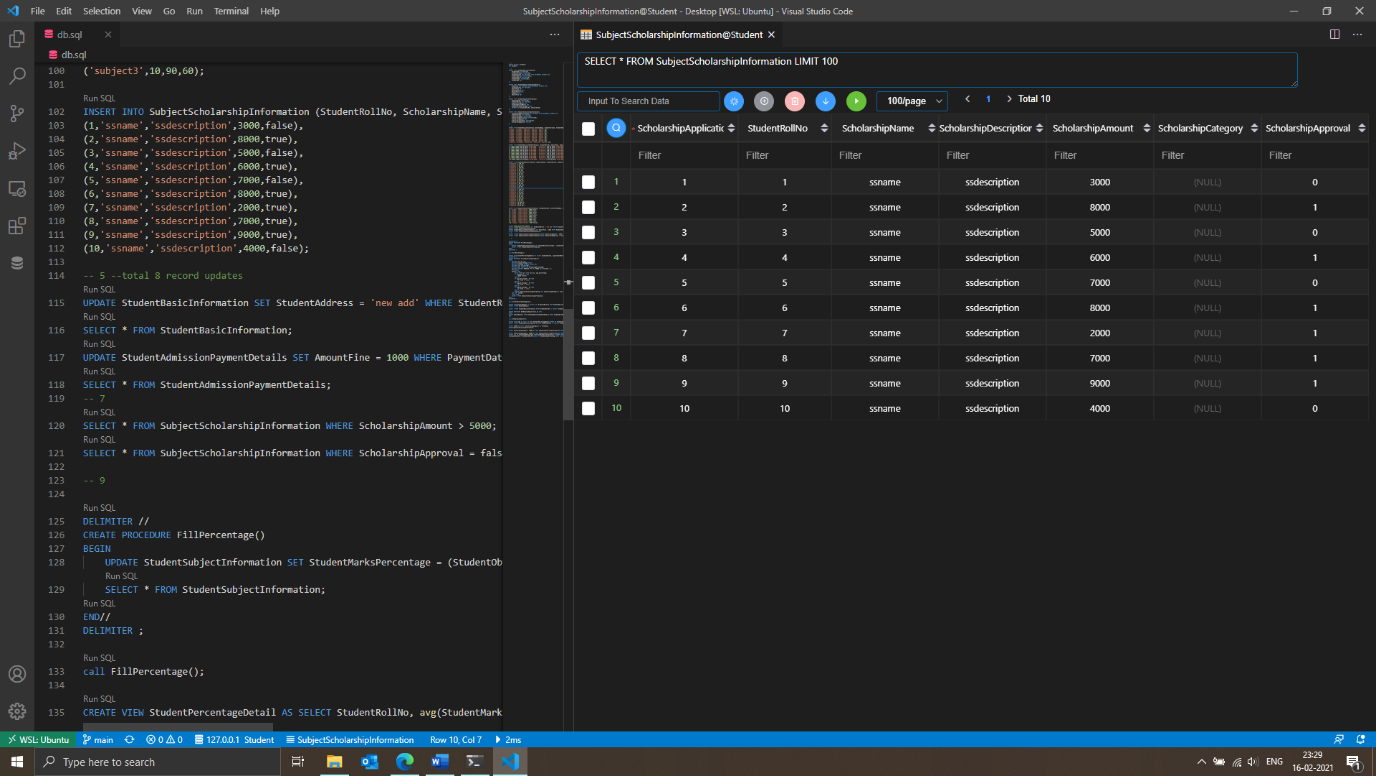
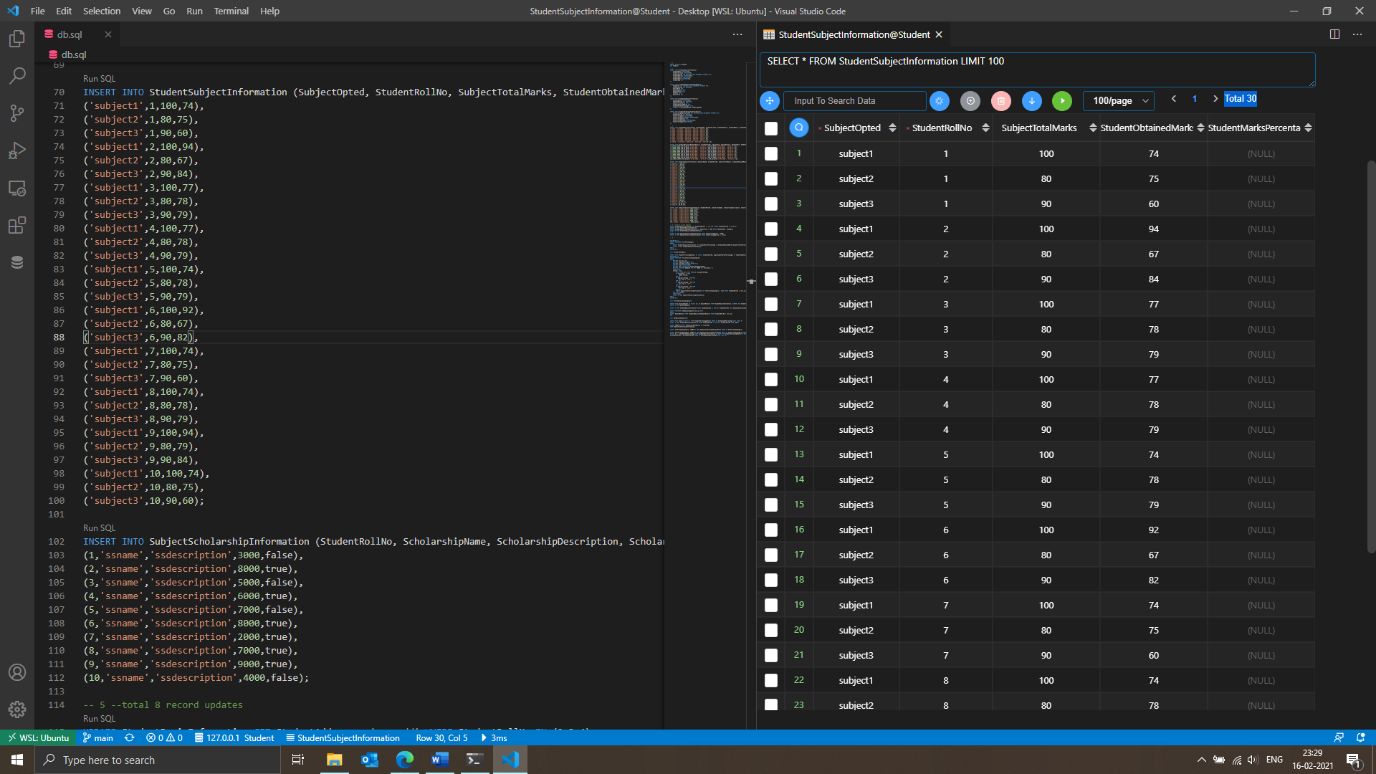
(7,'ssname','ssdescription',2000,true),

(8,'ssname','ssdescription',7000,true),

(9,'ssname','ssdescription',9000,true),

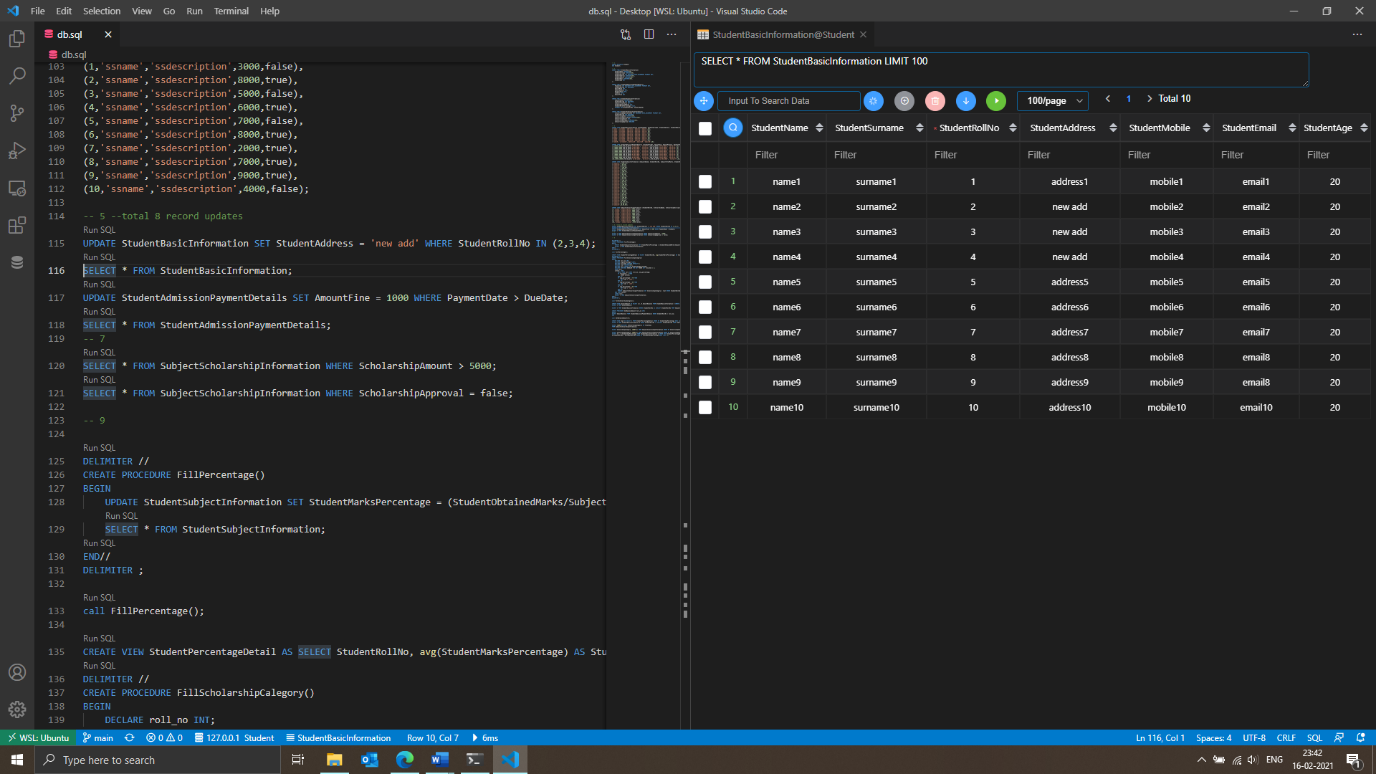
(10,'ssname','ssdescription',4000,false);

-- 4

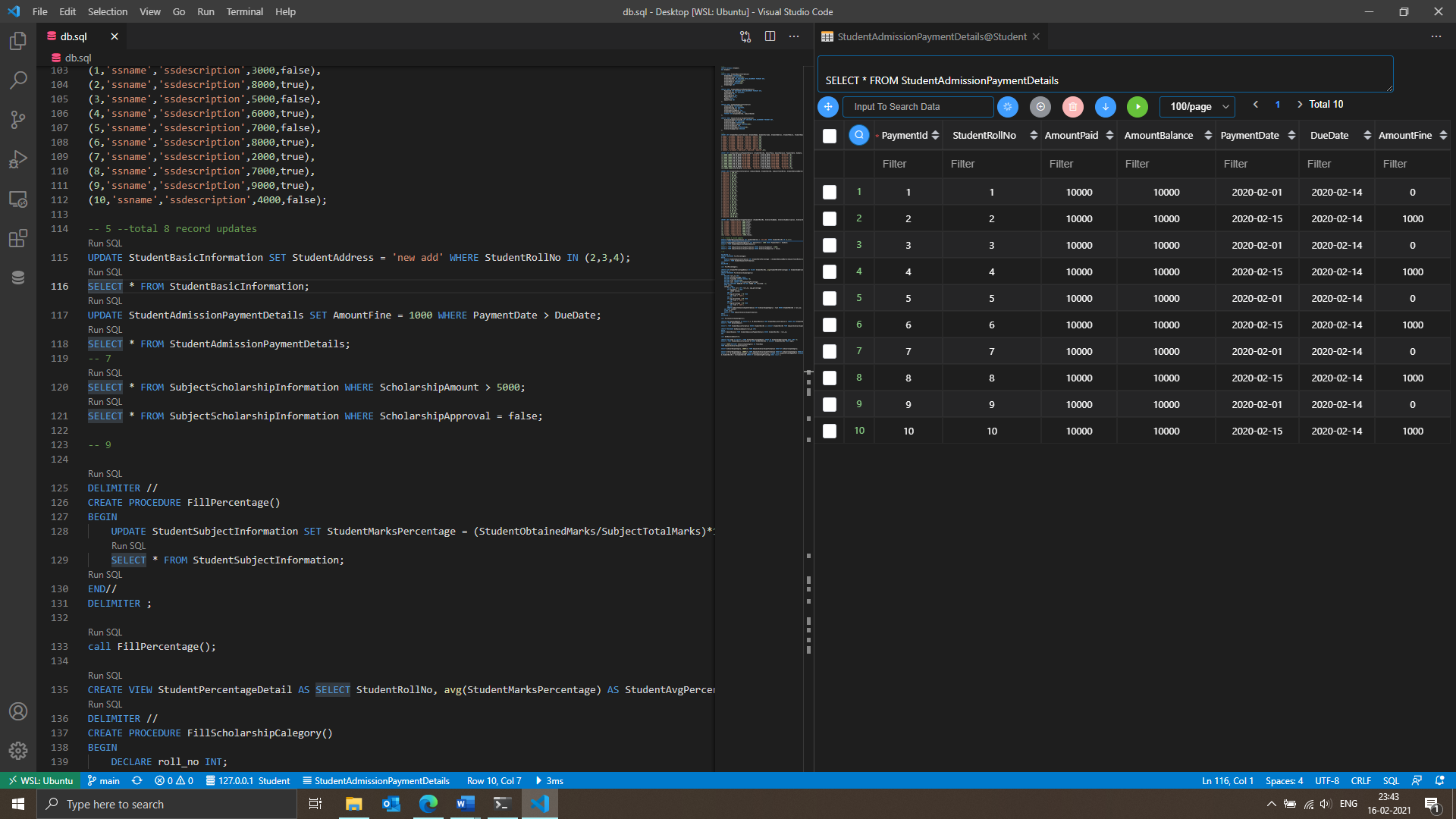


-- 5 --total 8 record updates

UPDATE StudentBasicInformation SET StudentAddress = 'new add' WHERE StudentRollNo IN (2,3,4);

SELECT \* FROM StudentBasicInformation;

UPDATE StudentAdmissionPaymentDetails SET AmountFine = 1000 WHERE PaymentDate > DueDate;

SELECT \* FROM StudentAdmissionPaymentDetails;

-- 7

SELECT \* FROM SubjectScholarshipInformation WHERE ScholarshipAmount > 5000;

-- 8

SELECT \* FROM SubjectScholarshipInformation WHERE ScholarshipApproval = false;

-- 9

DELIMITER //

CREATE PROCEDURE FillPercentage()

BEGIN

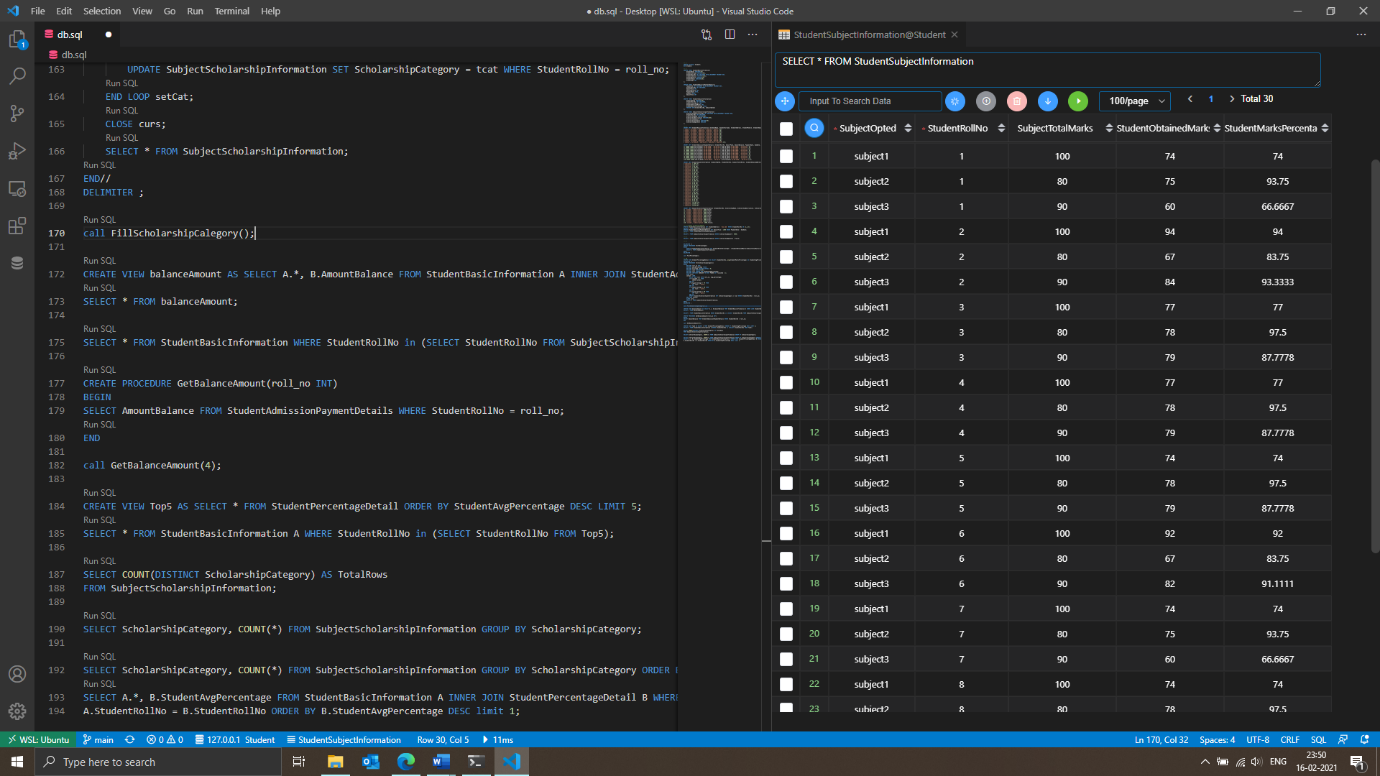
UPDATE StudentSubjectInformation SET StudentMarksPercentage = (StudentObtainedMarks/SubjectTotalMarks)\*100;

SELECT \* FROM StudentSubjectInformation;

END//

DELIMITER ;

call FillPercentage();



-- 10

CREATE VIEW StudentPercentageDetail AS SELECT StudentRollNo, avg(StudentMarksPercentage) AS StudentAvgPercentage FROM StudentSubjectInformation GROUP BY StudentRollNo;

DELIMITER //

CREATE PROCEDURE FillScholarshipCalegory()

BEGIN

DECLARE roll\_no INT;

DECLARE avg\_percentage FLOAT;

DECLARE finished INTEGER DEFAULT 0;

DECLARE tcat VARCHAR(30);

DECLARE curs CURSOR FOR SELECT \* FROM StudentPercentageDetail;

DECLARE CONTINUE HANDLER FOR NOT FOUND SET finished = 1;

OPEN curs;

setCat: LOOP

Fetch FROM curs INTO roll\_no, avg\_percentage;

IF finished = 1 THEN

LEAVE setCat;

END IF;

IF avg\_percentage > 70 THEN

SET tcat = 'cat3';

END IF;

IF avg\_percentage > 80 THEN

SET tcat = 'cat2';

END IF;

IF avg\_percentage > 90 THEN

SET tcat = 'cat1';

END IF;

UPDATE SubjectScholarshipInformation SET ScholarshipCategory = tcat WHERE StudentRollNo = roll\_no;

END LOOP setCat;

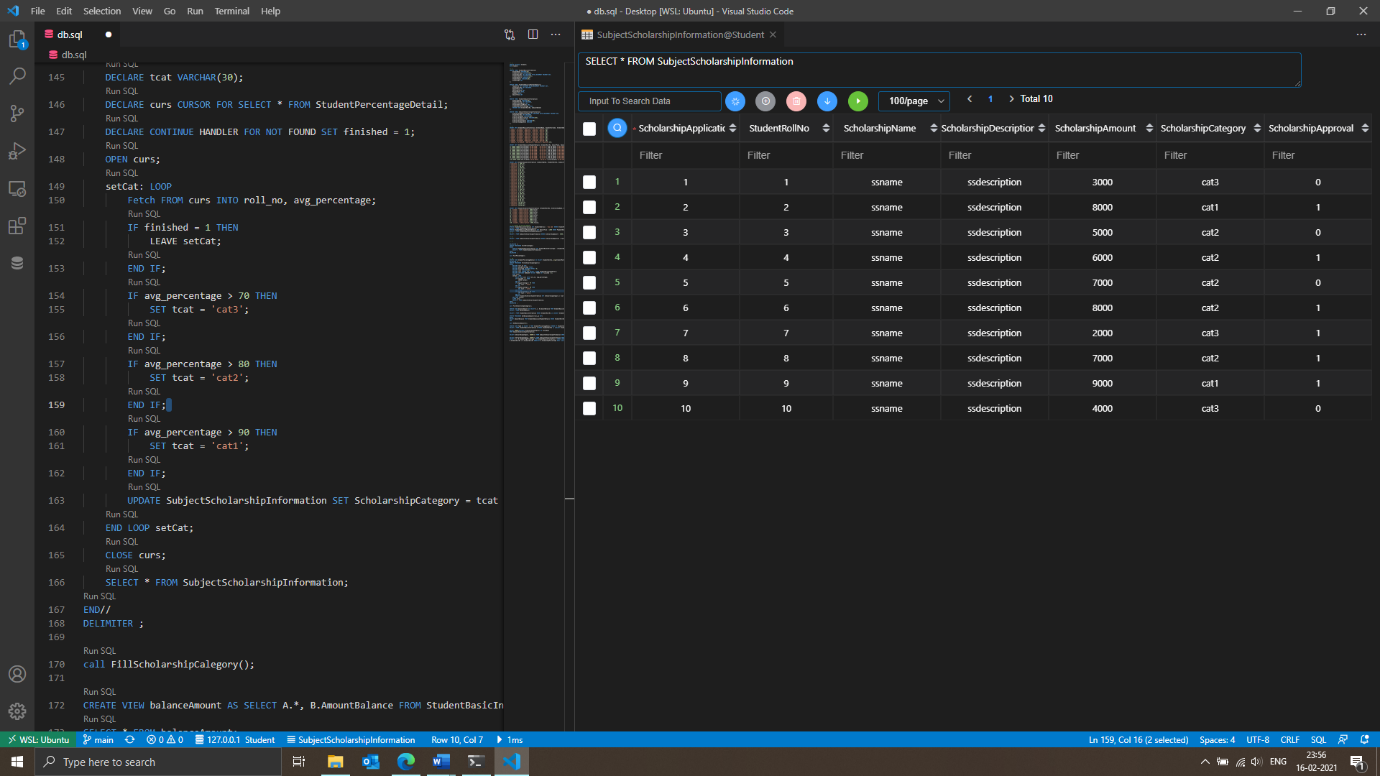
CLOSE curs;

SELECT \* FROM SubjectScholarshipInformation;

END//

DELIMITER ;

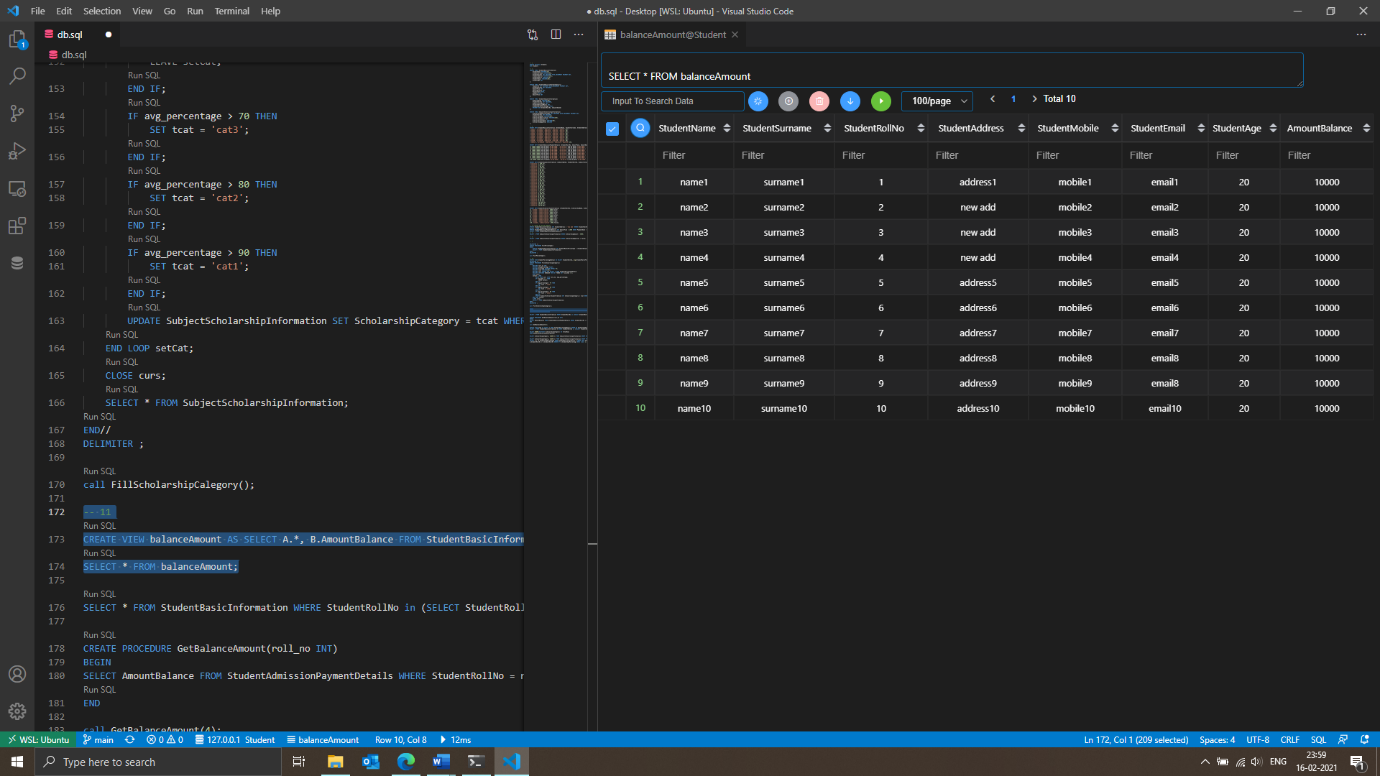
call FillScholarshipCalegory();



-- 11

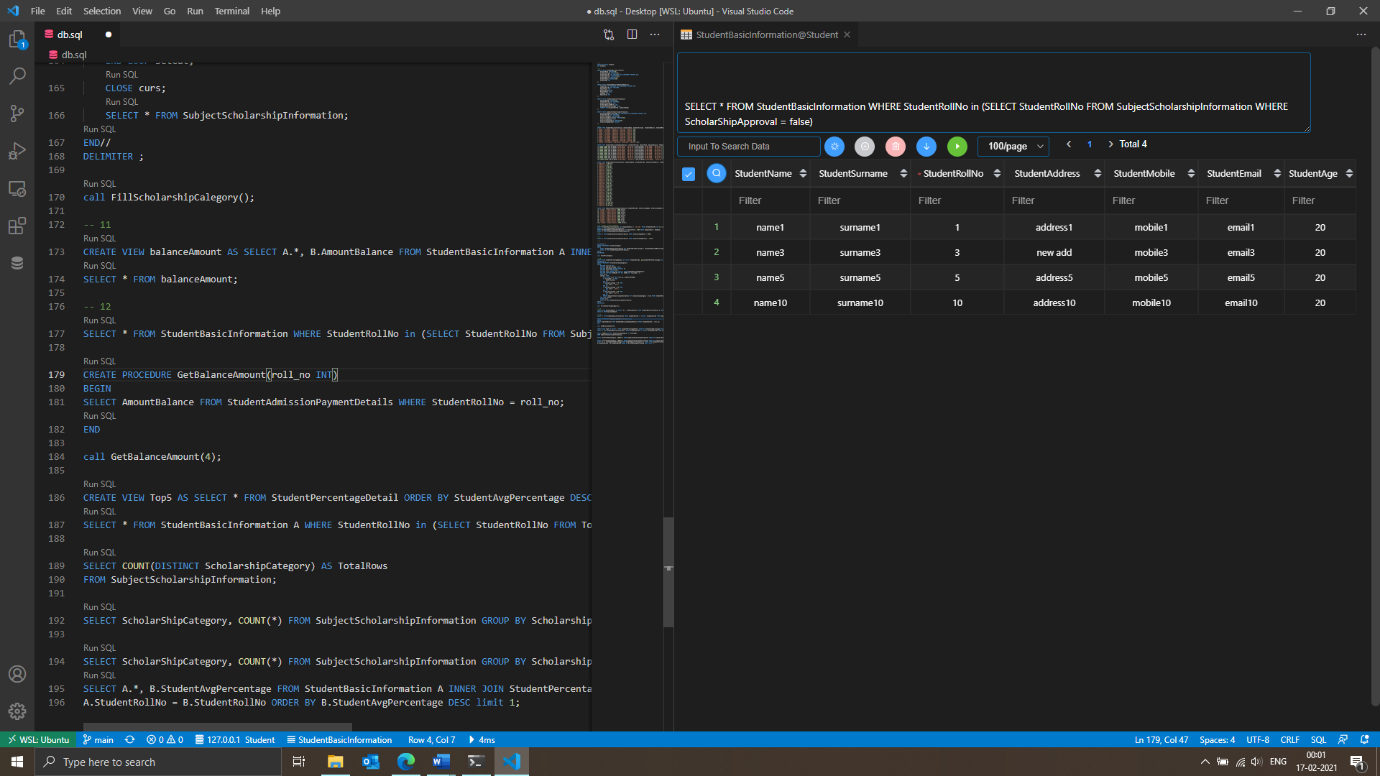
CREATE VIEW balanceAmount AS SELECT A.\*, B.AmountBalance FROM StudentBasicInformation A INNER JOIN StudentAdmissionPaymentDetails B WHERE A.StudentRollNo = B.StudentRollNo;

SELECT \* FROM balanceAmount;



-- 12

SELECT \* FROM StudentBasicInformation WHERE StudentRollNo in (SELECT StudentRollNo FROM SubjectScholarshipInformation WHERE ScholarShipApproval = false);



-- 13

DELIMITER //

CREATE PROCEDURE GetBalanceAmount(roll\_no INT)

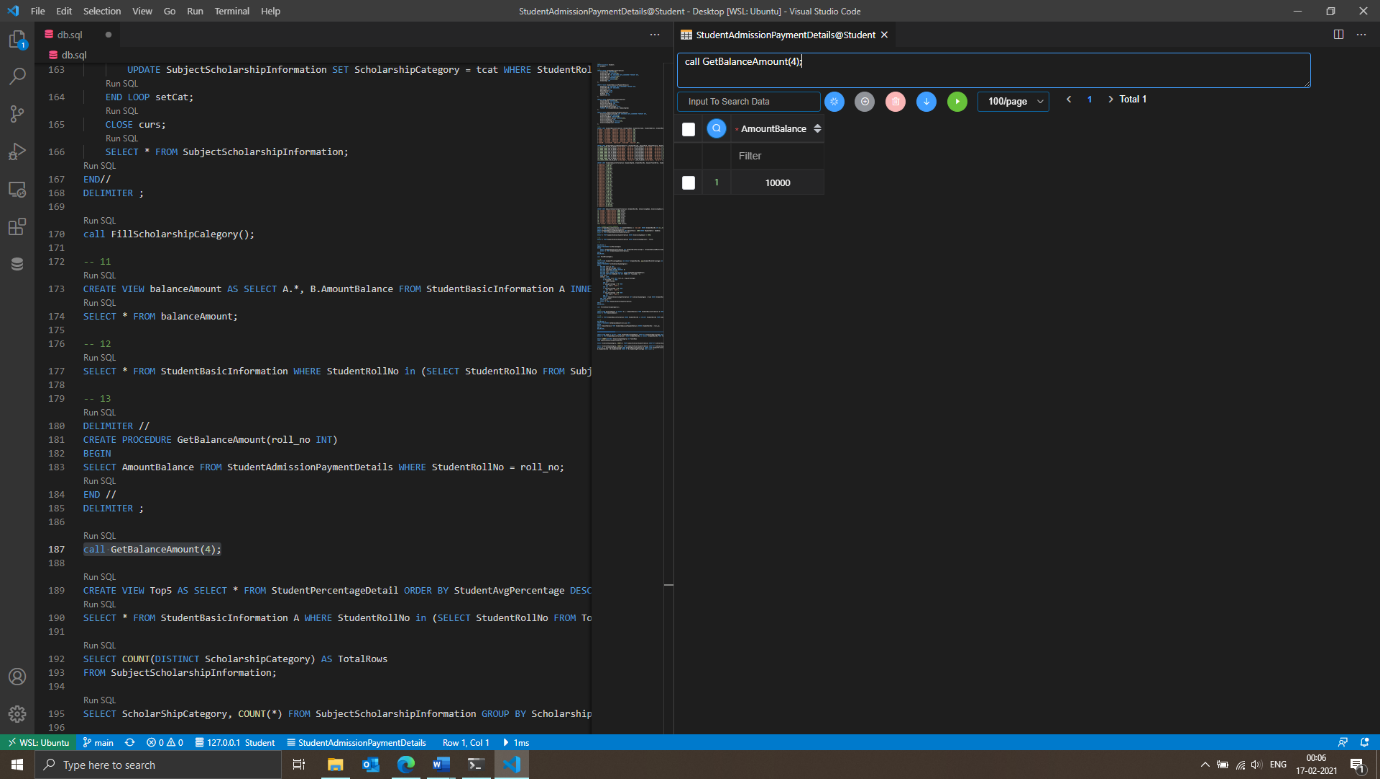
BEGIN

SELECT AmountBalance FROM StudentAdmissionPaymentDetails WHERE StudentRollNo = roll\_no;

END //

DELIMITER ;

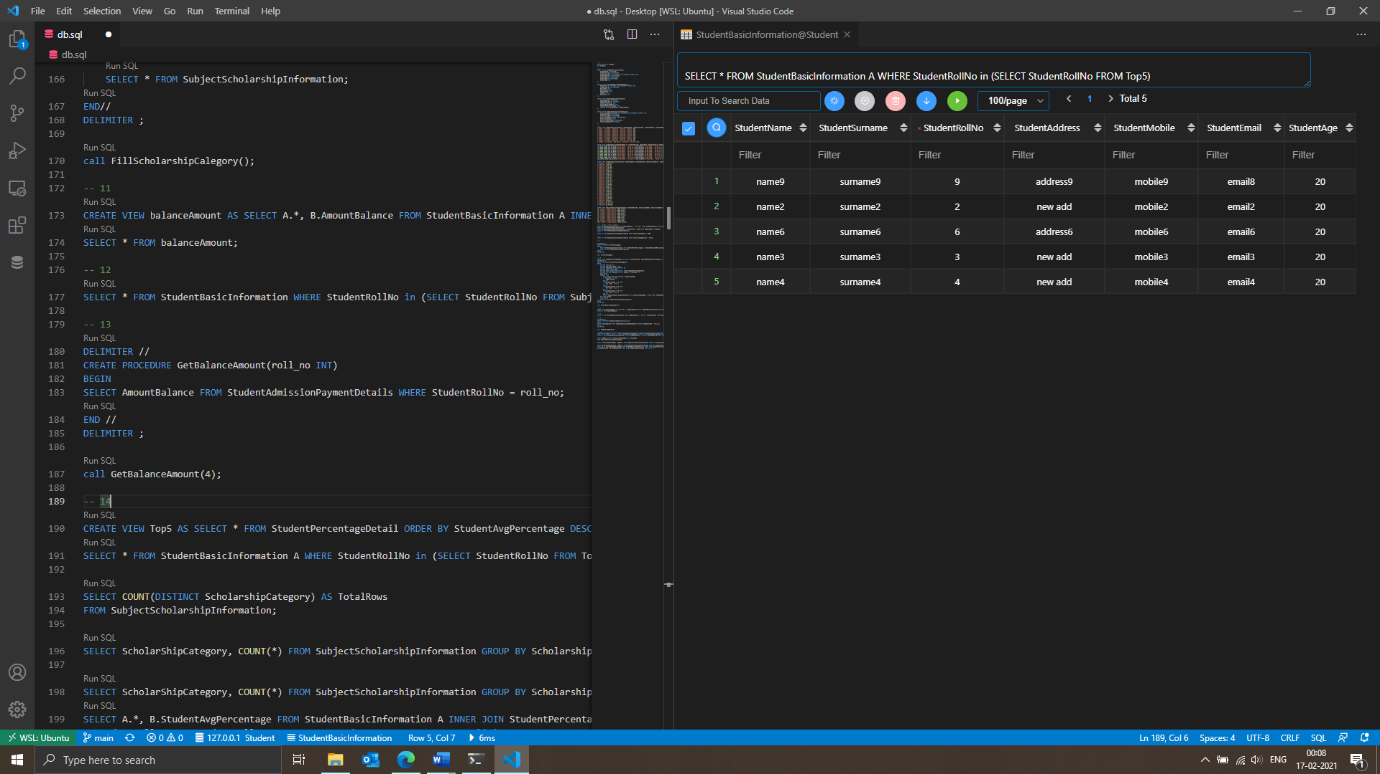
call GetBalanceAmount(4);



-- 14

CREATE VIEW Top5 AS SELECT \* FROM StudentPercentageDetail ORDER BY StudentAvgPercentage DESC LIMIT 5;

SELECT \* FROM StudentBasicInformation A WHERE StudentRollNo in (SELECT StudentRollNo FROM Top5);



-- 15

INNER JOIN (simple join)

It is the most common type of join. MySQL INNER JOINS return all rows from

multiple tables where the join condition is met.

LEFT OUTER JOIN / LEFT JOIN

This type of join returns all rows from the LEFT-hand table specified in the ON condition

and only those rows from the other table where the joined fields are equal.

RIGHT OUTER JOIN / RIGHT JOIN

This type of join returns all rows from the RIGHT-hand table specified in the ON condition

and only those rows from the other table where the joined fields are equal.

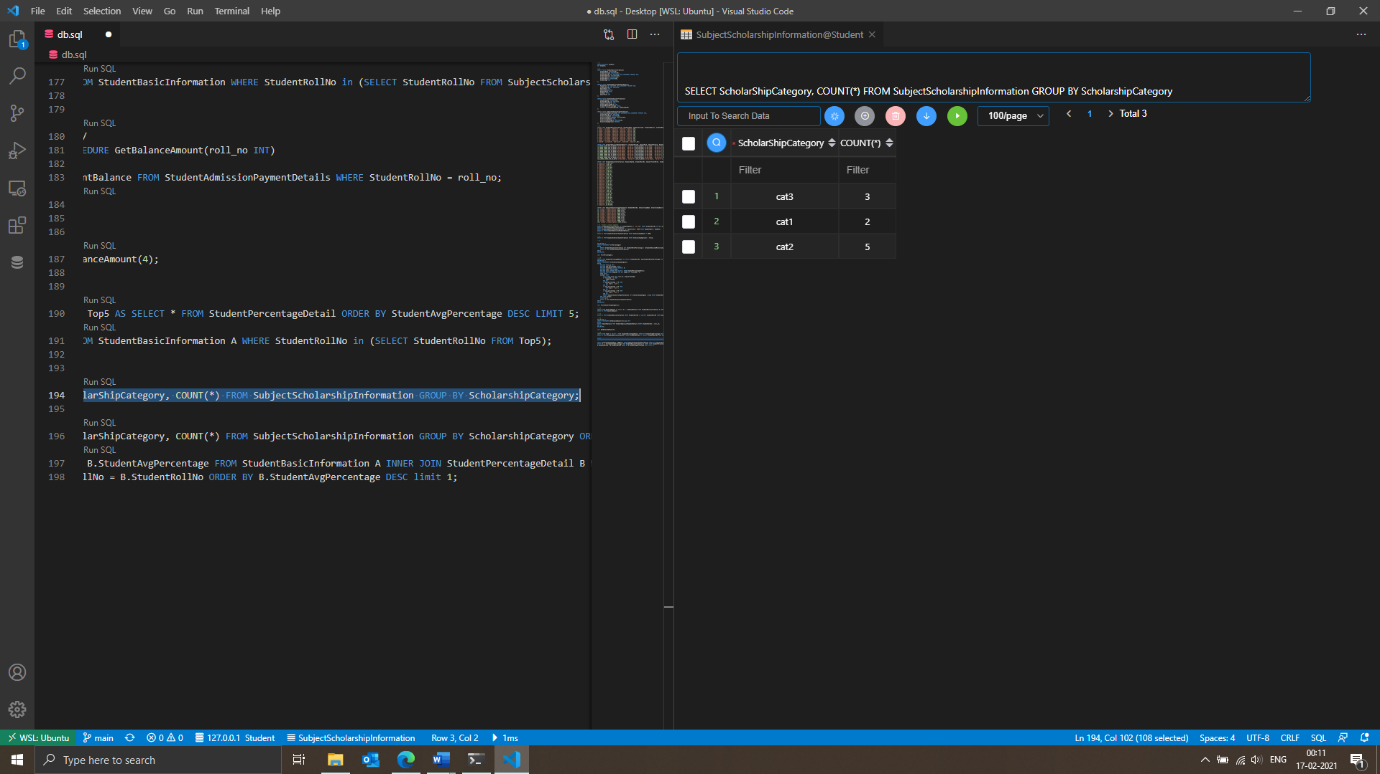
-- 16

DELETE – It is a DML command, used to delete rows from a table. User can apply WHERE clause to delete specific rows. The selection is deleted one by one. It does not free the memory of deleted rows.

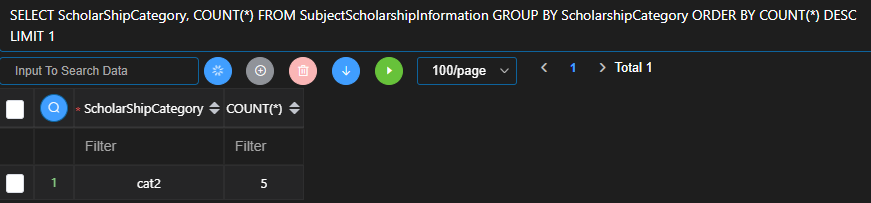
TRUNCTE – It is a DDL command, used to delete all the rows from the table in one go, but the table remains. It frees the memory.

DROP – It is a DDL command, used to drop the entire table in one go.

-- 17

SELECT ScholarShipCategory, COUNT(\*) FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory;

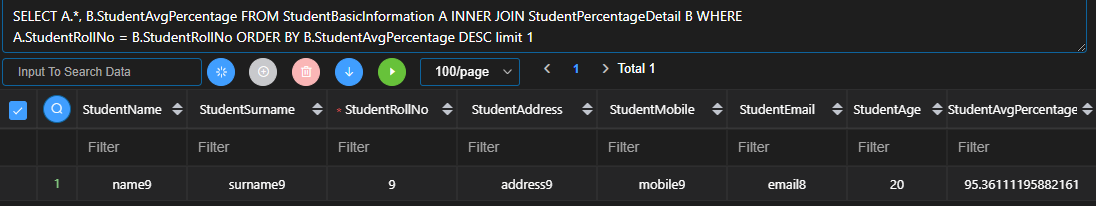
-- 18

SELECT ScholarShipCategory, COUNT(\*) FROM SubjectScholarshipInformation GROUP BY ScholarshipCategory ORDER BY COUNT(\*) DESC LIMIT 1;

-- 19

SELECT A.\*, B.StudentAvgPercentage FROM StudentBasicInformation A INNER JOIN StudentPercentageDetail B WHERE

A.StudentRollNo = B.StudentRollNo ORDER BY B.StudentAvgPercentage DESC limit 1;



-- 20

a) Triggers – A trigger is a special type of stored procedure that automatically runs when an event (insert, update, delete) occurs in the database server.

b) Stored Procedure – A stored procedure is a set of Structured Query Language (SQL) statements with an assigned name, they can be reused and shared by multiple programs.

c) View – A view is a virtual table based on the result set of an SQL query.

d) Functions – Set of SQL statement, which can take only input and always returns an output. User can only perform DQL inside a function.