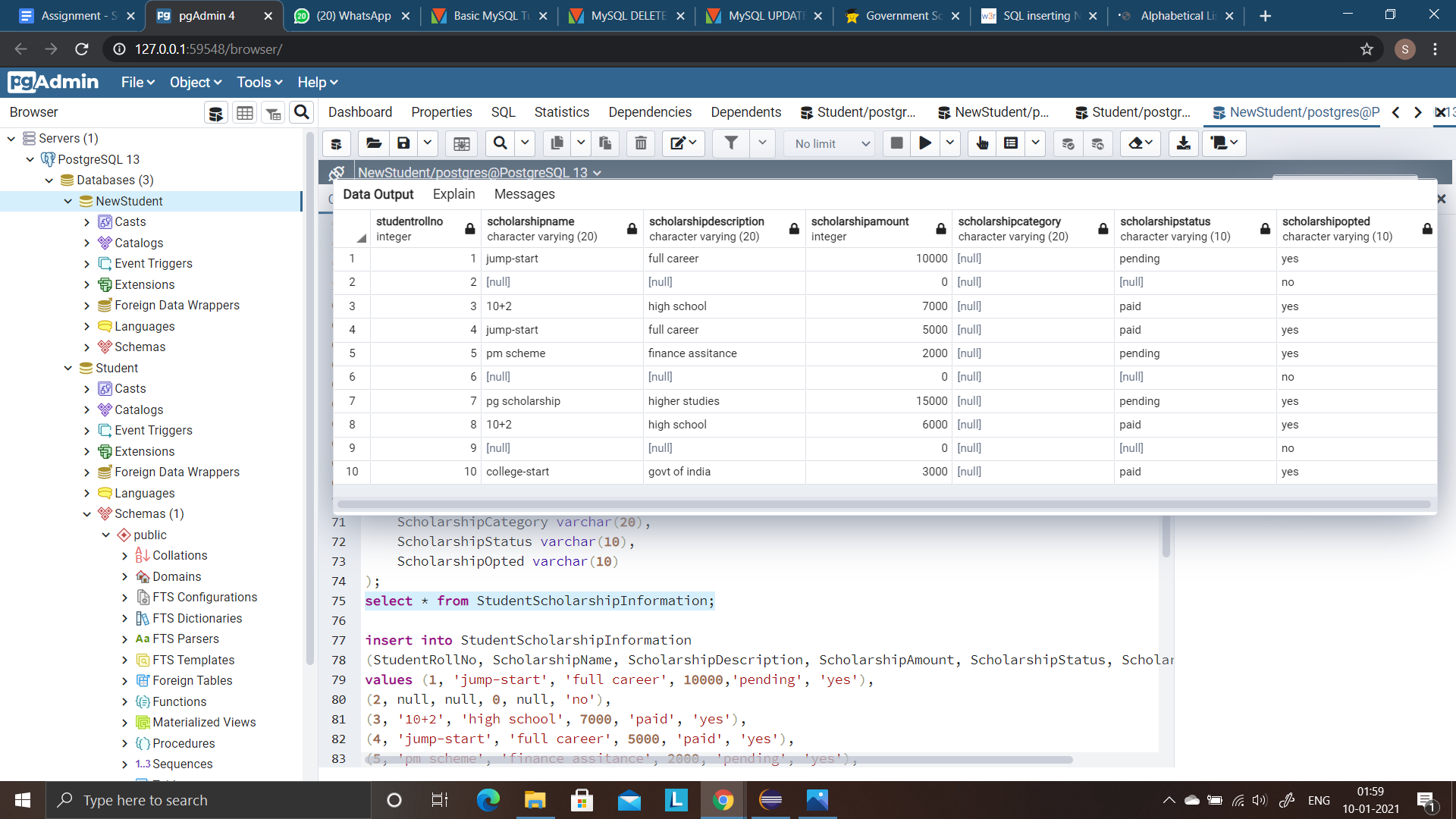
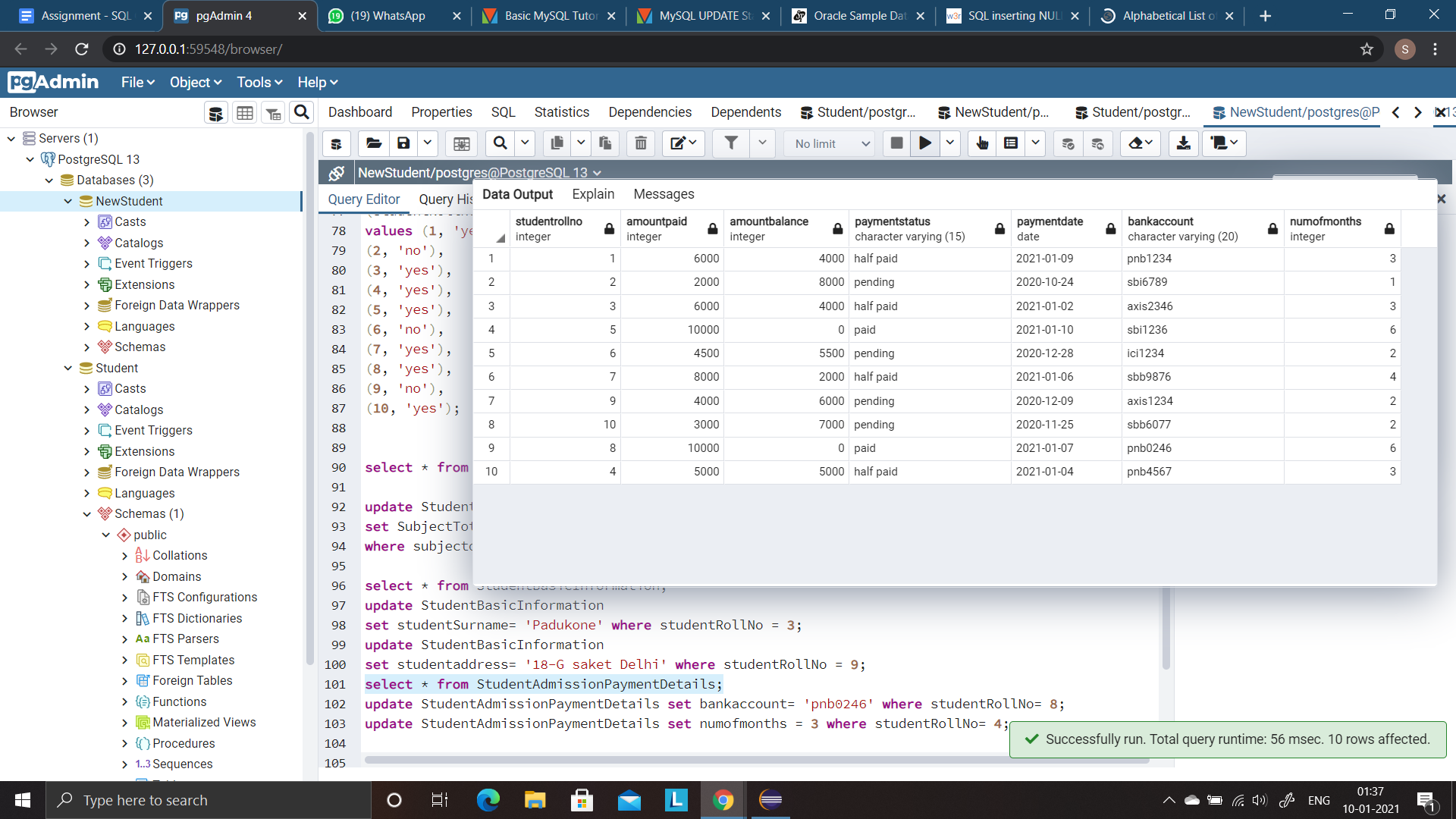
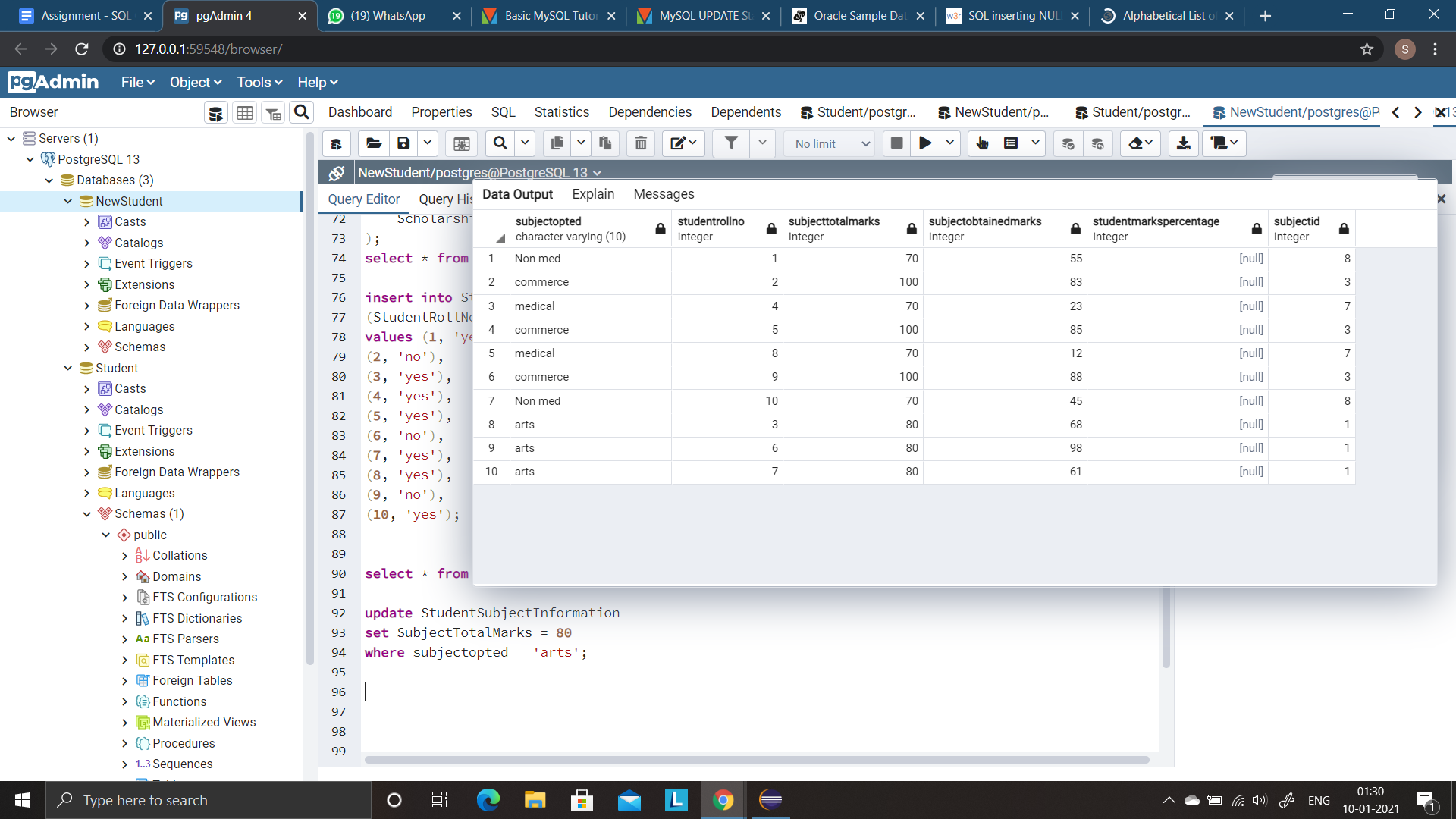
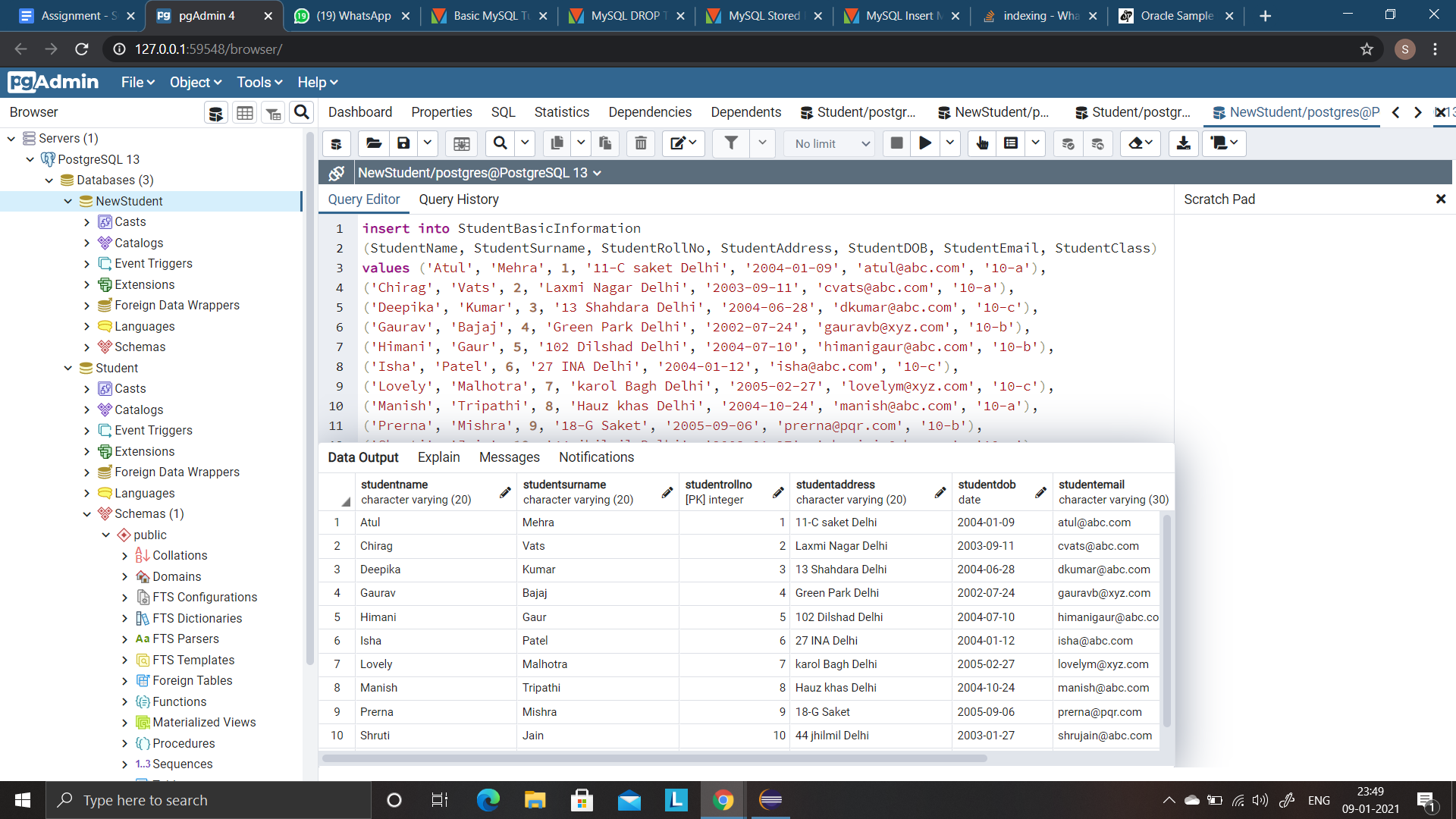
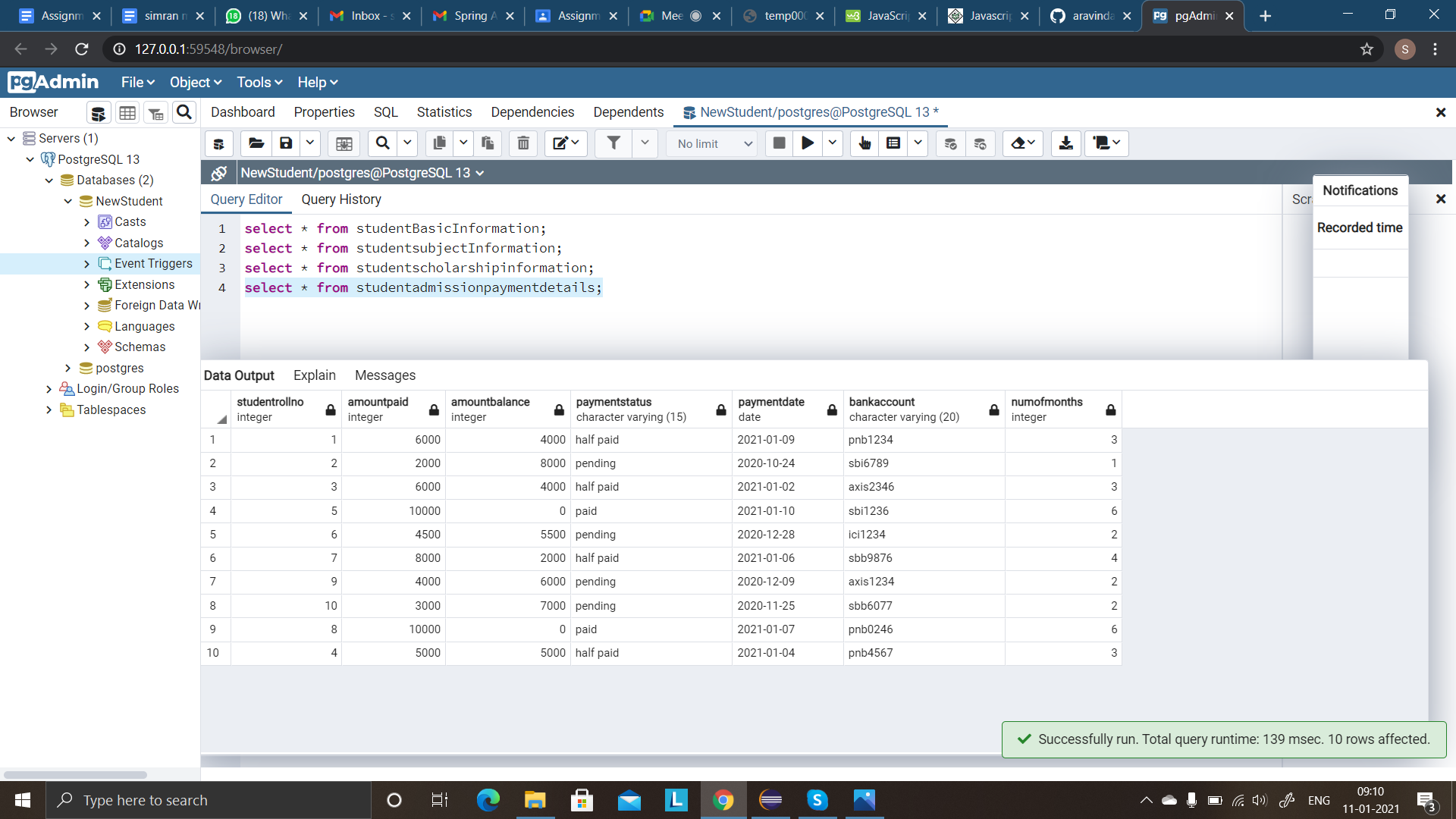
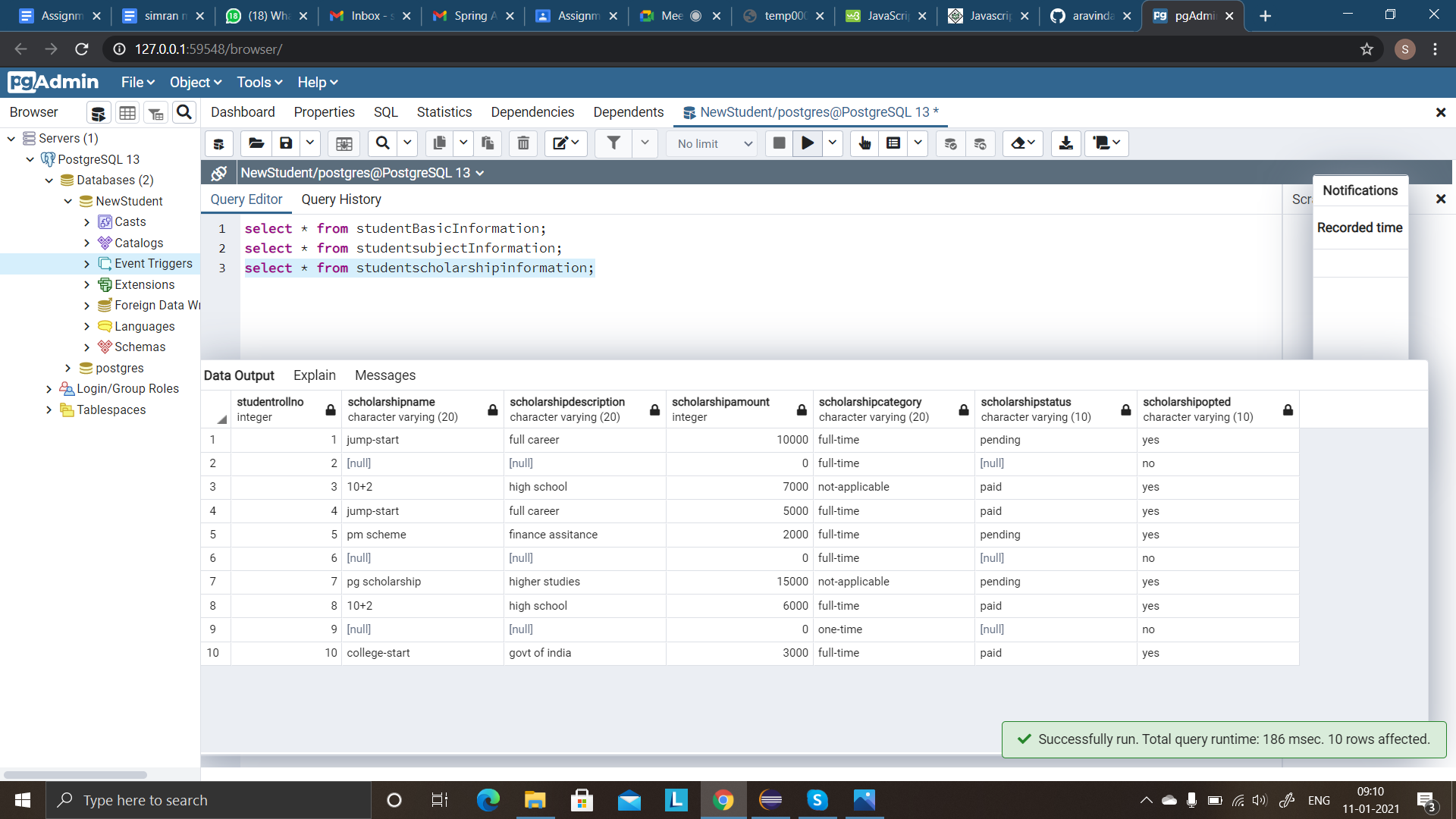
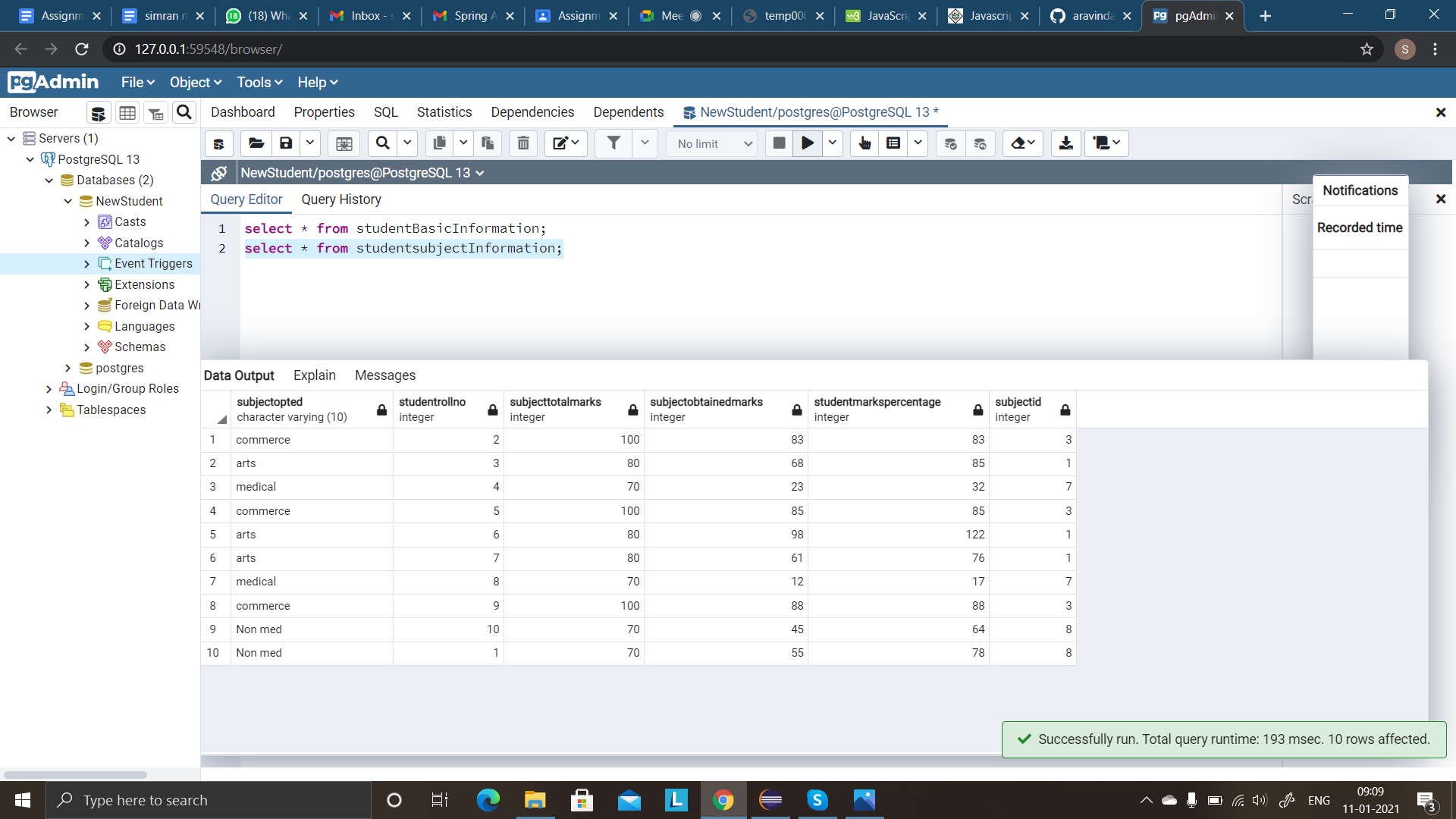
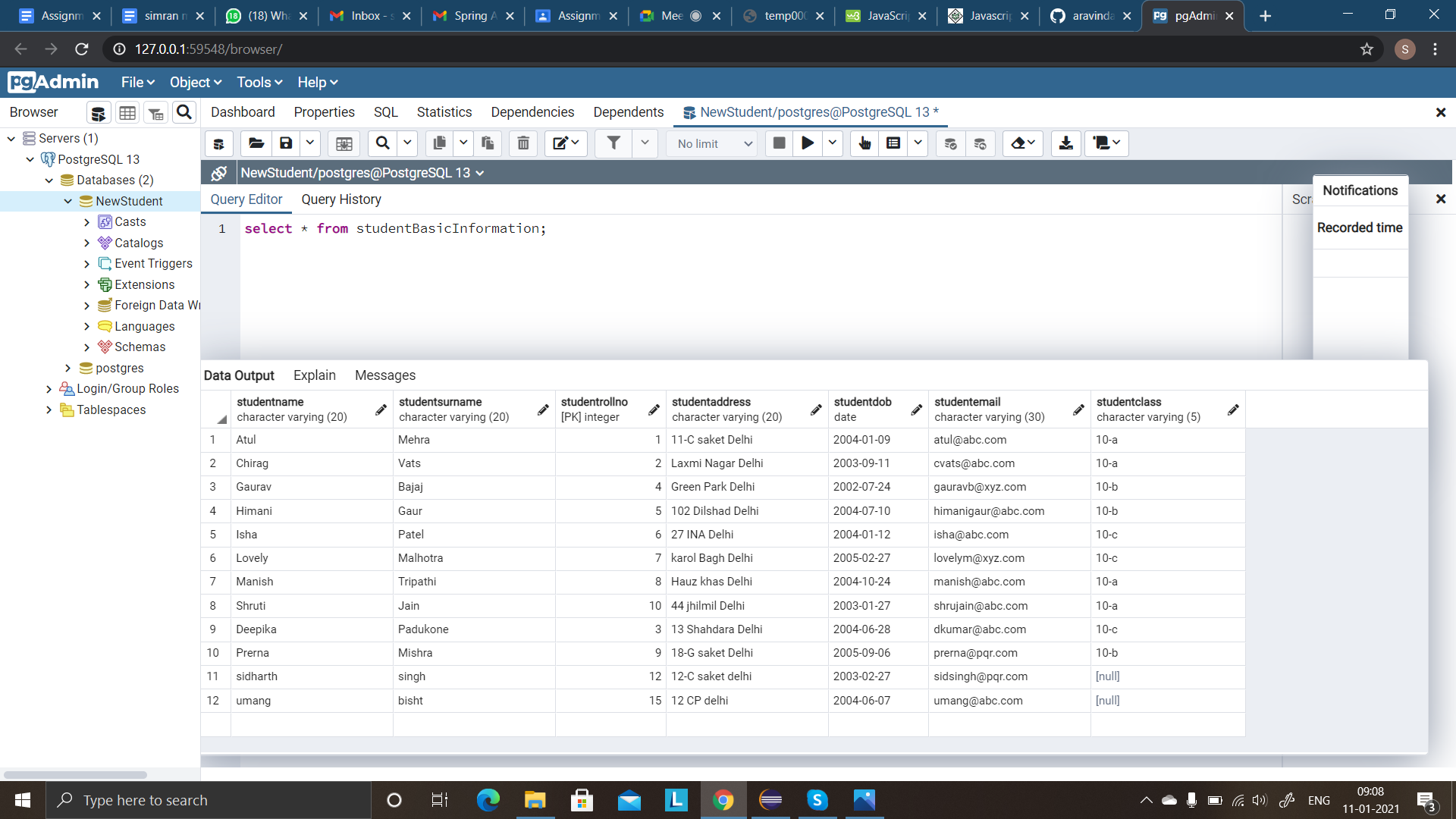
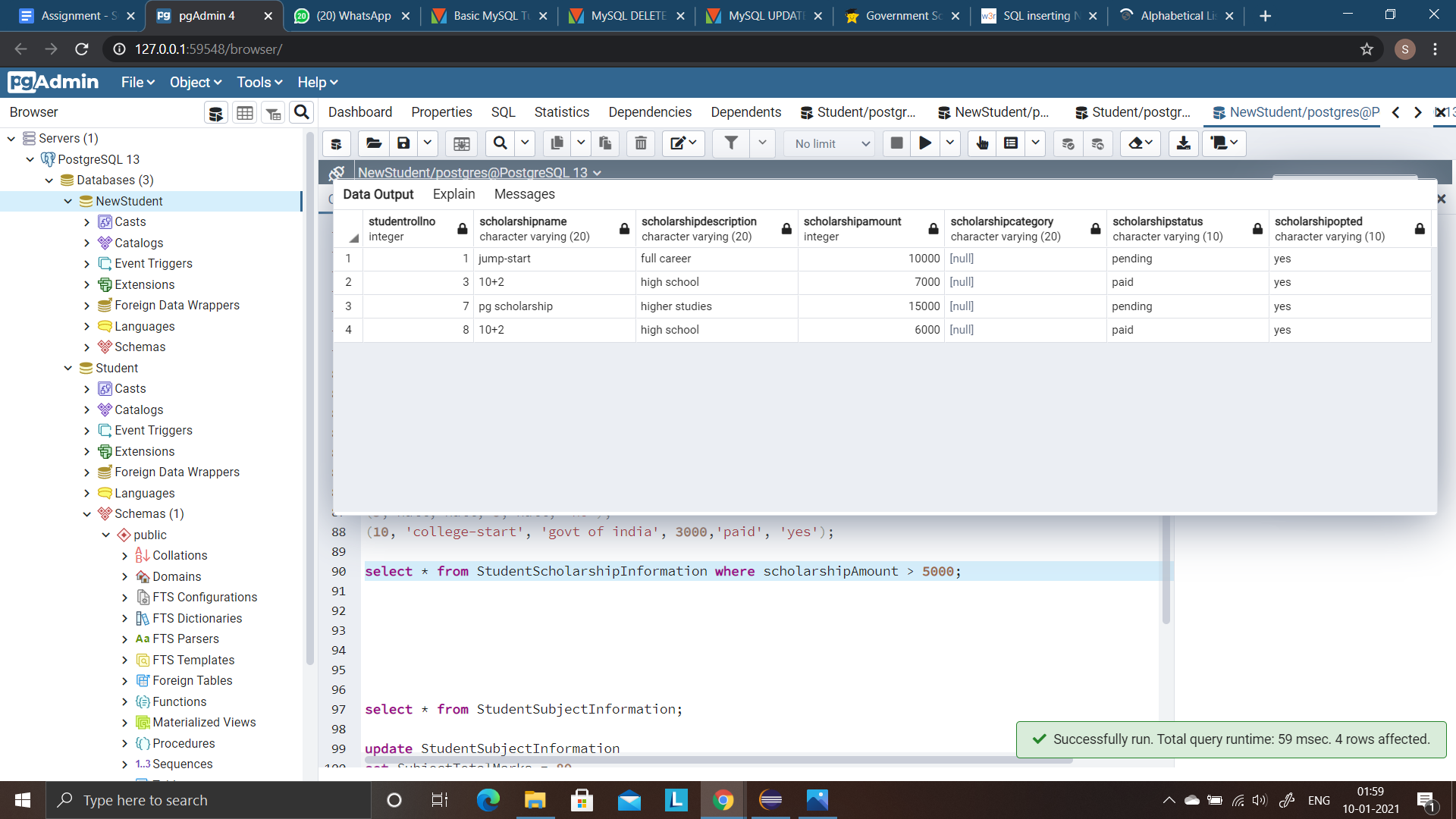
**Q4**

**Q6**

****

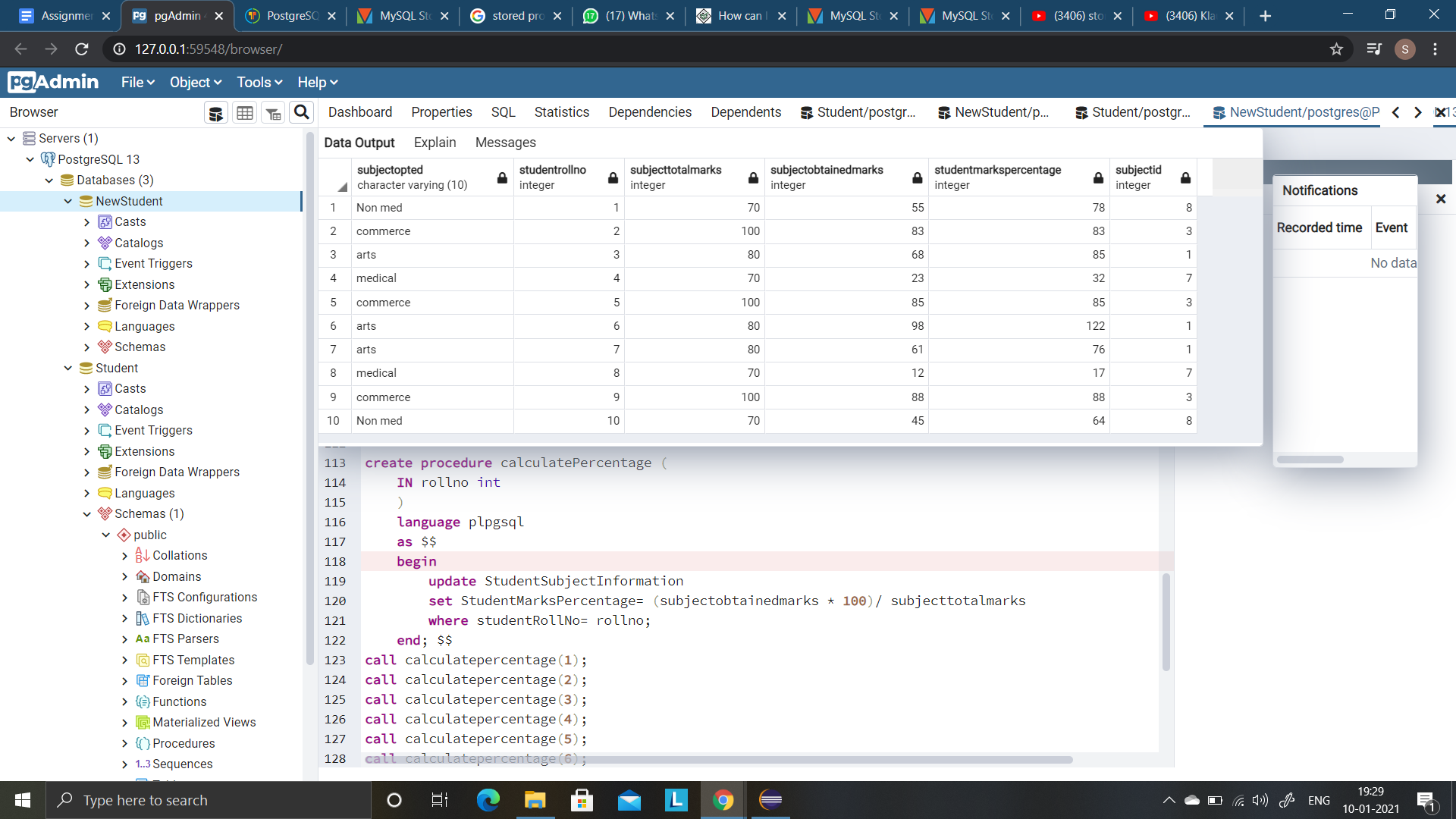
**Q7**

****

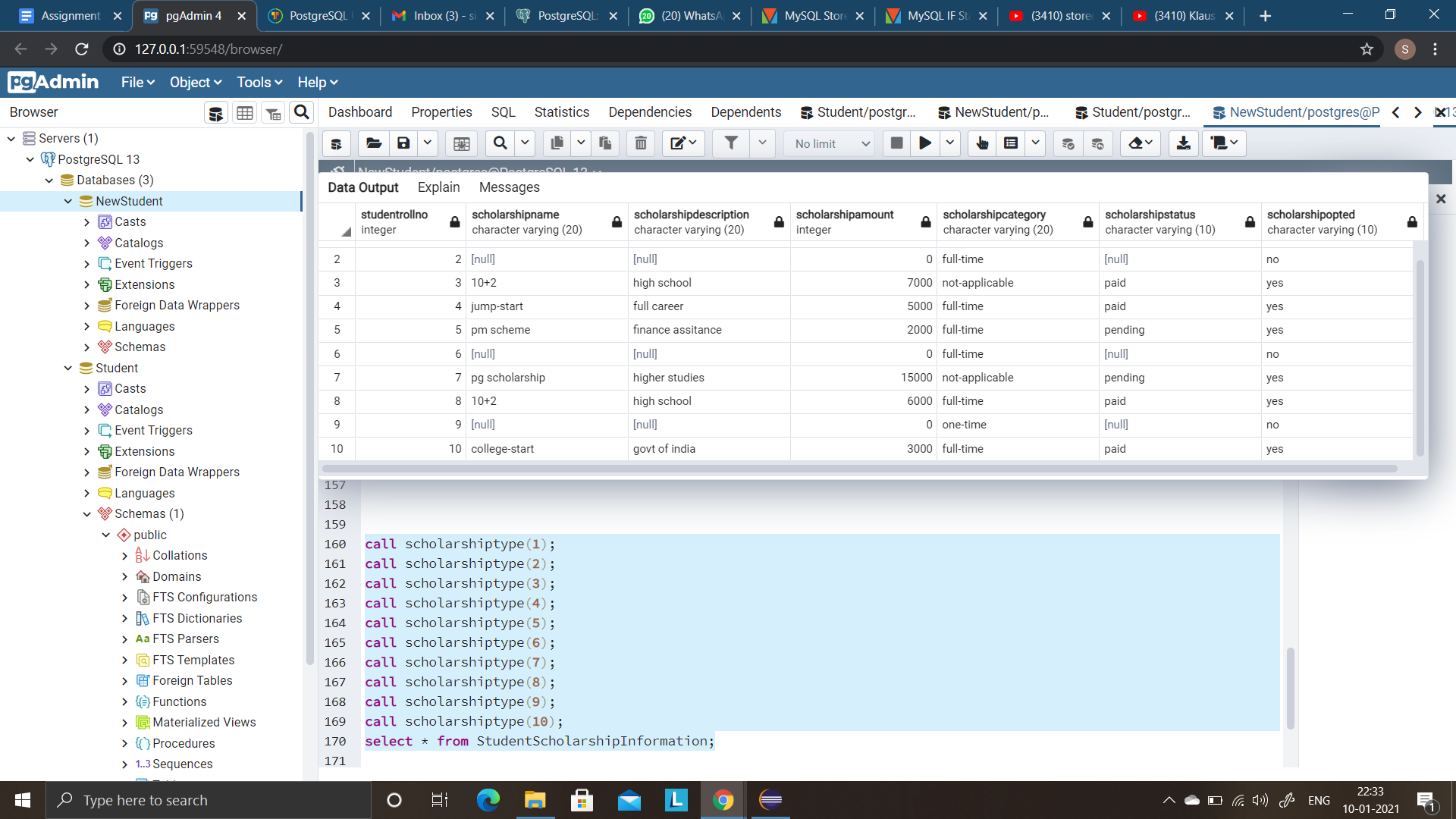
**Q8**

****

**Q9**

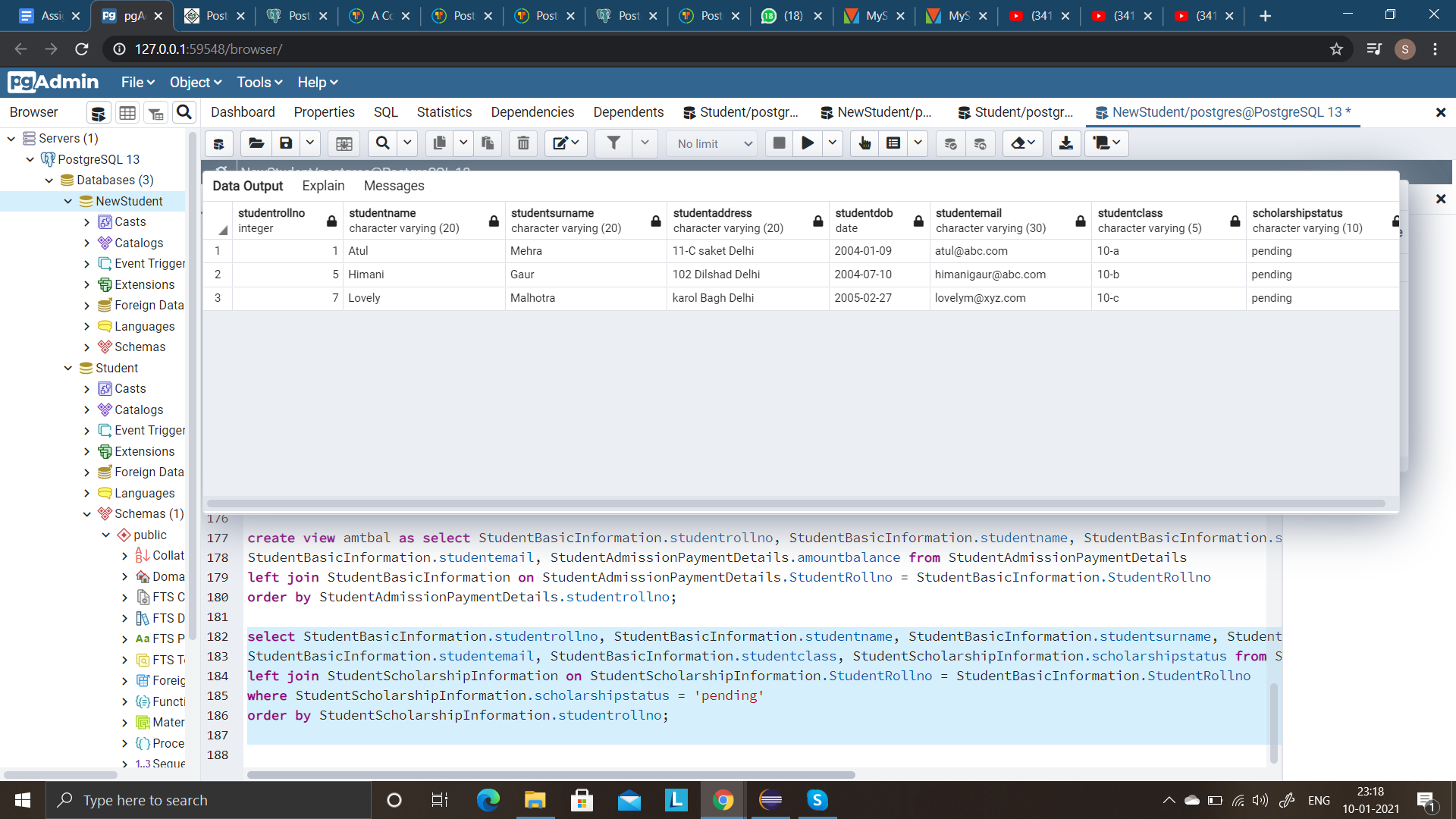
****

**Q10**

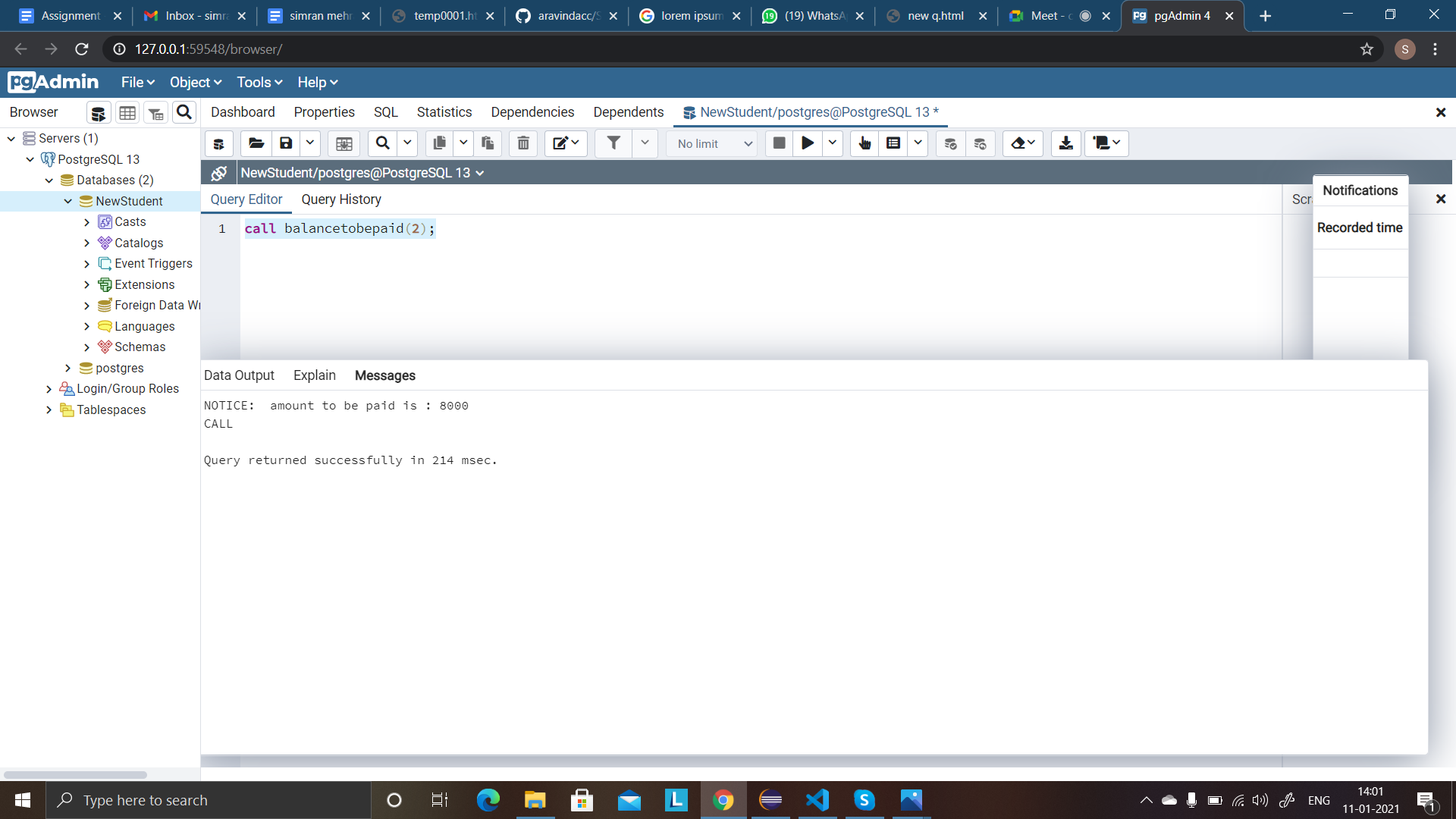
****

**Q11**

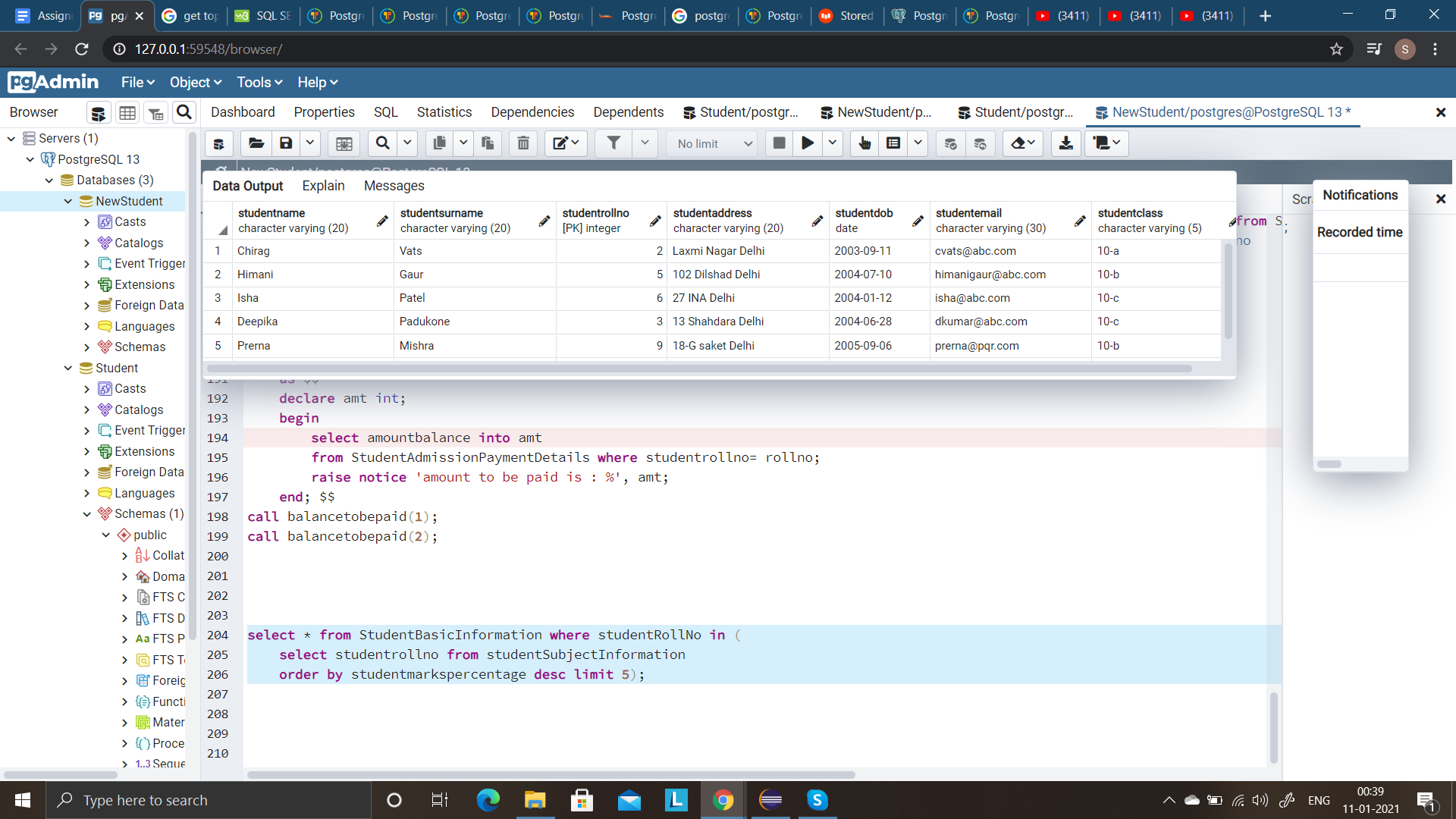
****

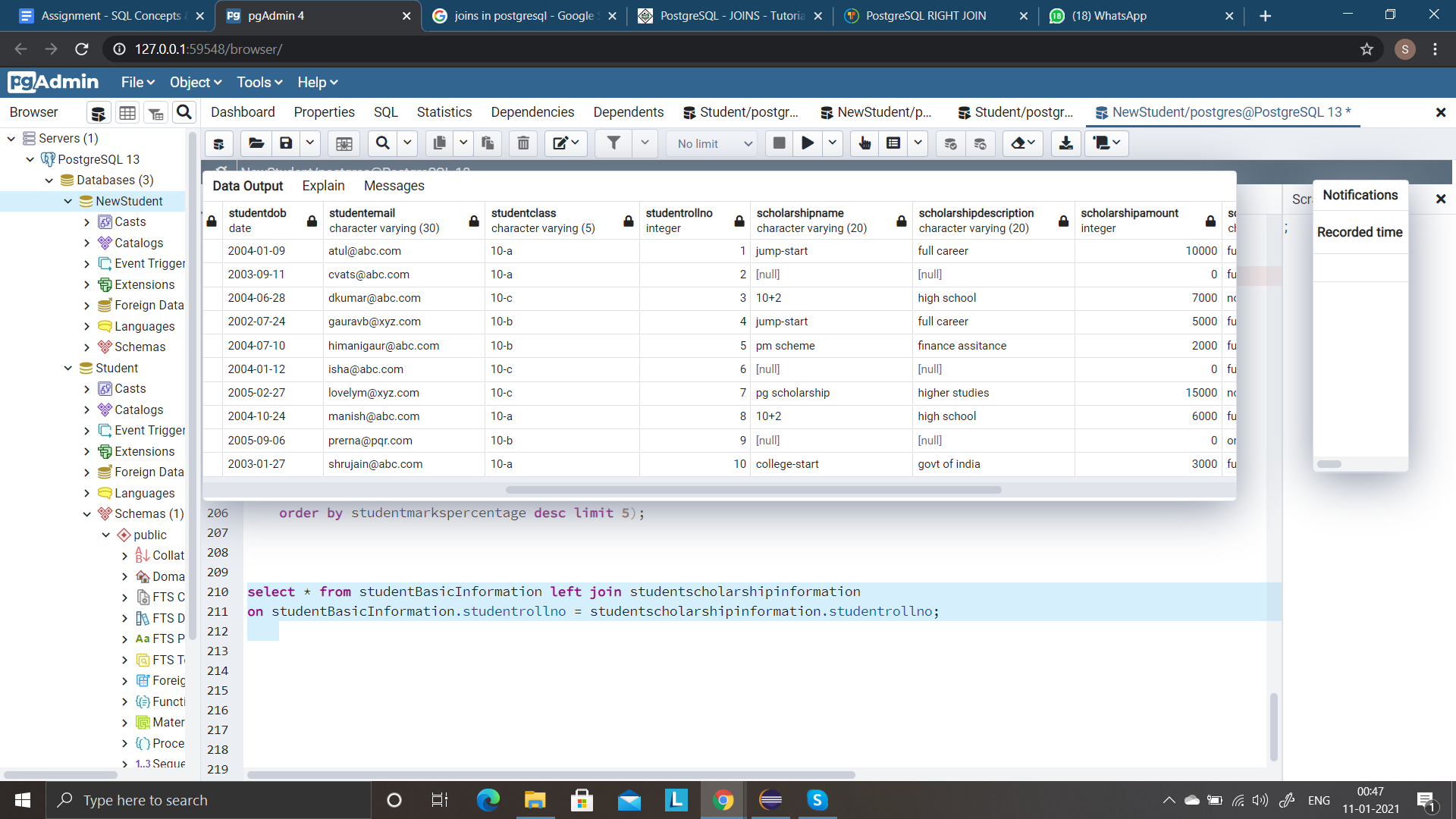
**Q12**

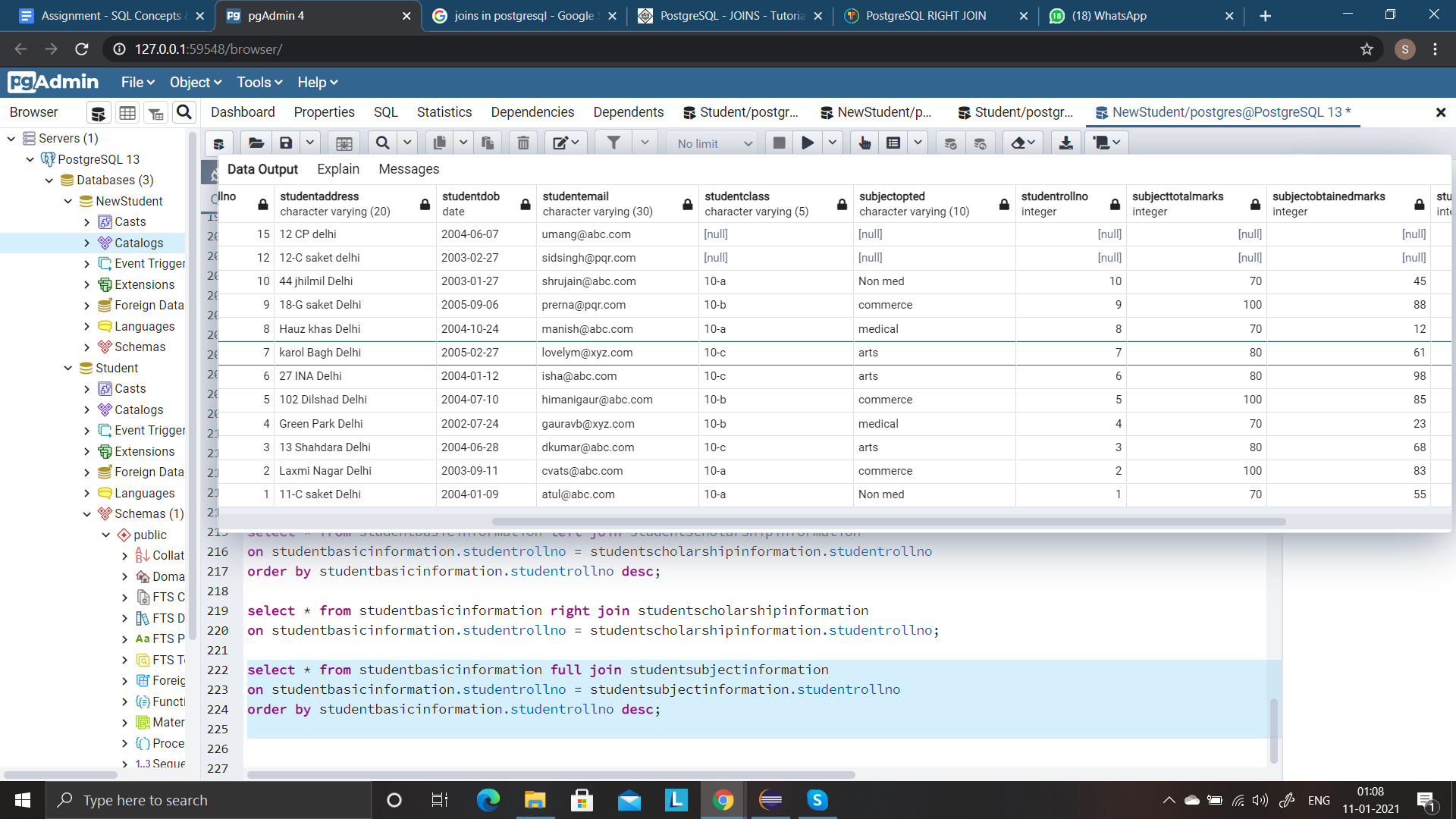
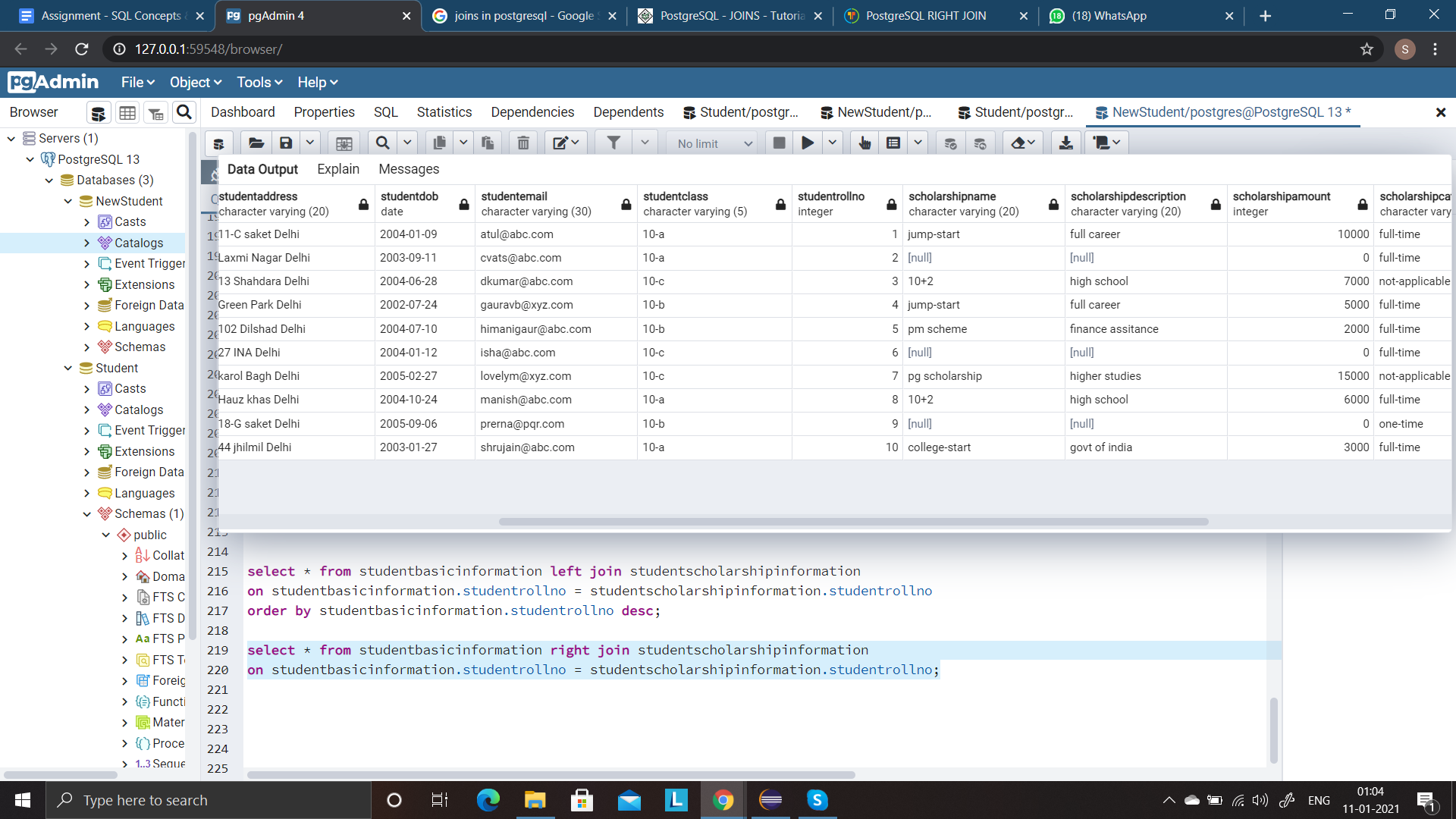
**Q13**

****

**Q14**

****

**Q15**

****

**Q16**

**DELETE** :

* Delete is a data manipulation language (DML) command.
* DML commands allow changing data within the database.
* Delete removes all or specified records (rows) from the table.
* The space for the table remains as it is.
* It doesn’t delete the table from the database.
* We can delete a single record or multiple or all records depending on the condition specified in the query. (conditions are specified using WHERE clause of DELETE statement, if omitted deletes all records of the table.
* We can use the ROLLBACK command to restore the records deleted

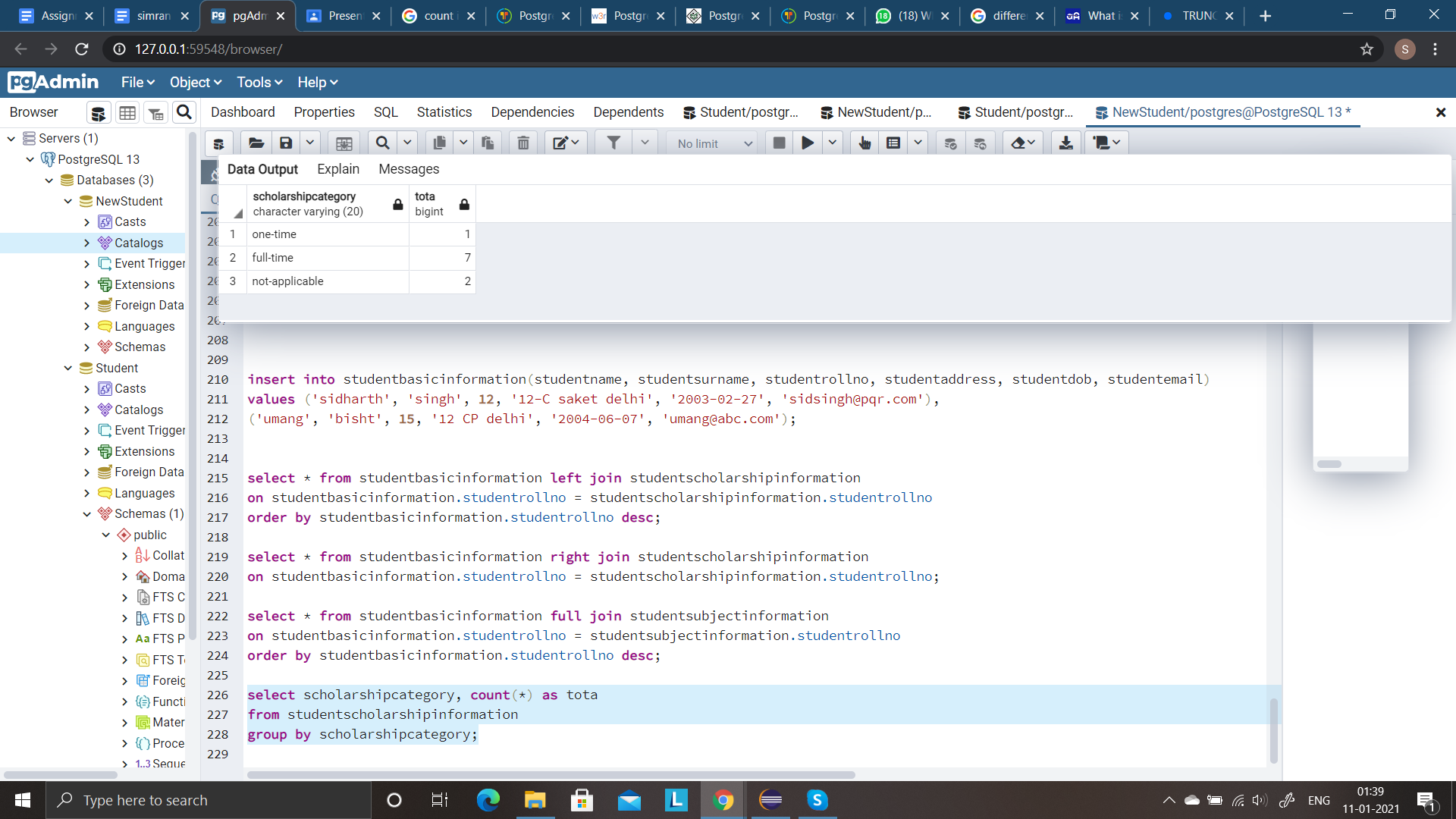
**DROP** :

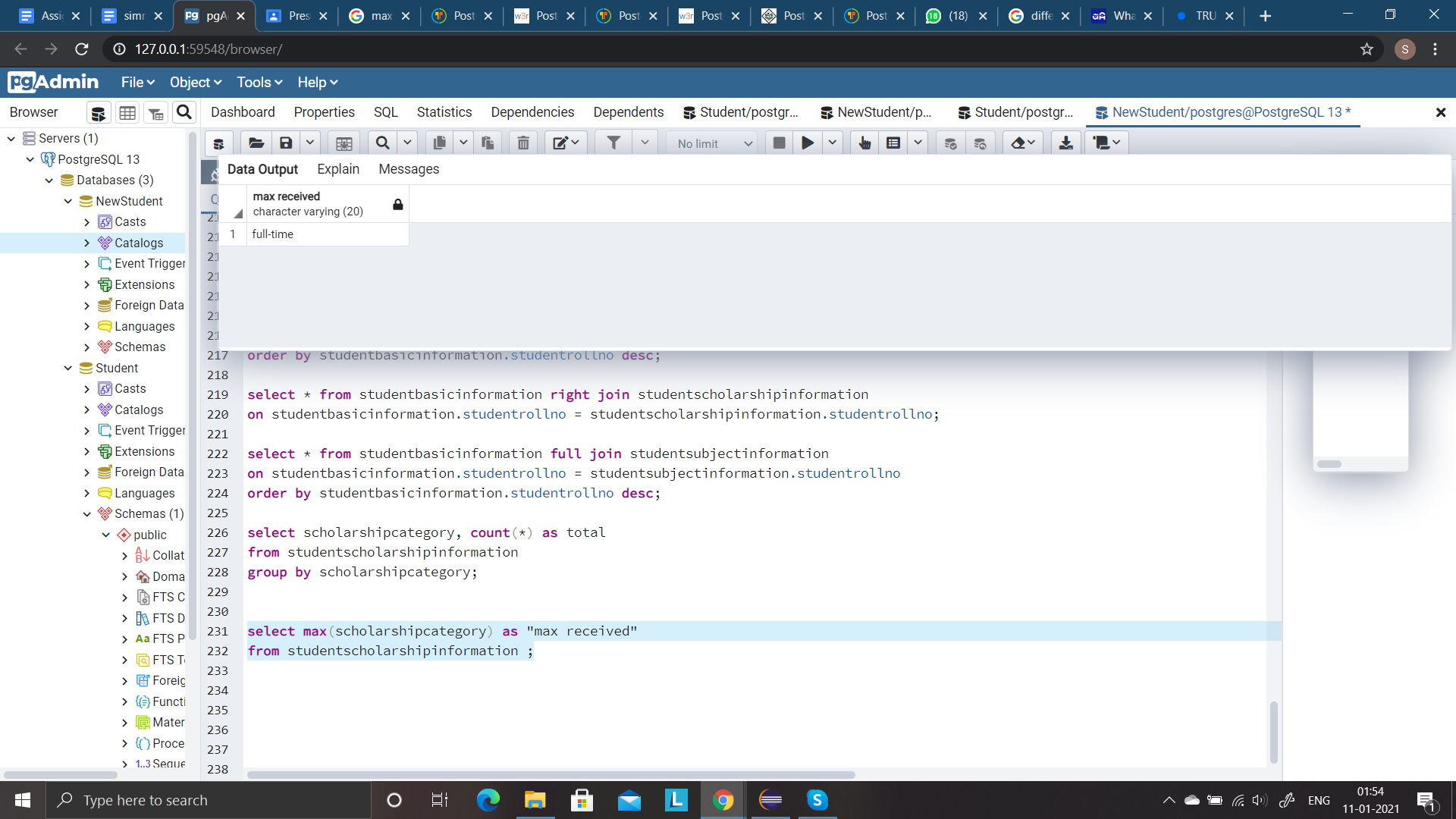
* Drop is a data definition language (DDL) command.
* These commands are normally not used by a general user. They are used by the DBA
* Drop deletes objects (databases, tables, views, triggers, etc) from the database
* Drop removes the entire structure of the table in one go along with the records.
* It deletes the table from the database.
* It deletes the entire existence of the table
* We cannot use the ROLLBACK command.

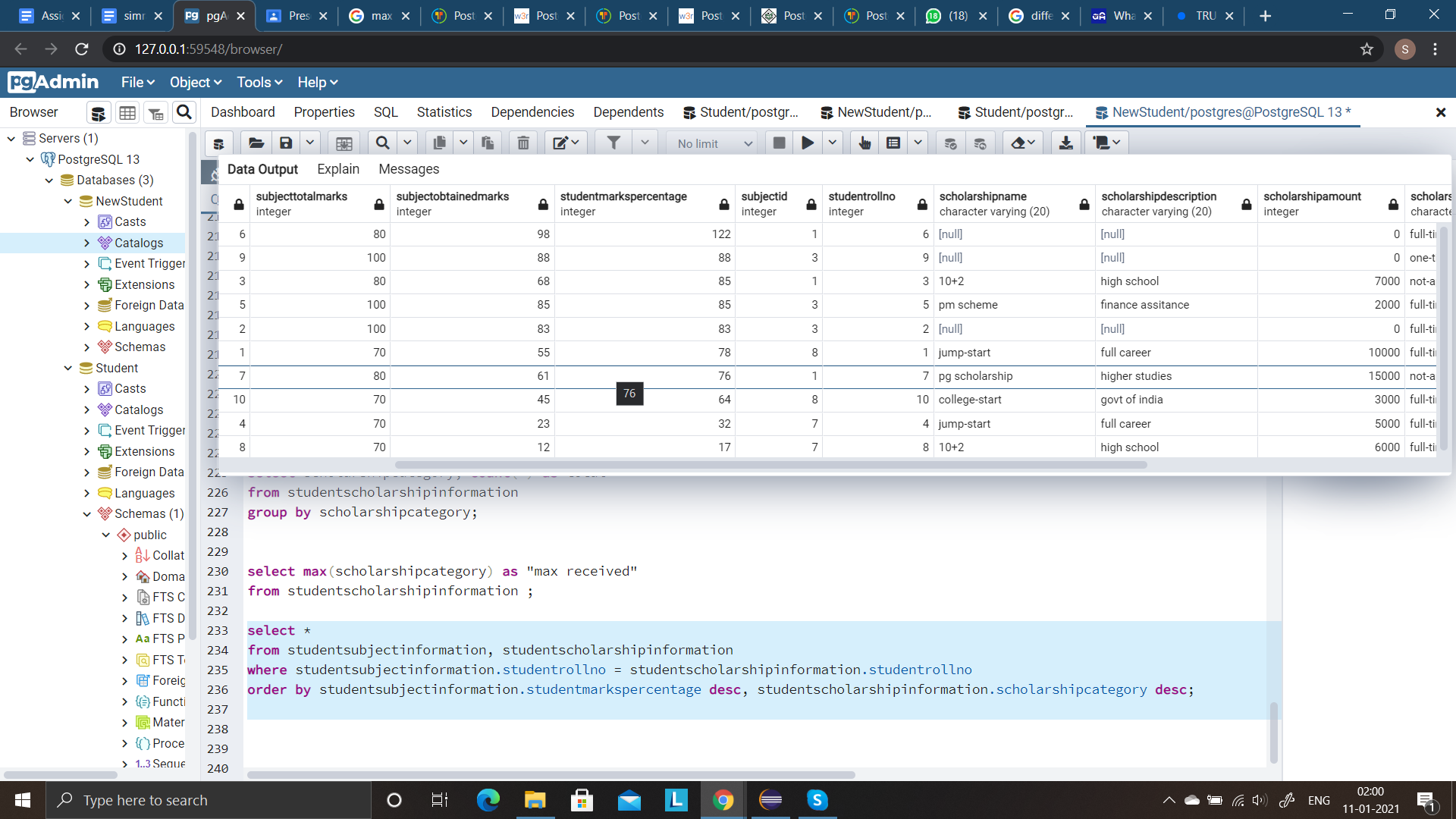
**TRUNCATE** :

* Truncate is also a data definition language (DDL) command.
* These commands are normally not used by a general user. They are used by the DBA
* Truncate removes all records from the table
* The space for the table is removed as well.
* It deletes all rows of a table in one go.
* Single or specified rows cannot be deleted with truncate.
* It is comparatively faster than Delete command
* We cannot use the ROLLBACK command.

**Q17**

****

**Q18**

**Q19**

**Q20**

**VIEW**

View is not a physical table, but rather, it is in a virtual table created by a query joining one or more tables.

Views does the following:

* Structure data in a way that users or classes of users find natural or intuitive.
* Restrict access to the data such that a user can only see limited data instead of a complete table.
* Summarize data from various tables, which can be used to generate reports.

Cannot execute a DELETE, INSERT, or UPDATE statement on a view.

Views are most commonly used in conjunction with joins.

We can drop a view.

**STORED PROCEDURES** :

Stored procedures are a piece of the code written to do some specific task. Stored procedures can be invoked explicitly by the user. It's like a java program , it can take some input as a parameter then can do some processing and can return values.

* We can execute the stored procedures when required.
* It can take input as a parameter. A stored procedure can accept zero or more parameters. Three types are : IN, OUT, INOUT
* It can return values.

Stored procedure has a declaration section (option) to declare variables.

We can drop a stored procedure as well.

Syntax:

create [or replace] procedure procedure\_name(parameter\_list)

language plpgsql

as $$

declare

*-- variable declaration*

begin

*-- stored procedure body*

end; $$

**TRIGGERS** :

Triggers are database callback functions, which are automatically performed/invoked when a specified database event occurs.

* It can execute automatically based on the events
* It can not take input as parameter
* Triggers can not return values

Syntax :

CREATE TRIGGER trigger\_name [BEFORE|AFTER|INSTEAD OF] event\_name

ON table\_name

[

-- Trigger logic goes here....

];

* The BEFORE, AFTER or INSTEAD OF keyword determines when the trigger actions will be executed relative to the insertion, modification or removal of the associated row.
* Triggers are automatically dropped when the table that they are associated with is dropped.

**FUNCTIONS** :

Functions is a set of SQL statements that accept only input parameters and produce output in a single value form or tabular form.

Functions allow database reuse as other applications can interact directly with your stored procedures instead of a middle-tier or duplicating code.

* Function must return any value.
* It cannot modify the data received as parameters

Syntax:

CREATE [OR REPLACE] FUNCTION function\_name (arguments)

RETURNS return\_datatype AS $variable\_name$

DECLARE

declaration;

[...]

BEGIN

< function\_body >

[...]

RETURN { variable\_name | value }

END; LANGUAGE plpgsql;