SECTION A

A1 and A2.

Using the enclosed reporting would fulfill a business need for the video stores to be able to filter their inventory by release year, the age rating, and the clarified rental duration that someone holds the movie. The use for this process is to quickly find the newest movies of the year for any given age range. When there are children present in the store, usually there are busy parents behind them wanting to get in and out as fast as possible. This allows the store to help those busy parents to find the movies their kids want, but also that they want for when the kids are in bed.

This would also give the choice of looking to see if a particular movie was being kept for long periods of time or not as well, which can indicate many things.

The detailed table would allow for a more in-depth search of the inventories, but the summary table is where the quickness would follow for those parents or just any busy person. Additionally, the extra facts would help advise any customer if they were looking for a popular movie, or searching by year, or any number of ways to search through the data.

The data used for this report and both tables would be the Inventory table and Film tables, both providing pertinent information to the stakeholders regarding popularity of films to hit shelves, but also in what areas certain films are more popular than others. This could help them decide which inventory to reduce to introduce when adding new inventory. These tables also provide the necessary information to provide these reports. The reports generated could give data on the rental duration of certain films, and what age range they are rated for. This could also help predict which movies are going to be brought into the store next per location.

A3. The specific fields being used in the detailed\_report:

film\_id INT ,

title varchar(45),

rental\_rate varchar(45),

rental\_duration INT,

inventory\_id INT,

store\_id INT,

release\_year INT,

rating varchar(45)

The fields necessary for the summary\_table are:

title varchar(45),

rating varchar(45),

release\_year INT,

count\_inventory\_count INT

A4. TRANSFORMATION

The rental duration was changed during the process to show the clear words of “low rental”, “medium rental” and “high rental” to help gauge if people are keeping the movies for a long time, or immediately returning them after a day. It helps to show what the numbers in the original column mean. Easier to read also means less likely for someone to misread the information being reported.

A5. The business use of the detailed table is having all the stats on file to decide whether to part with some of the copies of certain movies if being used by the stakeholders, but if being used in a daily business use in person, the inventory function could help customers trying to find certain movies.

The summary table is used to provide quick updates about movies and their inventory levels in the company as a whole to help determine which movies have lost their popularity, or just to narrow a search down by the length of the movie.

A6. This report should be run and refreshed weekly to see accurate data, as well as whenever new stock is being ordered or considered. This will help the stakeholders make the decisions they need based on the data in the refresh. I believe that it should be refreshed at least hourly for this business purpose.

-- Section B starts here--

--This is going to create the table that will house the detailed information --

CREATE TABLE IF NOT EXISTS detailed\_report(

film\_id INT ,

title varchar(45),

rental\_rate varchar(45),

rental\_duration INT,

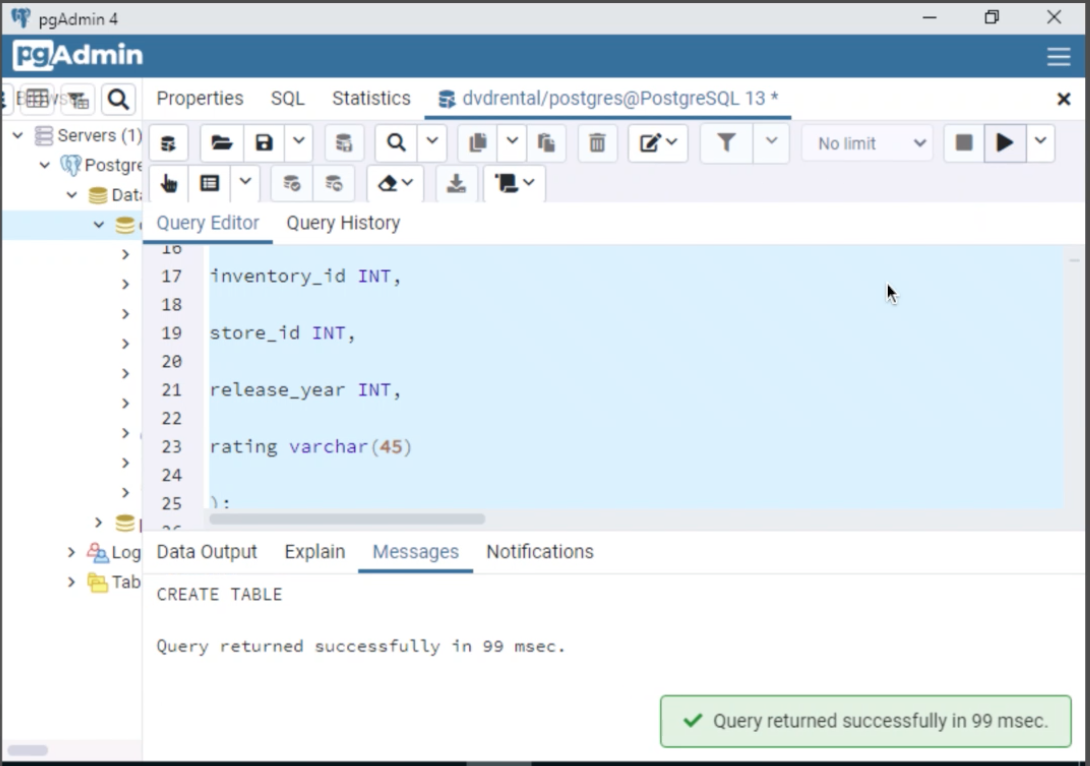
inventory\_id INT,

store\_id INT,

release\_year INT,

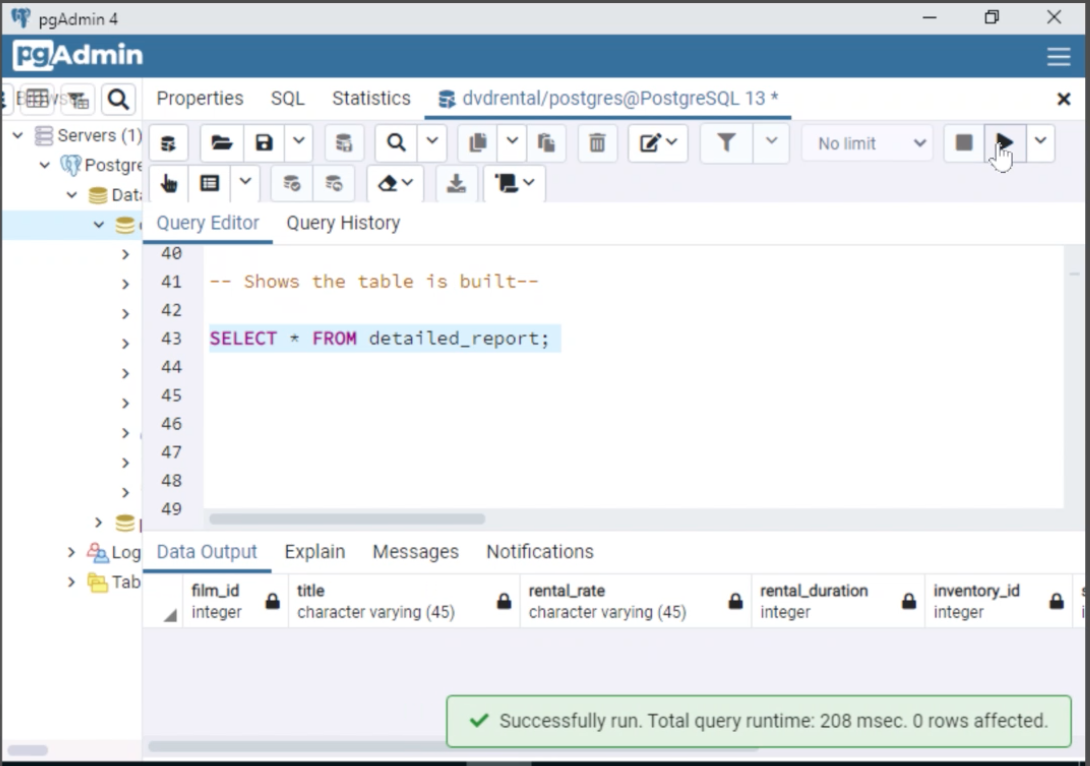
rating varchar(45)

);



-- Shows the table is built--

SELECT \* FROM detailed\_report;



--Creating the empty table that would house the Summary table --

CREATE TABLE IF NOT EXISTS summary\_table(

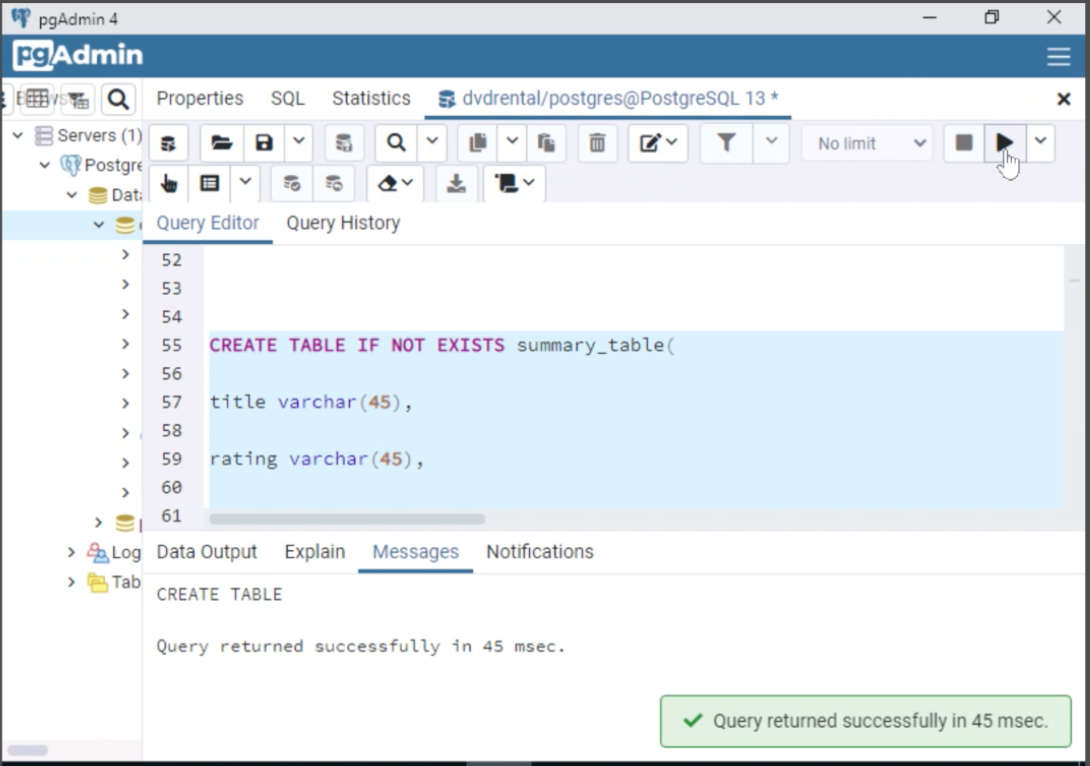
title varchar(45),

rating varchar(45),

release\_year INT,

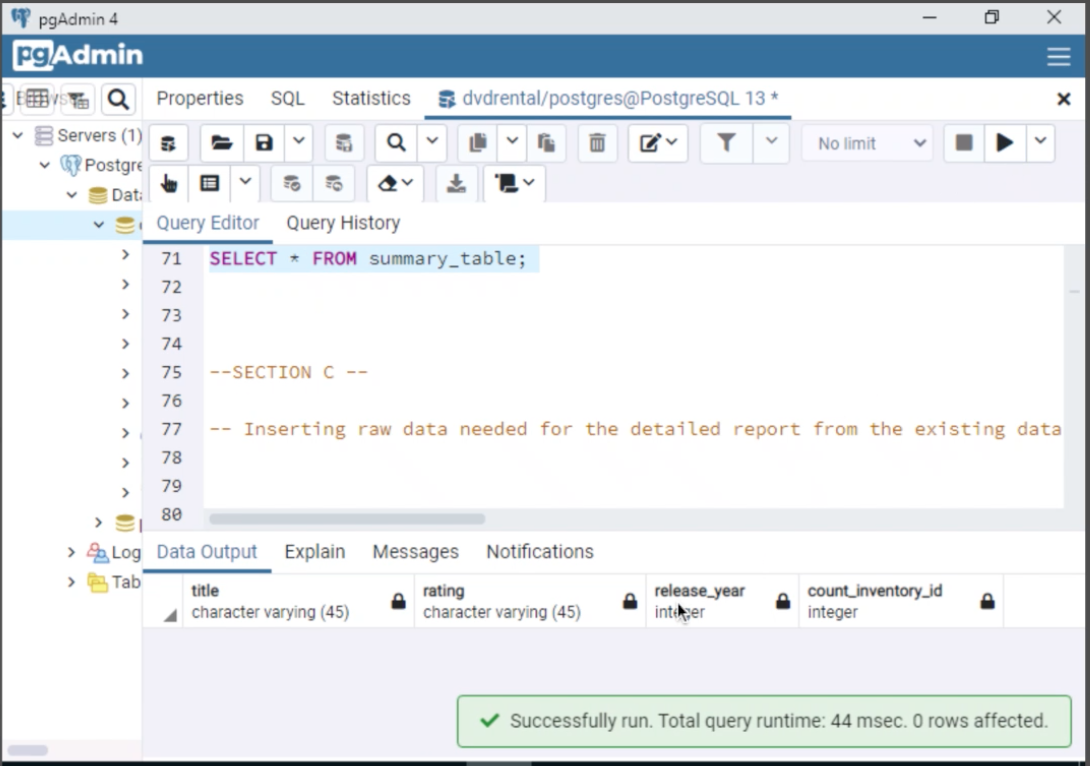
count\_inventory\_id INT

);



--Viewing the Summary Table --

SELECT \* FROM summary\_table;



--SECTION C --

-- Inserting raw data needed for the detailed report from the existing database--

INSERT INTO detailed\_report(

film\_id,

title,

rental\_rate,

rental\_duration,

inventory\_id,

store\_id ,

release\_year

, rating)

SELECT film.film\_id,

title,

rental\_rate

,rental\_duration

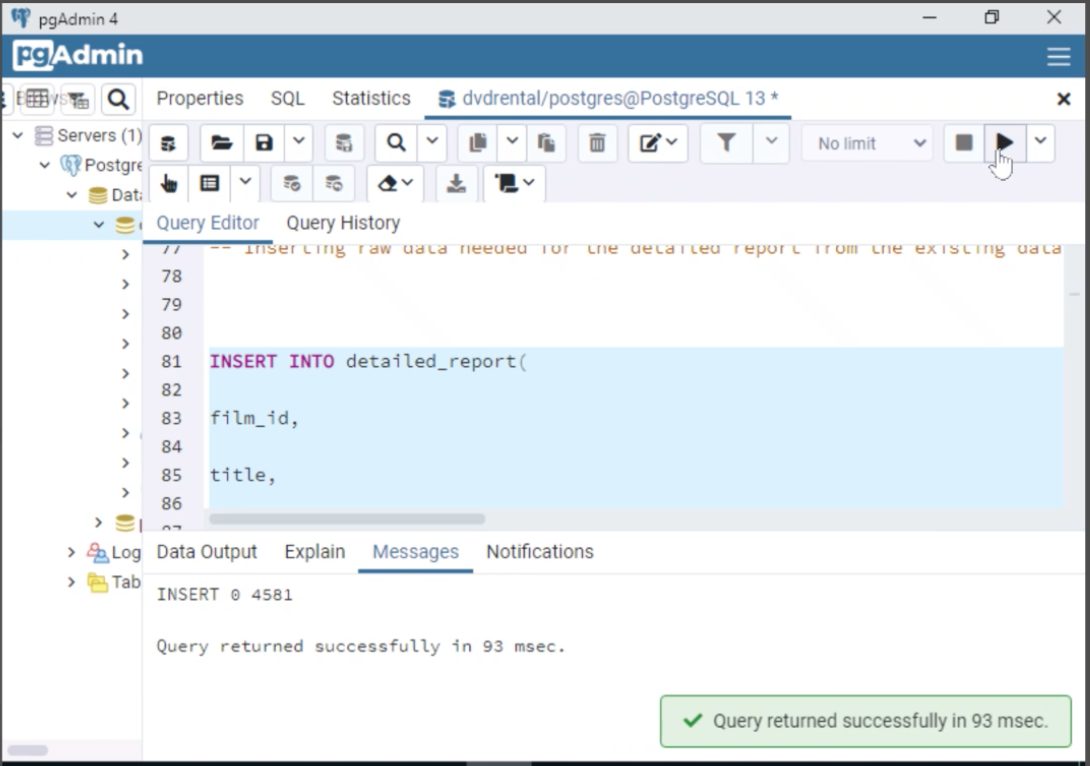
,inventory.inventory\_id

,inventory.store\_id

,release\_year,

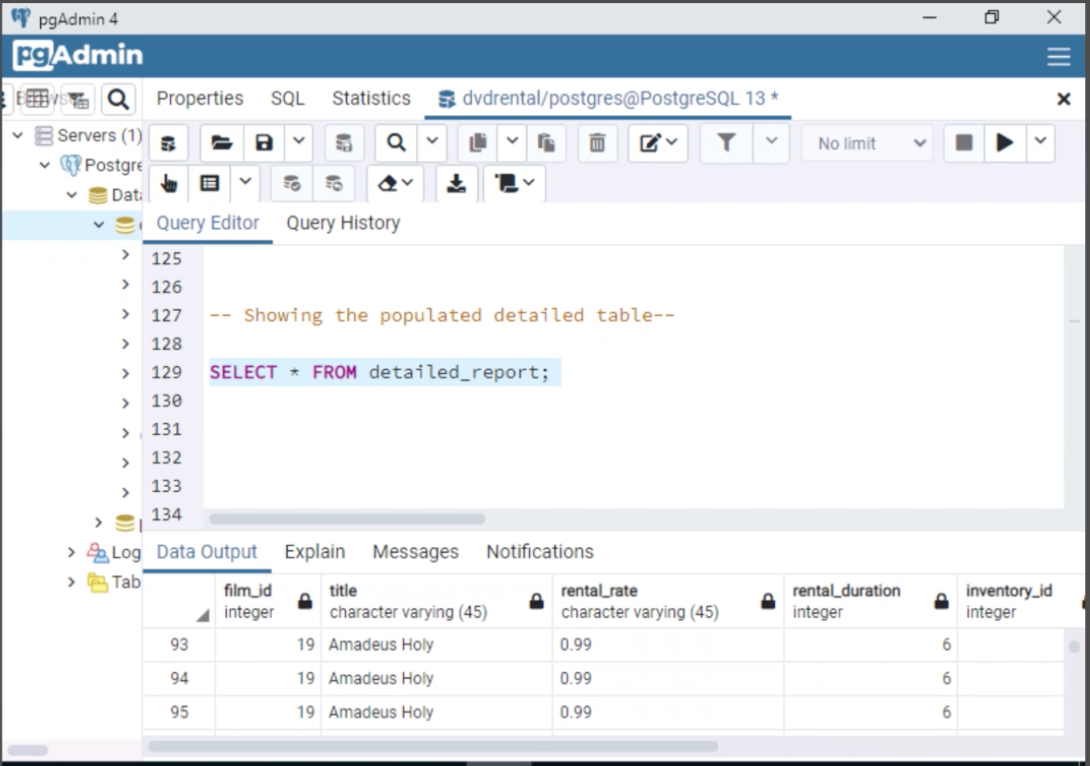
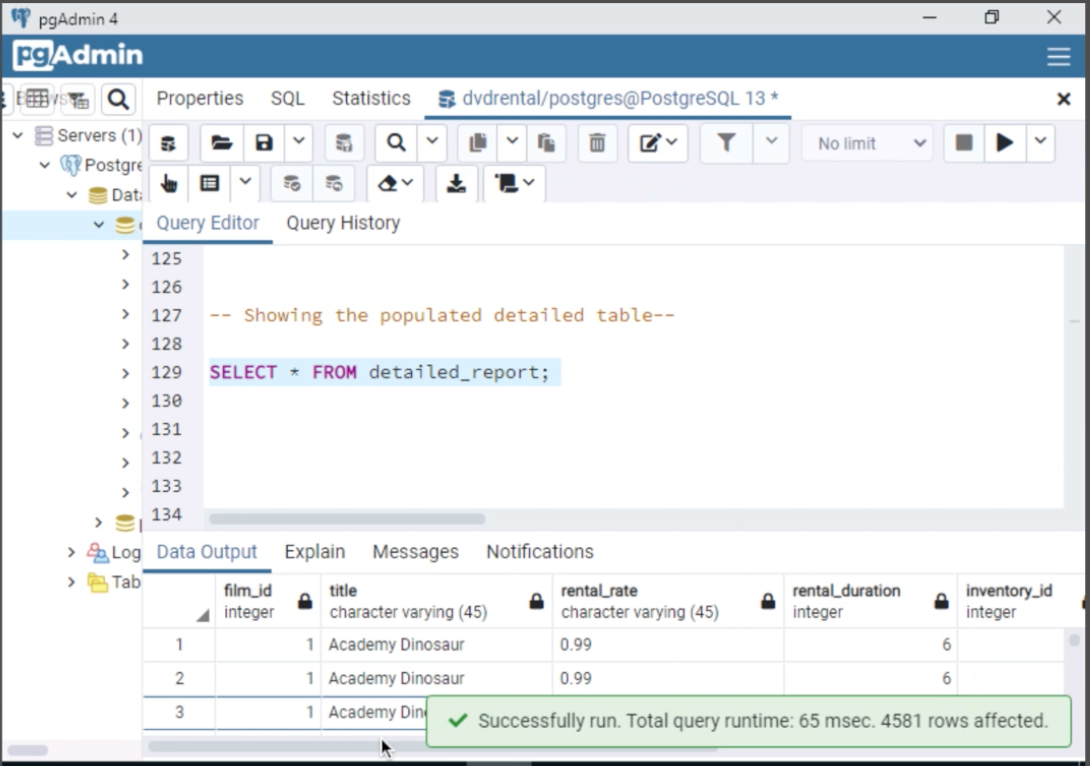
rating

from film join inventory on inventory.film\_id = film.film\_id;



-- Showing the populated detailed table--

SELECT \* FROM detailed\_report;



--Populating the Summary Table

INSERT INTO summary\_table(

title,

rating,

release\_year,

count\_inventory\_id)

SELECT

title,

rating,

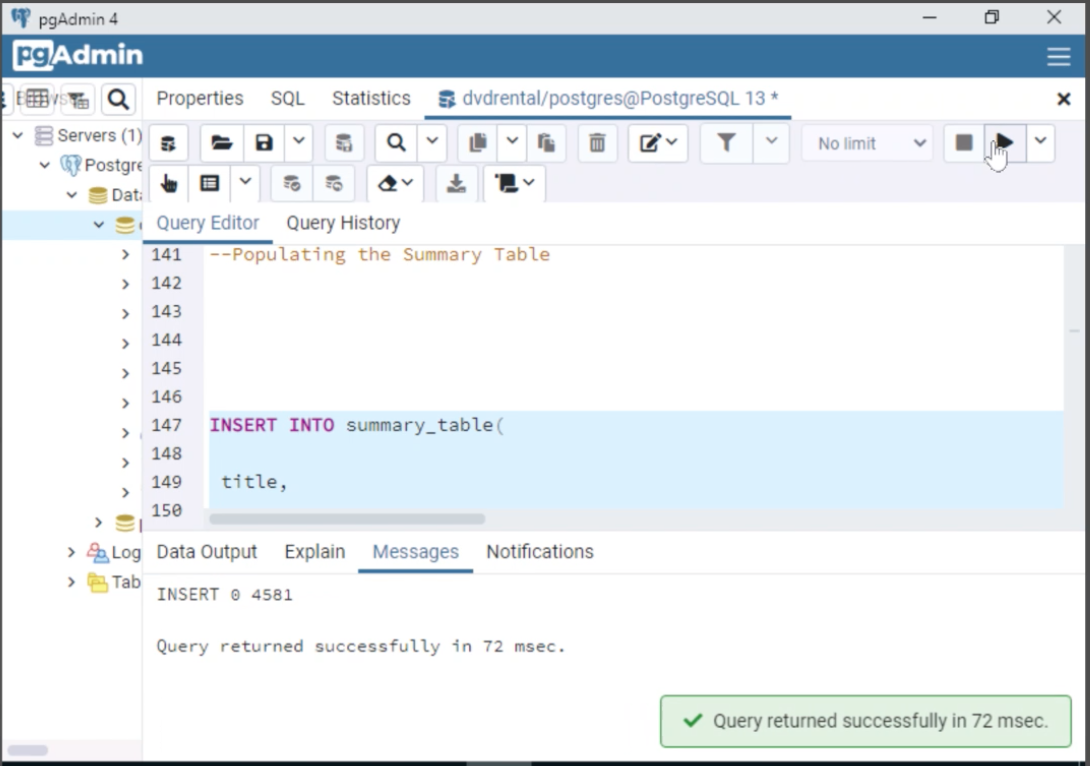
release\_year,

COUNT(inventory\_id )

FROM detailed\_report

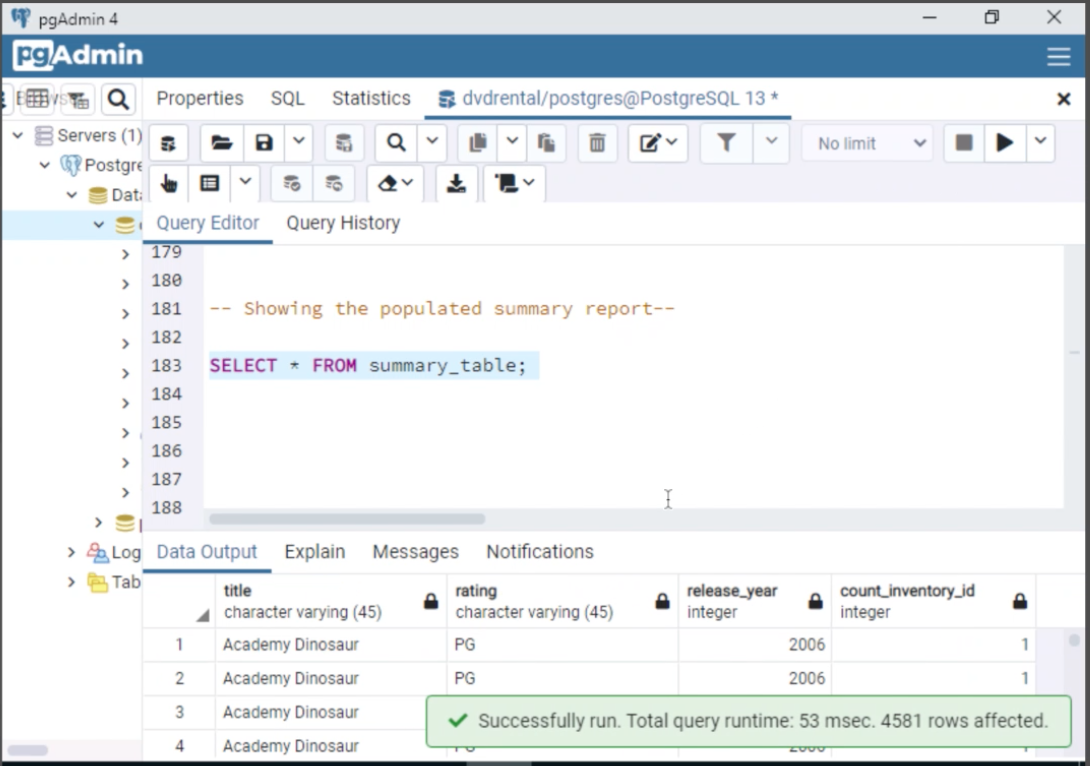
GROUP BY title, rating, release\_year,inventory\_id

ORDER BY release\_year DESC;



-- Showing the populated summary report--

SELECT \* FROM summary\_table;

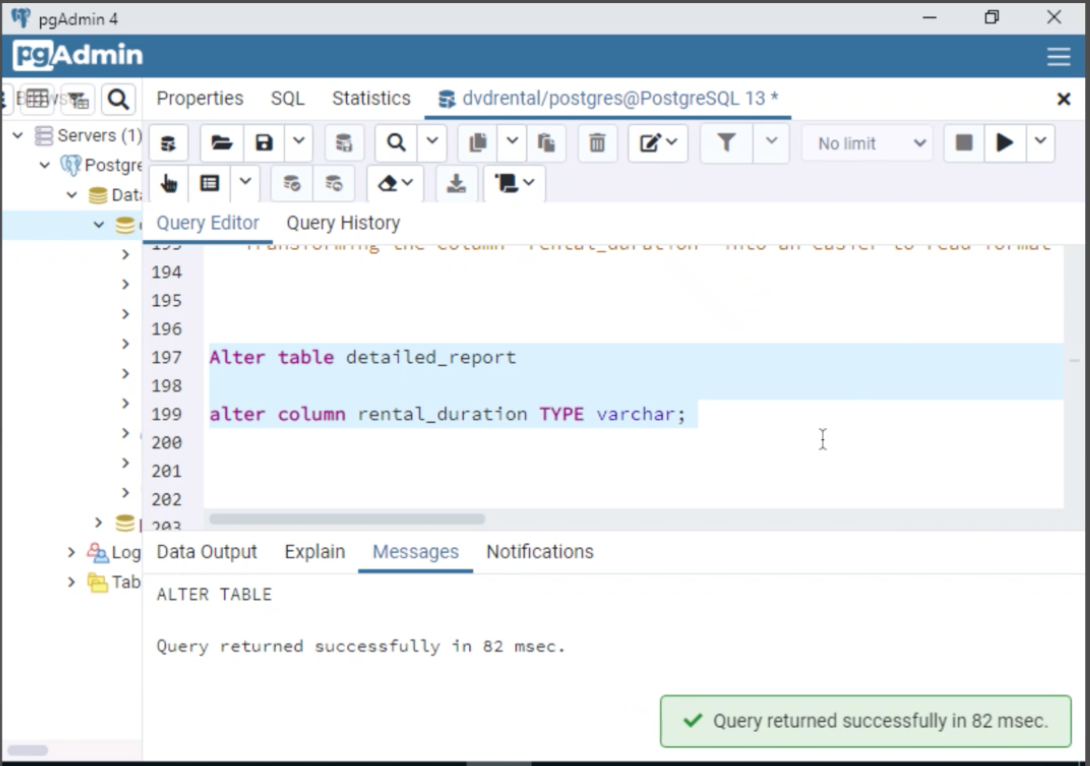


--SECTION D--

-- Transforming the column "rental\_duration” into an easier to read format showing whether the title is kept for a longer duration rental or not, compared to other movies.

Alter table detailed\_report

alter column rental\_duration TYPE varchar;



CREATE FUNCTION transformation\_complete()

RETURNS TRIGGER AS $$ --TRANSFORM\_COMPLETE$--

BEGIN

UPDATE detailed\_report

SET rental\_duration = 'low rental'

WHERE rental\_duration BETWEEN ‘1’ AND ‘3’;

UPDATE detailed\_report

SET rental\_duration = 'medium rental’

WHERE rental\_duration between ‘4’ AND ‘6’;

UPDATE detailed\_report

SET rental\_duration = 'high rental'

WHERE rental\_duration BETWEEN ‘7’ AND ‘9’;

Return New;

END;

$$ --$TRANSFORM\_COMPLETE$--

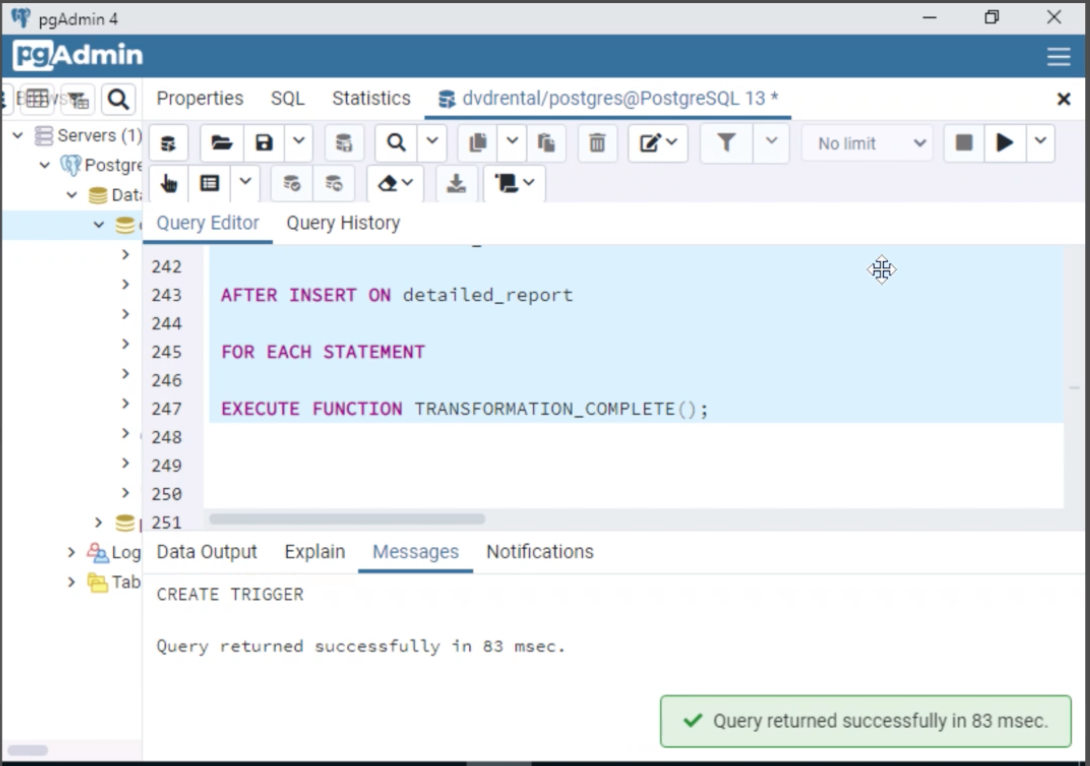
LANGUAGE plpgsql;

CREATE TRIGGER DETAILED\_REFRESH

AFTER INSERT ON detailed\_report

FOR EACH STATEMENT

EXECUTE FUNCTION TRANSFORMATION\_COMPLETE();



--SECTION E--

--Create the summary table refresh function which will refresh the summary table’s data with the data input into the detailed report--

CREATE FUNCTION REFRESH\_SUMMARY\_REPORT()

RETURNS TRIGGER AS $$ --REFRESH\_SUMMARY\_REPORTS$--

BEGIN

DELETE FROM summary\_table;

INSERT INTO summary\_table

(title, rating, release\_year, count\_inventory\_id)

SELECT

title, rating, release\_year, COUNT(inventory\_id)

FROM detailed\_report

GROUP BY title, rating, release\_year, inventory\_id

ORDER BY release\_year DESC;

RETURN NEW;

END;

$$ --$REFRESH\_SUMMARY\_REPORTS$--

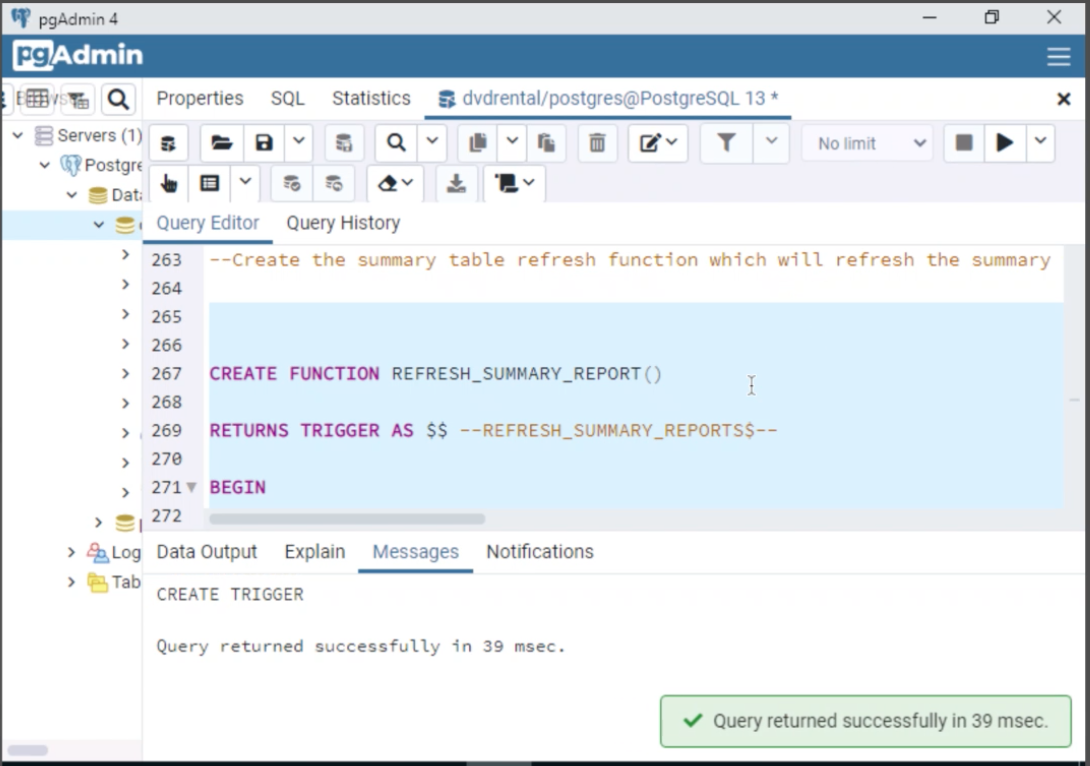
LANGUAGE plpgsql;

CREATE TRIGGER SUMMARY\_UPDATER

AFTER INSERT ON detailed\_report

FOR EACH STATEMENT

EXECUTE FUNCTION REFRESH\_SUMMARY\_REPORT();



-- This section will refresh the summary table and Detailed report whenever there is an added entry-

--Section F --

--SECTION F1 – This could be set up to refresh once a week or nightly or even hourly depending on the need from the company or individual stores. This would be accomplished via the Agent pgAgent job scheduling tool that is involved with Postgres.

CREATE PROCEDURE REFRESH\_BOTH\_TABLES() AS $REFRESH\_BOTH\_TABLES$

BEGIN

DELETE FROM detailed\_report;

INSERT INTO detailed\_report(

film\_id,

title,

rental\_rate,

rental\_duration,

inventory\_id,

store\_id ,

release\_year

, rating)

SELECT film.film\_id,

title,

rental\_rate,

rental\_duration,

inventory.inventory\_id,

inventory.store\_id,

release\_year,

rating

from film join inventory on inventory.film\_id = film.film\_id;

DELETE FROM summary\_table;

INSERT INTO summary\_table

(title, rating, release\_year, count\_inventory\_id)

SELECT

title, rating, release\_year, COUNT(inventory\_id)

FROM detailed\_report

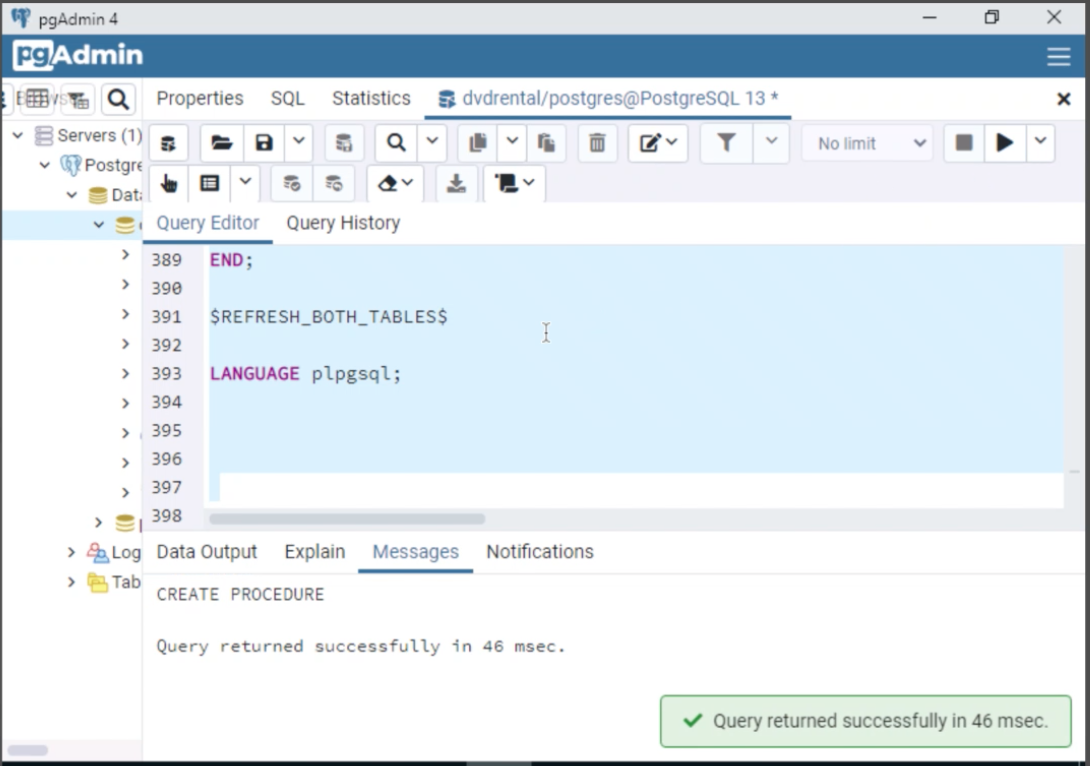
GROUP BY title, rating, release\_year, inventory\_id

ORDER BY release\_year DESC;

END;

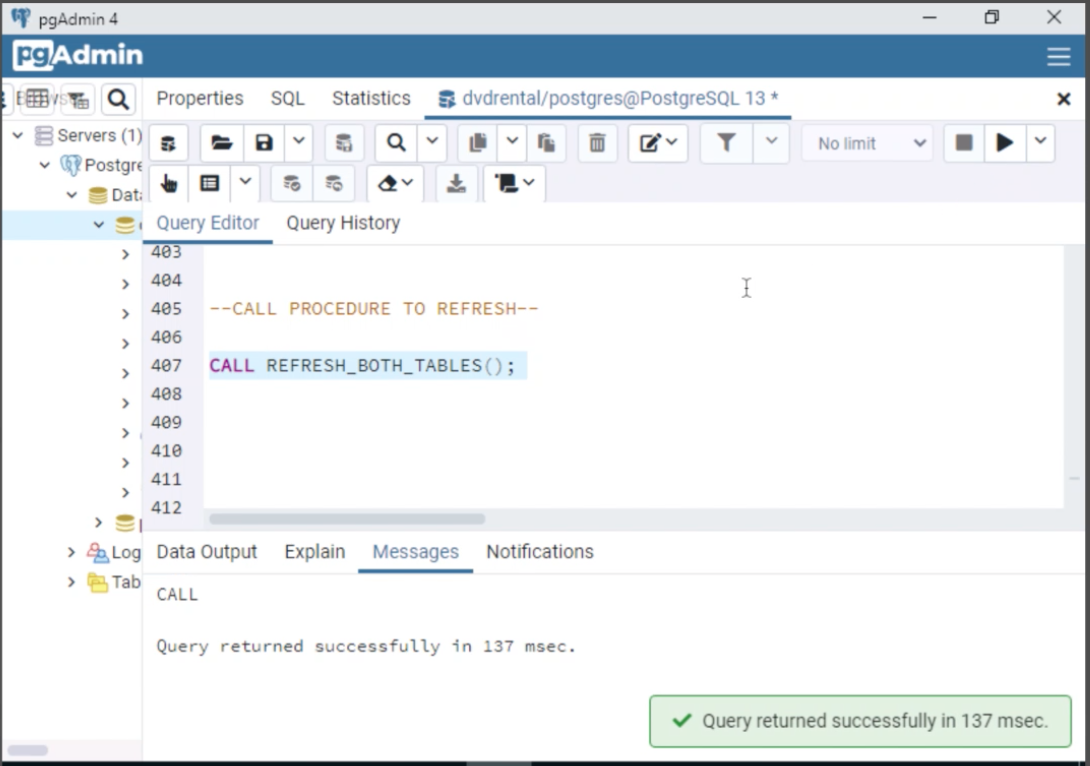
$REFRESH\_BOTH\_TABLES$

LANGUAGE plpgsql;



--CALL PROCEDURE TO REFRESH--

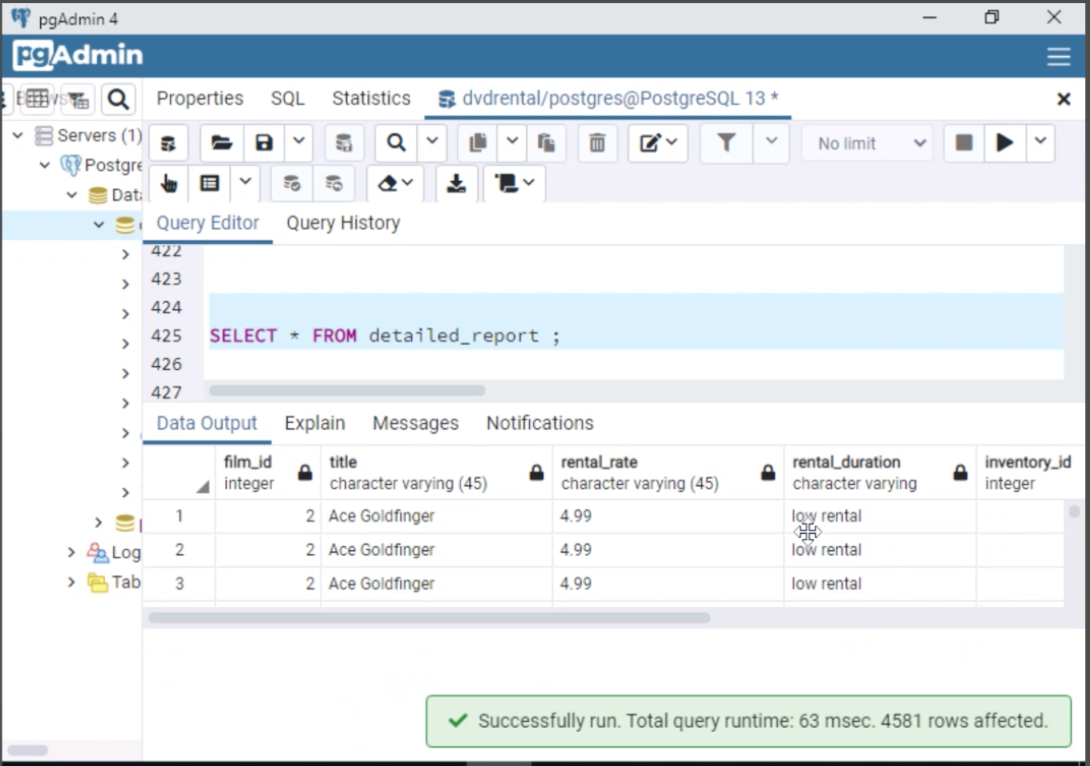
CALL REFRESH\_BOTH\_TABLES();



--This trigger will execute the function that I just created summary\_table\_refresh()

-- when there is an insert into the detailed function.

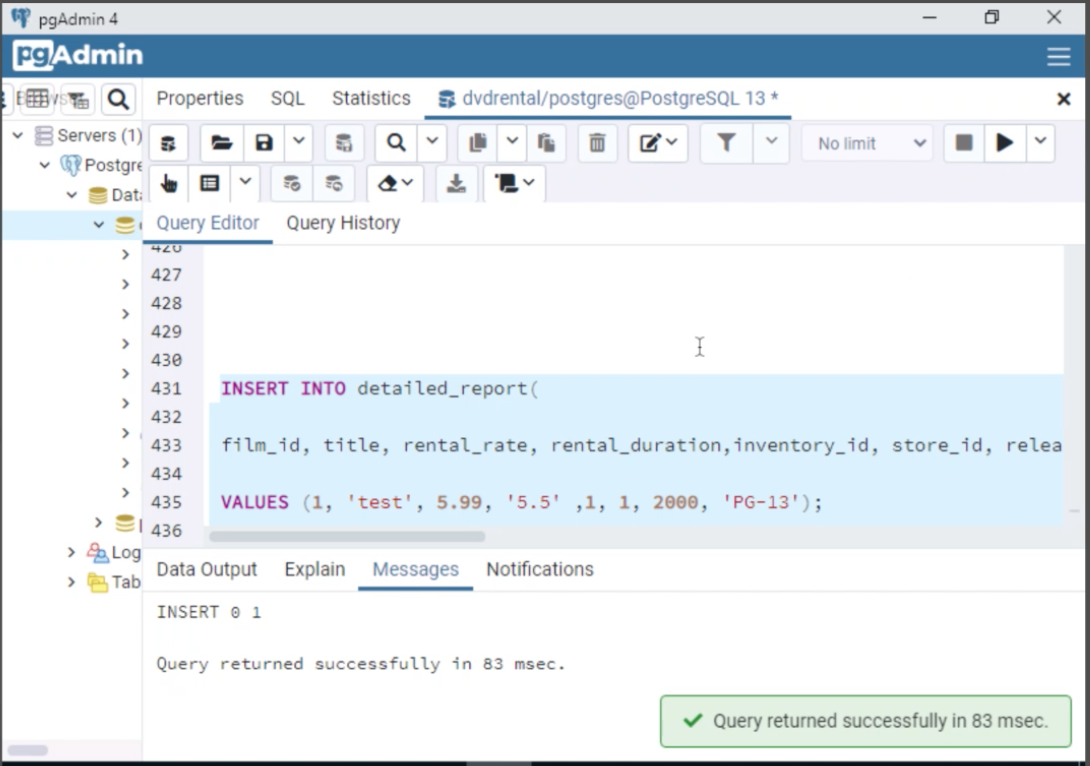
SELECT \* FROM detailed\_report;

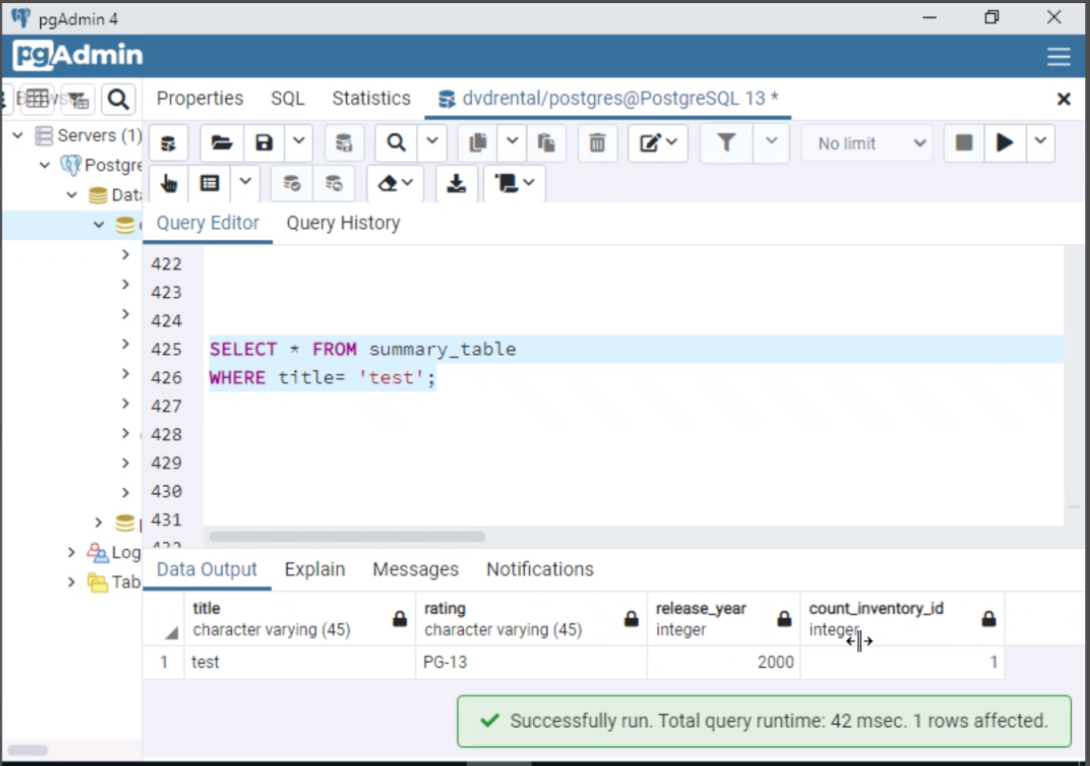
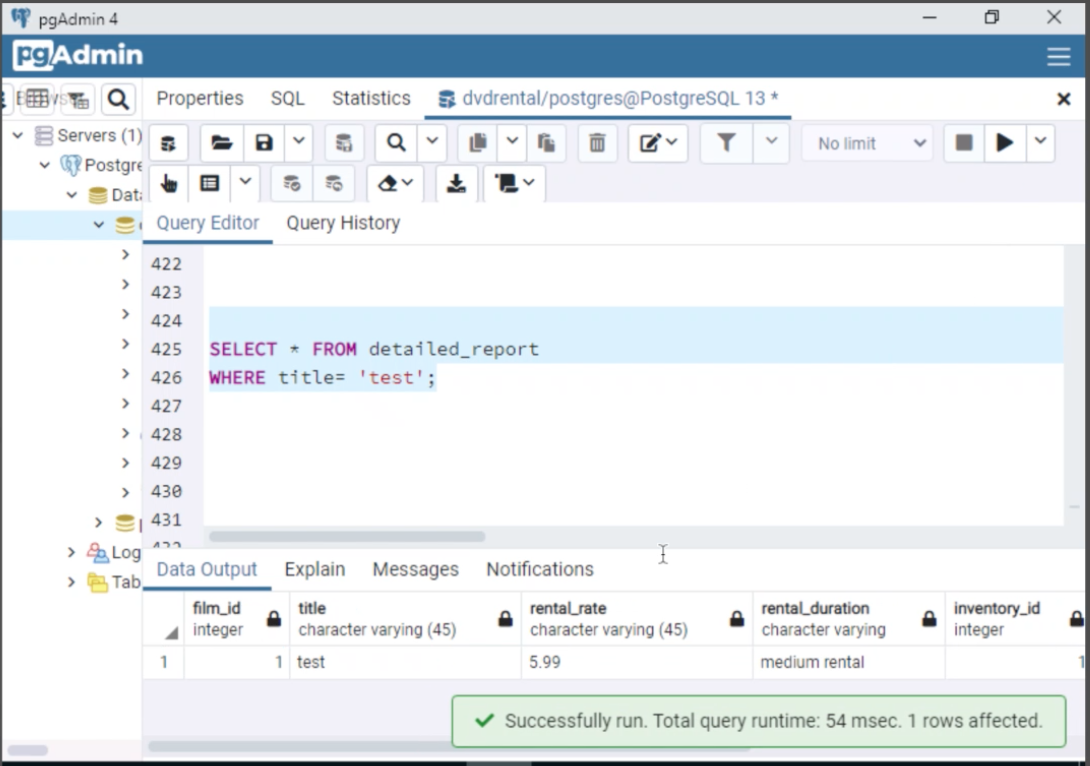


INSERT INTO detailed\_report(

film\_id, title, rental\_rate, rental\_duration,inventory\_id, store\_id, release\_year, rating)

VALUES (1, 'test', 5.99, '5.5' ,1, 1, 2000, 'PG-13');





--SECTION H--

Web Citations

*PostgreSQL Tutorials*. PostgreSQL tutorial. (2022). <https://www.postgresqltutorial.com/>

--SECTION I--

Sources

I did not use any outside sources for this business report.