# DATABASES LAB - 04

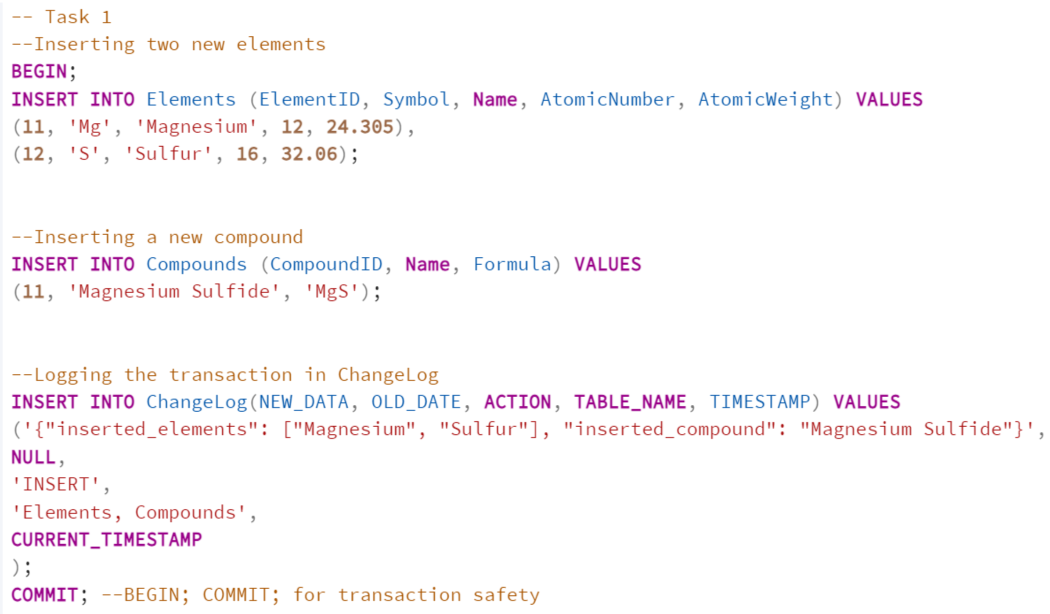
Group 2, Team 12

Thilakraj Soundararajan – 2705370
Mir Md Redwon Sagar – 2613747

Date – 28.06.2025

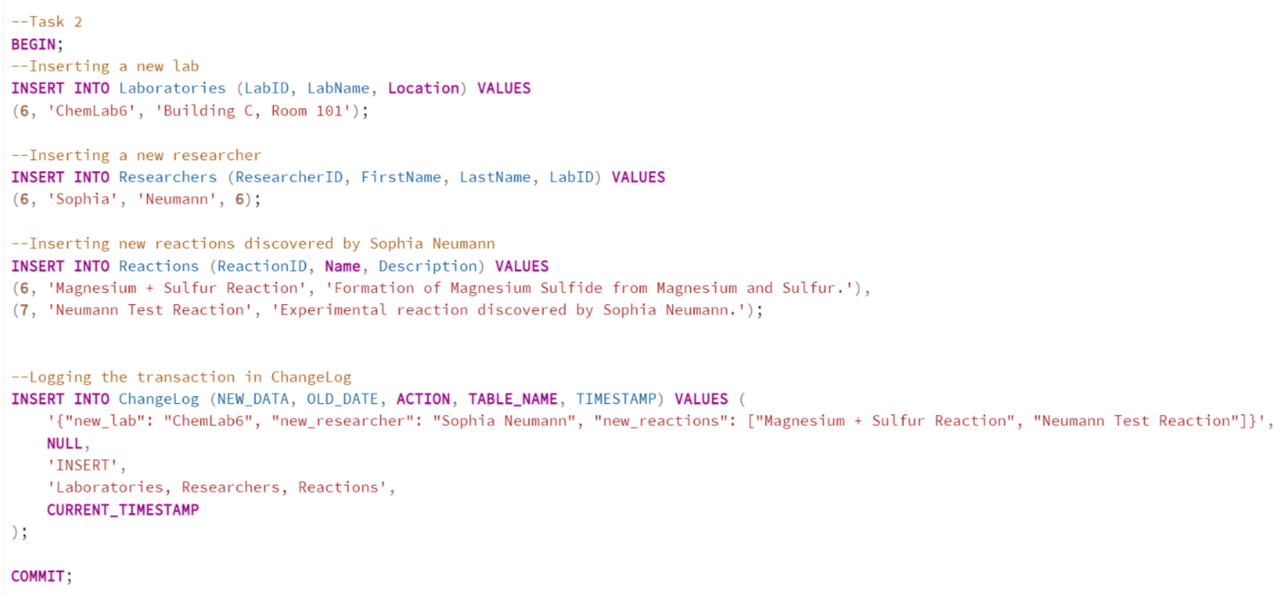
## Assignment 1: Chemistry Database

1. Write a transaction to insert two new elements in table ELEMENTS and a new connection in table COMPOUNDS into the database. If an error occurs during one of the insertions, all changes should be reversed. In addition, the transaction should be logged in table CHANGELOG.



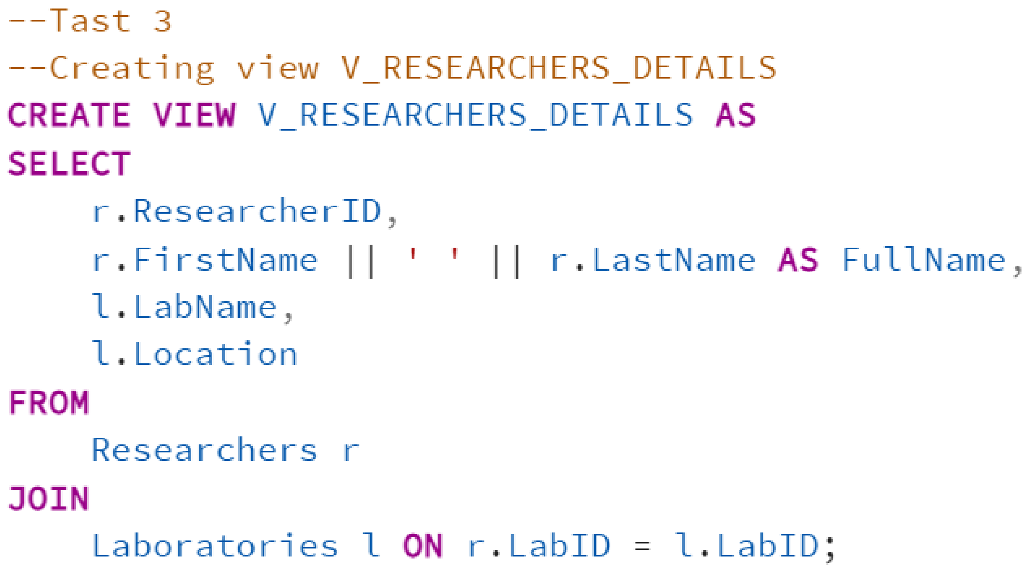
```
-- Task 1
--Inserting two new elements
BEGIN;
INSERT INTO Elements (ElementID, Symbol, Name, AtomicNumber, AtomicWeight) VALUES
(11, 'Mg', 'Magnesium', 12, 24.305),
(12, 'S', 'Sulfur', 16, 32.06);


--Inserting a new compound
INSERT INTO Compounds (CompoundID, Name, Formula) VALUES
(11, 'Magnesium Sulfide', 'MgS');


--Logging the transaction in ChangeLog
INSERT INTO ChangeLog(NEW_DATA, OLD_DATE, ACTION, TABLE_NAME, TIMESTAMP) VALUES
('{"inserted_elements": ["Magnesium", "Sulfur"], "inserted_compound": "Magnesium Sulfide"}',
NULL,
'INSERT',
'Elements, Compounds',
CURRENT_TIMESTAMP
);
COMMIT; --BEGIN; COMMIT; for transaction safety
```

2. Write a transaction to add • a new lab (ChemLab6 located at Building C, Room 101) into table LABORATORIES, • a new researcher (Sophia Neumann) assigned to this lab into table RESEARCHERS, and • two new reactions discovered by this new researcher into table REACTIONS.



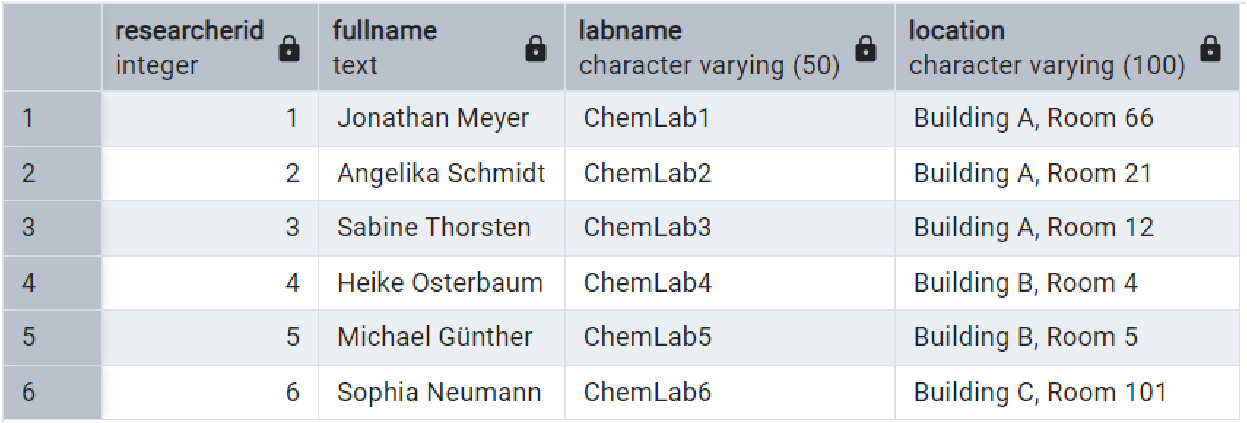
```
--Task 2
BEGIN;
--Inserting a new lab
INSERT INTO Laboratories (LabID, LabName, Location) VALUES
(6, 'ChemLab6', 'Building C, Room 101');

--Inserting a new researcher
INSERT INTO Researchers (ResearcherID, FirstName, LastName, LabID) VALUES
(6, 'Sophia', 'Neumann', 6);

--Inserting new reactions discovered by Sophia Neumann
INSERT INTO Reactions (ReactionID, Name, Description) VALUES
(6, 'Magnesium + Sulfur Reaction', 'Formation of Magnesium Sulfide from Magnesium and Sulfur.'),
(7, 'Neumann Test Reaction', 'Experimental reaction discovered by Sophia Neumann.');


--Logging the transaction in ChangeLog
INSERT INTO ChangeLog (NEW_DATA, OLD_DATE, ACTION, TABLE_NAME, TIMESTAMP) VALUES (
    '{"new_lab": "ChemLab6", "new_researcher": "Sophia Neumann", "new_reactions": ["Magnesium + Sulfur Reaction", "Neumann Test Reaction"]}',
    NULL,
    'INSERT',
    'Laboratories, Researchers, Reactions',
    CURRENT_TIMESTAMP
);

COMMIT;
```

3. Create the view V_RESEARCHERS_DETAILS to display information about all researchers in detail, including their full name, the name of their lab and the location of the lab. The view should combine data from the tables RESEARCHERS and LABS.



```
--Tast 3
--Creating view V_RESEARCHERS_DETAILS
CREATE VIEW V_RESEARCHERS_DETAILS AS
SELECT
    r.ResearcherID,
    r.FirstName || ' ' || r.LastName AS FullName,
    l.LabName,
    l.Location
FROM
    Researchers r
JOIN
    Laboratories l ON r.LabID = l.LabID;
```

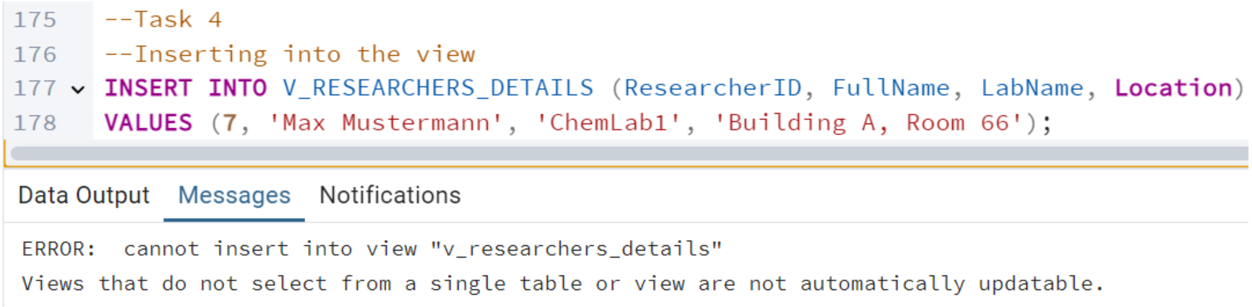
**Table:**



| | researcherid<br>integer | fullname<br>text | labname<br>character varying (50) | location<br>character varying (100) |
|---|---|---|---|---|
| 1 | 1 | Jonathan Meyer | ChemLab1 | Building A, Room 66 |
| 2 | 2 | Angelika Schmidt | ChemLab2 | Building A, Room 21 |
| 3 | 3 | Sabine Thorsten | ChemLab3 | Building A, Room 12 |
| 4 | 4 | Heike Osterbaum | ChemLab4 | Building B, Room 4 |
| 5 | 5 | Michael Günther | ChemLab5 | Building B, Room 5 |
| 6 | 6 | Sophia Neumann | ChemLab6 | Building C, Room 101 |

4. Try to insert, delete, and update tuples in the view V_RESEARCHERS_DETAILS. Which operations (INSERT, DELETE, and UPDATE) can be executed and which not? Explain your answer.

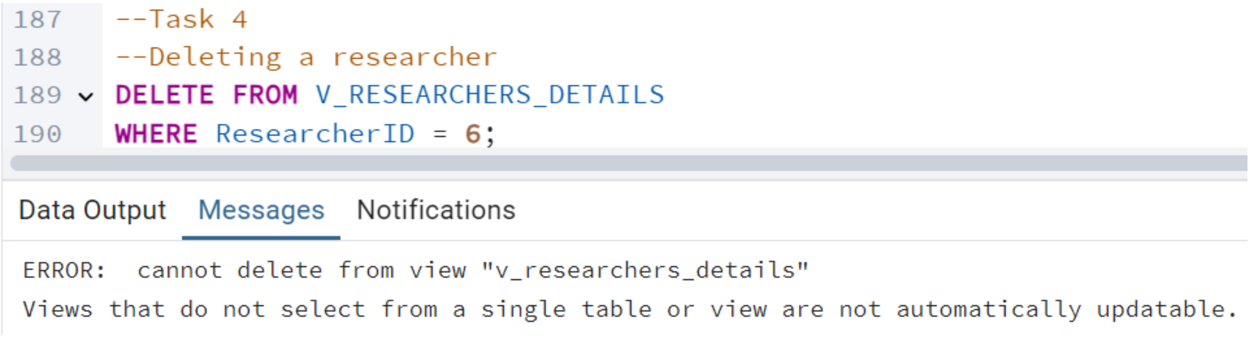
**Insert:**



```
--Task 4
--Inserting into the view
INSERT INTO V_RESEARCHERS_DETAILS (ResearcherID, FullName, LabName, Location)
VALUES (7, 'Max Mustermann', 'ChemLab1', 'Building A, Room 66');
```

```
ERROR:  cannot insert into view "v_researchers_details"
Views that do not select from a single table or view are not automatically updatable.
```

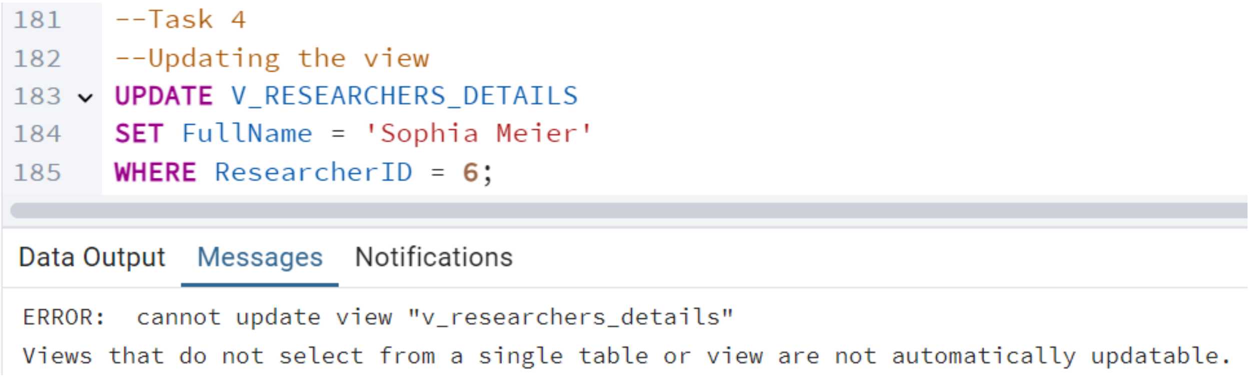
**Delete:**



```
--Task 4
--Deleting a researcher
DELETE FROM V_RESEARCHERS_DETAILS
WHERE ResearcherID = 6;
```

```
ERROR:  cannot delete from view "v_researchers_details"
Views that do not select from a single table or view are not automatically updatable.
```

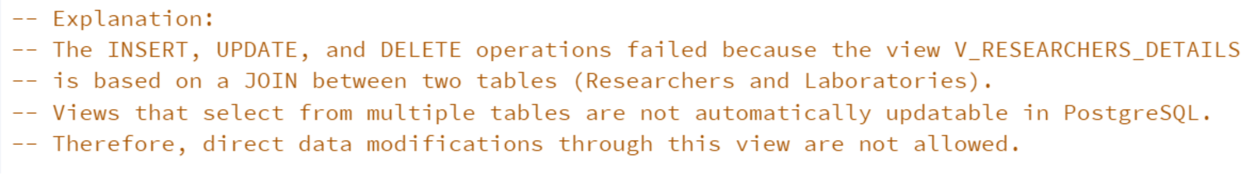
**Update:**



```
--Task 4
--Updating the view
UPDATE V_RESEARCHERS_DETAILS
SET FullName = 'Sophia Meier'
WHERE ResearcherID = 6;
```

```
ERROR:  cannot update view "v_researchers_details"
Views that do not select from a single table or view are not automatically updatable.
```

**Explain:**



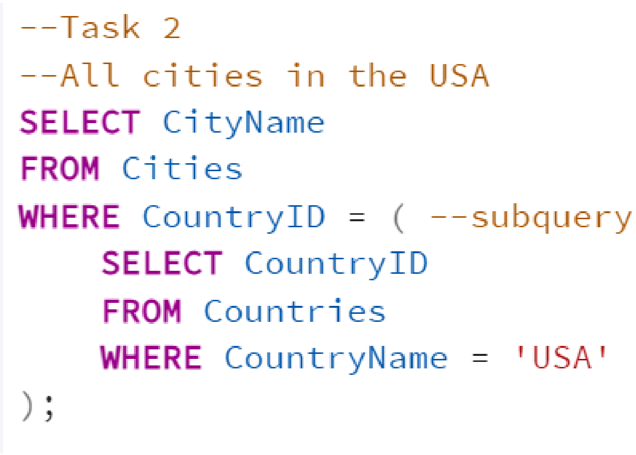
```
-- Explanation:
-- The INSERT, UPDATE, and DELETE operations failed because the view V_RESEARCHERS_DETAILS
-- is based on a JOIN between two tables (Researchers and Laboratories).
-- Views that select from multiple tables are not automatically updatable in PostgreSQL.
-- Therefore, direct data modifications through this view are not allowed.
```

## Assignment 2: Geography Database

1. What is the capital of Germany?

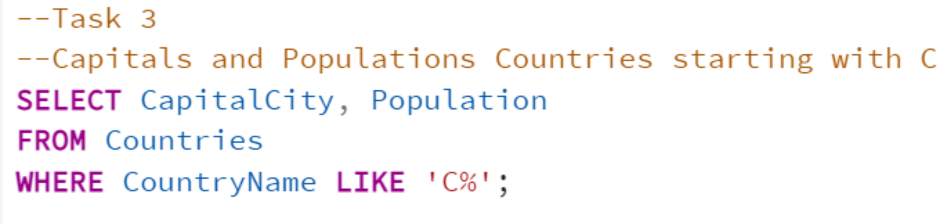
```
--Assignment 2
--Task 1
--Capital City of Germany
SELECT CapitalCity
FROM Countries
WHERE CountryName = 'Germany';
```

2. Write an SQL query to list all cities in the USA.



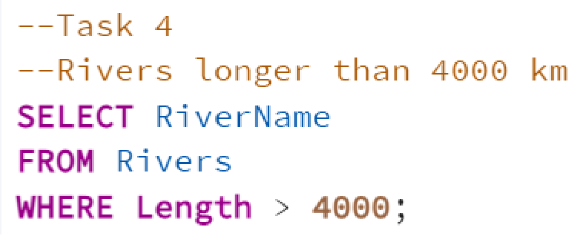
```sql
--Task 2
--All cities in the USA
SELECT CityName
FROM Cities
WHERE CountryID = ( --subquery
    SELECT CountryID
    FROM Countries
    WHERE CountryName = 'USA'
);
```

3. Write an SQL query to find the capitals and populations of all countries with names beginning with the letter “C”.



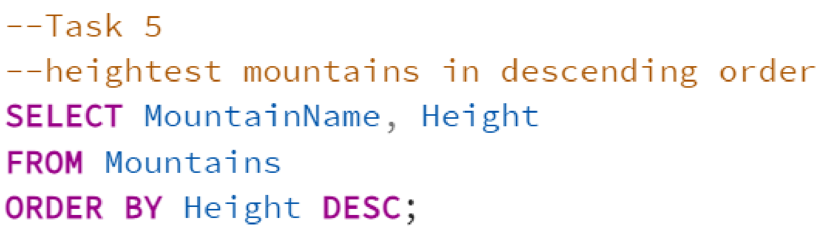
```sql
--Task 3
--Capitals and Populations Countries starting with C
SELECT CapitalCity, Population
FROM Countries
WHERE CountryName LIKE 'C%';
```

4. List all rivers that are longer than 4000 km.



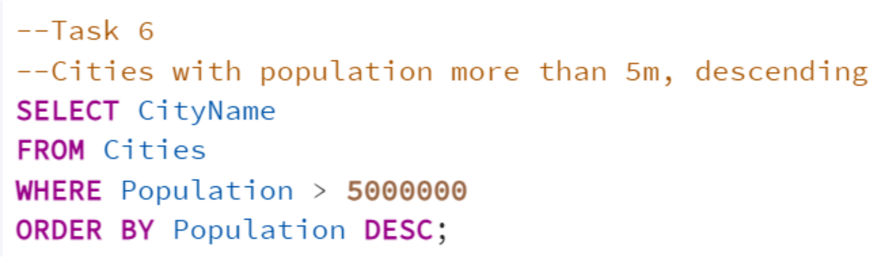
```sql
--Task 4
--Rivers longer than 4000 km
SELECT RiverName
FROM Rivers
WHERE Length > 4000;
```

5. Identify the highest mountains in descending order of height.



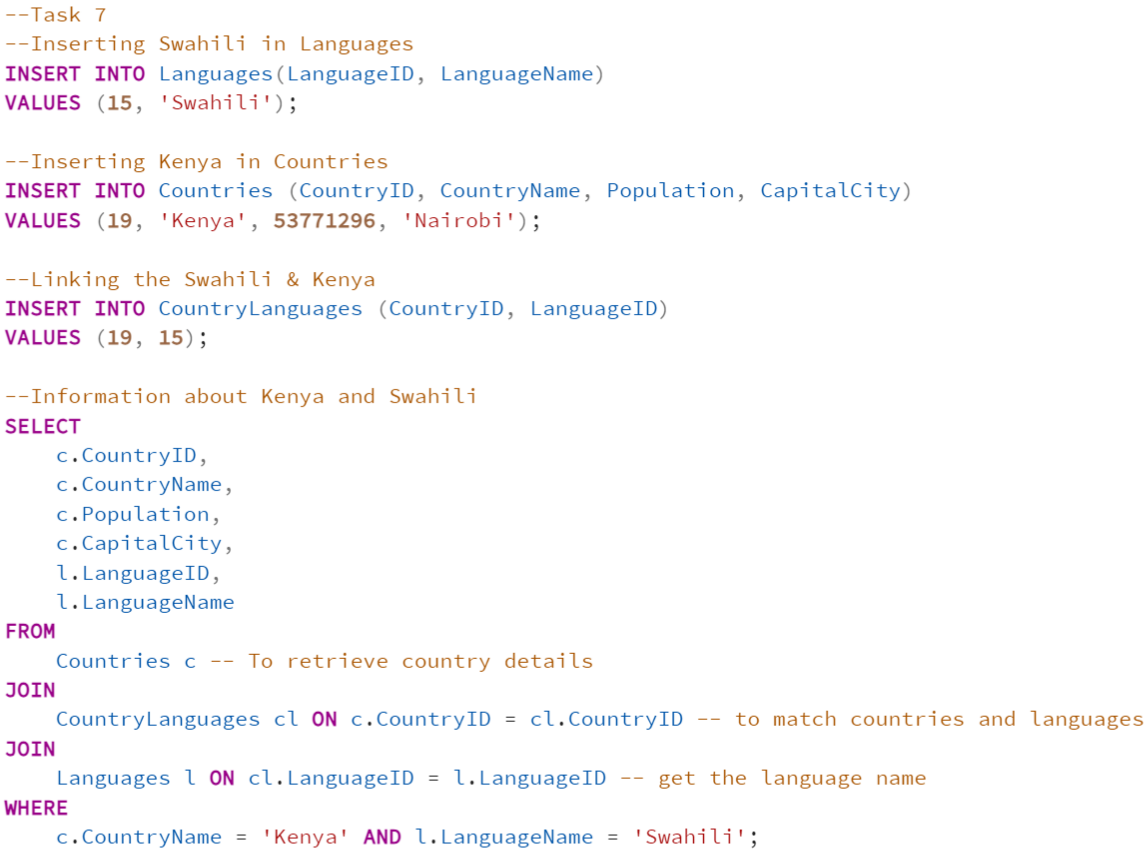
```sql
--Task 5
--heightest mountains in descending order
SELECT MountainName, Height
FROM Mountains
ORDER BY Height DESC;
```

6. List all cities with a population over 5 million in descending order of population.



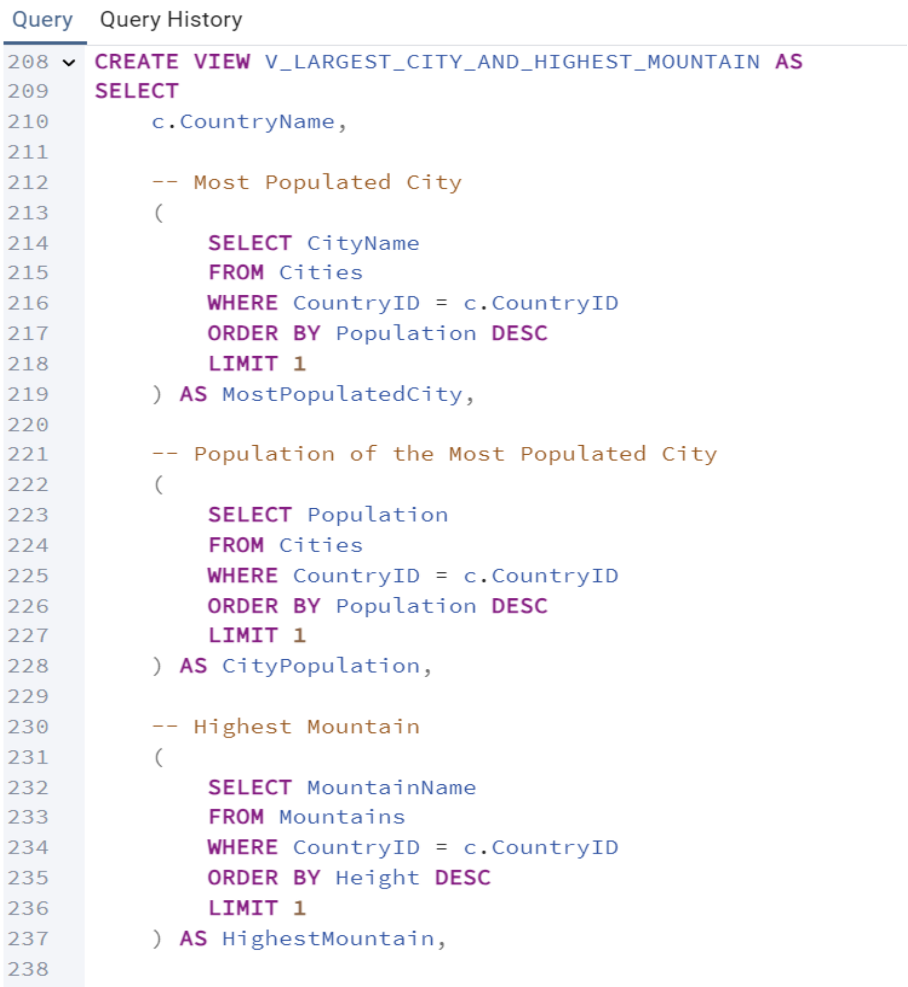
```
--Task 6
--Cities with population more than 5m, descending
SELECT CityName
FROM Cities
WHERE Population > 5000000
ORDER BY Population DESC;
```

7. Add a new language called “Swahili” and a new country “Kenya” with CountryID 19. Then, link the language and the country in table COUNTRYLANGUAGES. Write a query displaying all information about the country “Kenya” and the language “Swahili” that checks the completeness of the data.

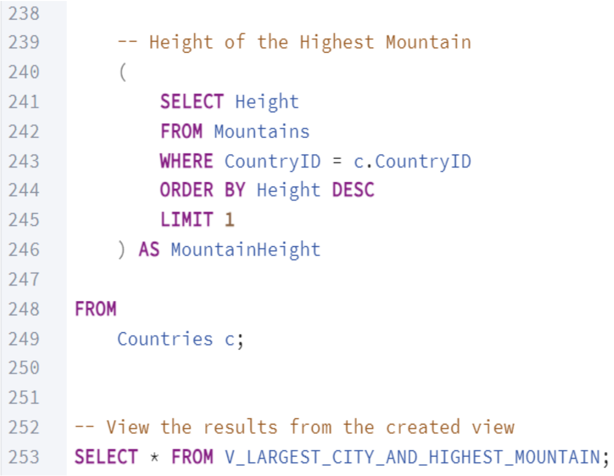


```
--Task 7
--Inserting Swahili in Languages
INSERT INTO Languages(LanguageID, LanguageName)
VALUES (15, 'Swahili');

--Inserting Kenya in Countries
INSERT INTO Countries (CountryID, CountryName, Population, CapitalCity)
VALUES (19, 'Kenya', 53771296, 'Nairobi');

--Linking the Swahili & Kenya
INSERT INTO CountryLanguages (CountryID, LanguageID)
VALUES (19, 15);

--Information about Kenya and Swahili
SELECT
    c.CountryID,
    c.CountryName,
    c.Population,
    c.CapitalCity,
    l.LanguageID,
    l.LanguageName
FROM
    Countries c -- To retrieve country details
JOIN
    CountryLanguages cl ON c.CountryID = cl.CountryID -- to match countries and languages
JOIN
    Languages l ON cl.LanguageID = l.LanguageID -- get the language name
WHERE
    c.CountryName = 'Kenya' AND l.LanguageName = 'Swahili';
```

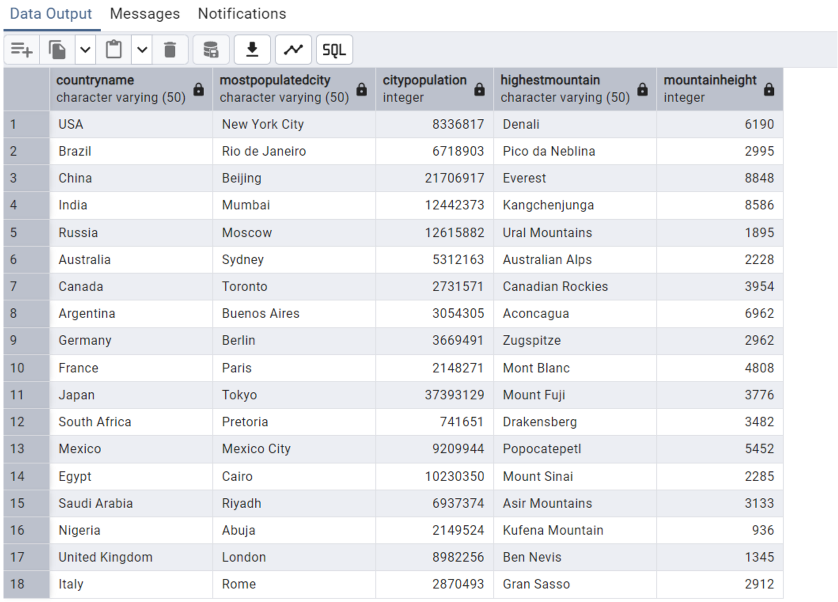
8. Create the view V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN that shows for each country the name of the country, the name of the most populated city, the population of this city, the name of the highest mountain and the height of this mountain.



```sql
CREATE VIEW V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN AS
SELECT
    c.CountryName,

    -- Most Populated City
    (
        SELECT CityName
        FROM Cities
        WHERE CountryID = c.CountryID
        ORDER BY Population DESC
        LIMIT 1
    ) AS MostPopulatedCity,

    -- Population of the Most Populated City
    (
        SELECT Population
        FROM Cities
        WHERE CountryID = c.CountryID
        ORDER BY Population DESC
        LIMIT 1
    ) AS CityPopulation,

    -- Highest Mountain
    (
        SELECT MountainName
        FROM Mountains
        WHERE CountryID = c.CountryID
        ORDER BY Height DESC
        LIMIT 1
    ) AS HighestMountain,

```



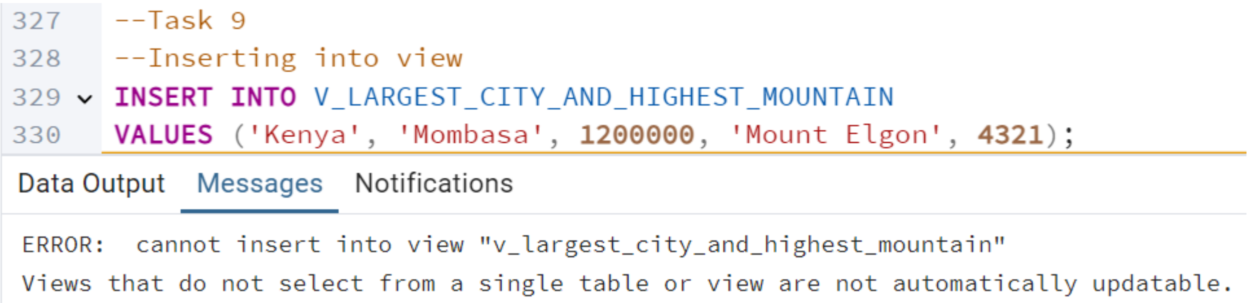
```sql

    -- Height of the Highest Mountain
    (
        SELECT Height
        FROM Mountains
        WHERE CountryID = c.CountryID
        ORDER BY Height DESC
        LIMIT 1
    ) AS MountainHeight

FROM
    Countries c;


-- View the results from the created view
SELECT * FROM V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN;
```



| | countryname<br>character varying (50) | mostpopulatedcity<br>character varying (50) | citypopulation<br>integer | highestmountain<br>character varying (50) | mountainheight<br>integer |
|---|---|---|---|---|---|
| 1 | USA | New York City | 8336817 | Denali | 6190 |
| 2 | Brazil | Rio de Janeiro | 6718903 | Pico da Neblina | 2995 |
| 3 | China | Beijing | 21706917 | Everest | 8848 |
| 4 | India | Mumbai | 12442373 | Kangchenjunga | 8586 |
| 5 | Russia | Moscow | 12615882 | Ural Mountains | 1895 |
| 6 | Australia | Sydney | 5312163 | Australian Alps | 2228 |
| 7 | Canada | Toronto | 2731571 | Canadian Rockies | 3954 |
| 8 | Argentina | Buenos Aires | 3054305 | Aconcagua | 6962 |
| 9 | Germany | Berlin | 3669491 | Zugspitze | 2962 |
| 10 | France | Paris | 2148271 | Mont Blanc | 4808 |
| 11 | Japan | Tokyo | 37393129 | Mount Fuji | 3776 |
| 12 | South Africa | Pretoria | 741651 | Drakensberg | 3482 |
| 13 | Mexico | Mexico City | 9209944 | Popocatepetl | 5452 |
| 14 | Egypt | Cairo | 10230350 | Mount Sinai | 2285 |
| 15 | Saudi Arabia | Riyadh | 6937374 | Asir Mountains | 3133 |
| 16 | Nigeria | Abuja | 2149524 | Kufena Mountain | 936 |
| 17 | United Kingdom | London | 8982256 | Ben Nevis | 1345 |
| 18 | Italy | Rome | 2870493 | Gran Sasso | 2912 |

9. Try to insert, delete, and update tuples in the view V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN. Which operations (INSERT, DELETE, and UPDATE) can be executed and which not? Explain your answer.

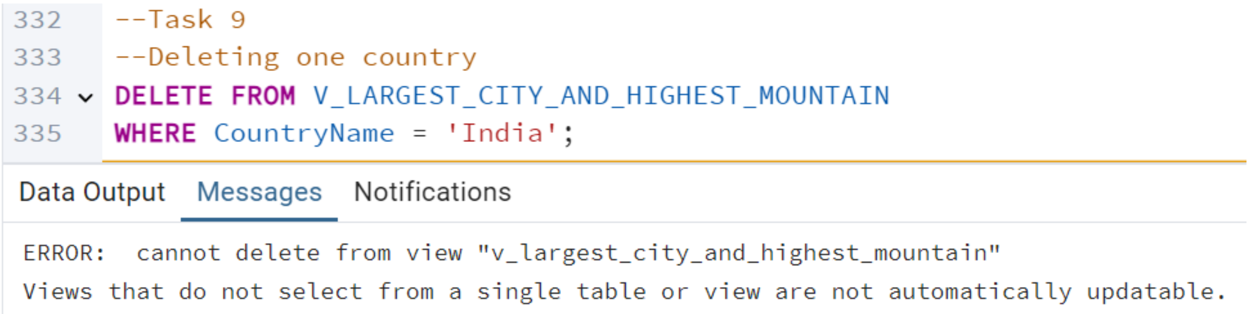
**Insert:**



```
--Task 9
--Inserting into view
INSERT INTO V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN
VALUES ('Kenya', 'Mombasa', 1200000, 'Mount Elgon', 4321);
```

```
ERROR:  cannot insert into view "v_largest_city_and_highest_mountain"
Views that do not select from a single table or view are not automatically updatable.
```

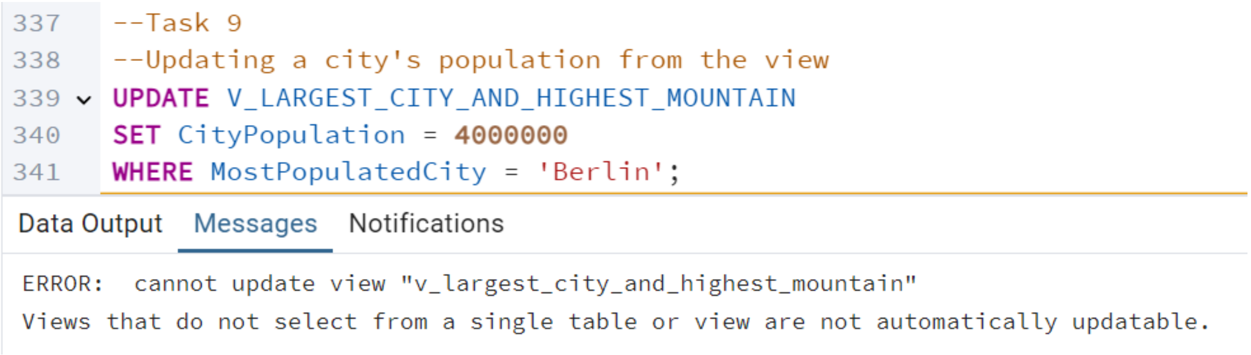
**Delete:**



```
--Task 9
--Deleting one country
DELETE FROM V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN
WHERE CountryName = 'India';
```

```
ERROR:  cannot delete from view "v_largest_city_and_highest_mountain"
Views that do not select from a single table or view are not automatically updatable.
```

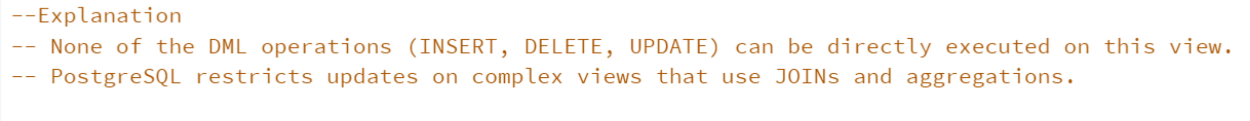
**Update:**



```
--Task 9
--Updating a city's population from the view
UPDATE V_LARGEST_CITY_AND_HIGHEST_MOUNTAIN
SET CityPopulation = 4000000
WHERE MostPopulatedCity = 'Berlin';
```

```
ERROR:  cannot update view "v_largest_city_and_highest_mountain"
Views that do not select from a single table or view are not automatically updatable.
```

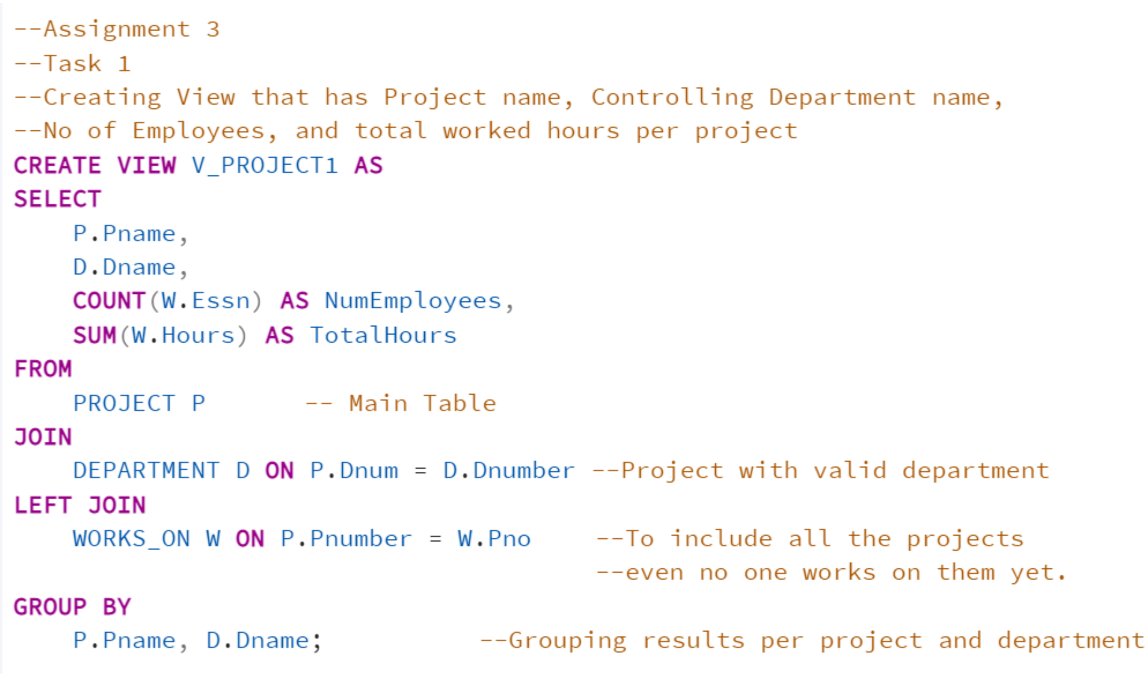
**Explain:**



```
--Explanation
-- None of the DML operations (INSERT, DELETE, UPDATE) can be directly executed on this view.
-- PostgreSQL restricts updates on complex views that use JOINs and aggregations.
```

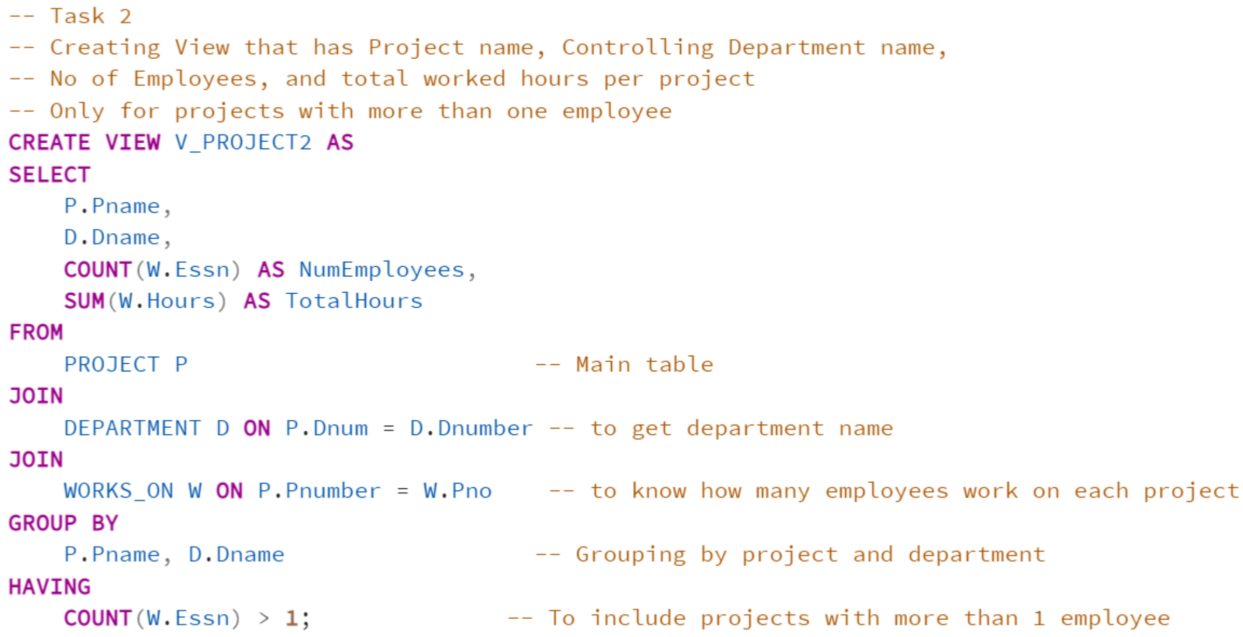
## Assignment 3: SQL-statements for the COMPANY example from Elmasri also used in the lecture

1. Create a SQL statement which creates the view V_PROJECT1. View V_PROJECT1 has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project.



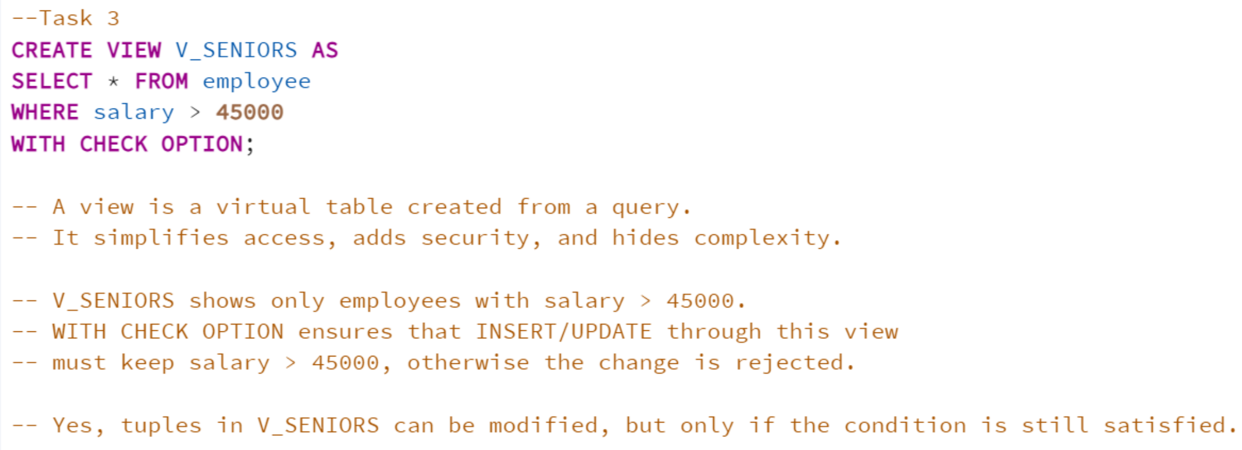
```
--Assignment 3
--Task 1
--Creating View that has Project name, Controlling Department name,
--No of Employees, and total worked hours per project
CREATE VIEW V_PROJECT1 AS
SELECT
    P.Pname,
    D.Dname,
    COUNT(W.Essn) AS NumEmployees,
    SUM(W.Hours) AS TotalHours
FROM
    PROJECT P       -- Main Table
JOIN
    DEPARTMENT D ON P.Dnum = D.Dnumber --Project with valid department
LEFT JOIN
    WORKS_ON W ON P.Pnumber = W.Pno     --To include all the projects
                                        --even no one works on them yet.
GROUP BY
    P.Pname, D.Dname;           --Grouping results per project and department
```

2. Create a SQL statement which creates the view V_PROJECT2. View V_PROJECT2 that has the project name, controlling department name, number of employees, and total hours worked per week on the project for each project with more than one employee working on it.



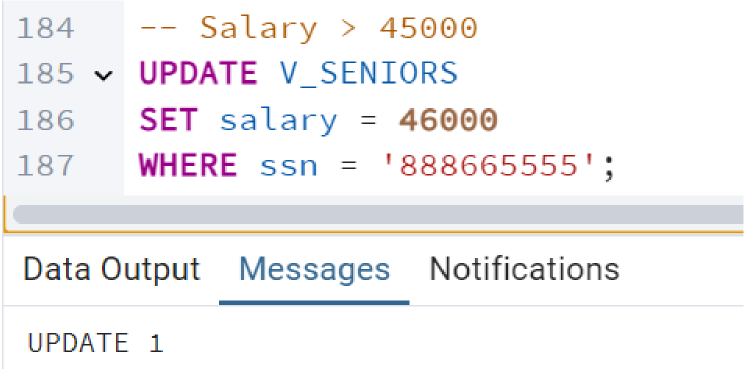
```
-- Task 2
-- Creating View that has Project name, Controlling Department name,
-- No of Employees, and total worked hours per project
-- Only for projects with more than one employee
CREATE VIEW V_PROJECT2 AS
SELECT
    P.Pname,
    D.Dname,
    COUNT(W.Essn) AS NumEmployees,
    SUM(W.Hours) AS TotalHours
FROM
    PROJECT P                          -- Main table
JOIN
    DEPARTMENT D ON P.Dnum = D.Dnumber -- to get department name
JOIN
    WORKS_ON W ON P.Pnumber = W.Pno    -- to know how many employees work on each project
GROUP BY
    P.Pname, D.Dname                   -- Grouping by project and department
HAVING
    COUNT(W.Essn) > 1;               -- To include projects with more than 1 employee
```

3. Explain the purpose of views in general and the function of the view V_SENIORS. Is it possible to modify the tuples in view V_SENIORS? In theory.
**Explanation:**



```sql
--Task 3
CREATE VIEW V_SENIORS AS
SELECT * FROM employee
WHERE salary > 45000
WITH CHECK OPTION;

-- A view is a virtual table created from a query.
-- It simplifies access, adds security, and hides complexity.

-- V_SENIORS shows only employees with salary > 45000.
-- WITH CHECK OPTION ensures that INSERT/UPDATE through this view
-- must keep salary > 45000, otherwise the change is rejected.

-- Yes, tuples in V_SENIORS can be modified, but only if the condition is still satisfied.
```

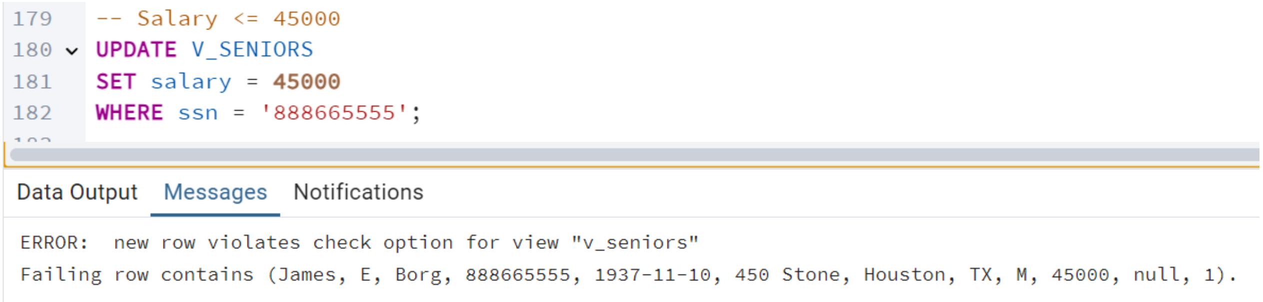
**Update:**



```sql
-- Salary > 45000
UPDATE V_SENIORS
SET salary = 46000
WHERE ssn = '888665555';
```

```
UPDATE 1
```

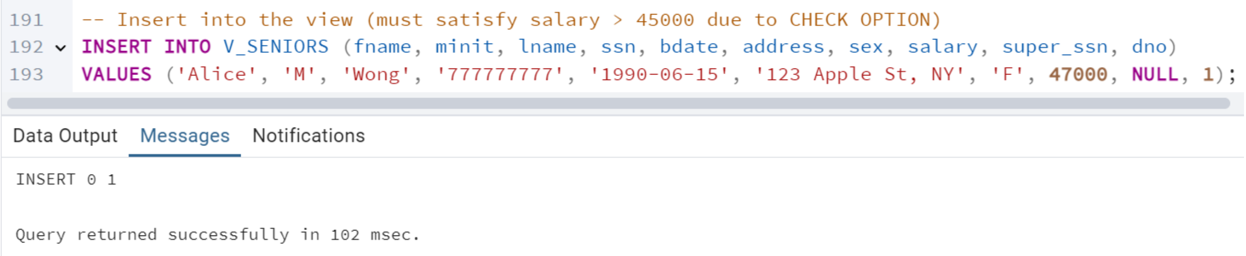
**Update rejection:**



```sql
-- Salary <= 45000
UPDATE V_SENIORS
SET salary = 45000
WHERE ssn = '888665555';
```

```
ERROR:  new row violates check option for view "v_seniors"
Failing row contains (James, E, Borg, 888665555, 1937-11-10, 450 Stone, Houston, TX, M, 45000, null, 1).
```

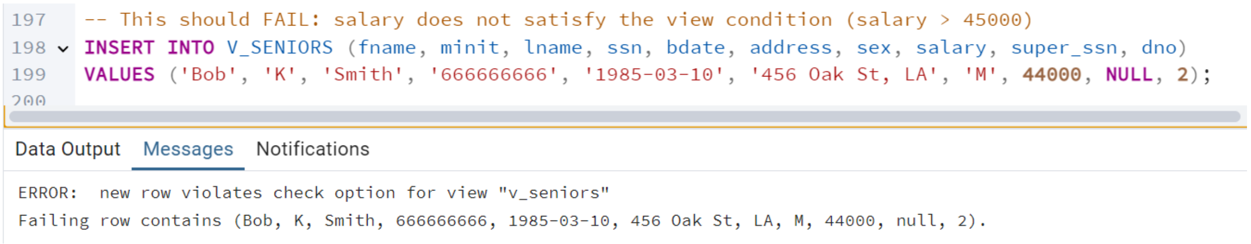
**Insert:**



```sql
-- Insert into the view (must satisfy salary > 45000 due to CHECK OPTION)
INSERT INTO V_SENIORS (fname, minit, lname, ssn, bdate, address, sex, salary, super_ssn, dno)
VALUES ('Alice', 'M', 'Wong', '777777777', '1990-06-15', '123 Apple St, NY', 'F', 47000, NULL, 1);
```

```
INSERT 0 1

Query returned successfully in 102 msec.
```

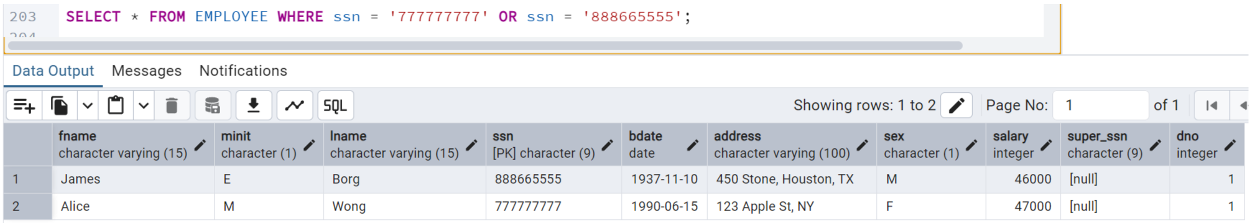
**Insert rejection:**



```sql
-- This should FAIL: salary does not satisfy the view condition (salary > 45000)
INSERT INTO V_SENIORS (fname, minit, lname, ssn, bdate, address, sex, salary, super_ssn, dno)
VALUES ('Bob', 'K', 'Smith', '666666666', '1985-03-10', '456 Oak St, LA', 'M', 44000, NULL, 2);
```

```
ERROR:  new row violates check option for view "v_seniors"
Failing row contains (Bob, K, Smith, 666666666, 1985-03-10, 456 Oak St, LA, M, 44000, null, 2).
```

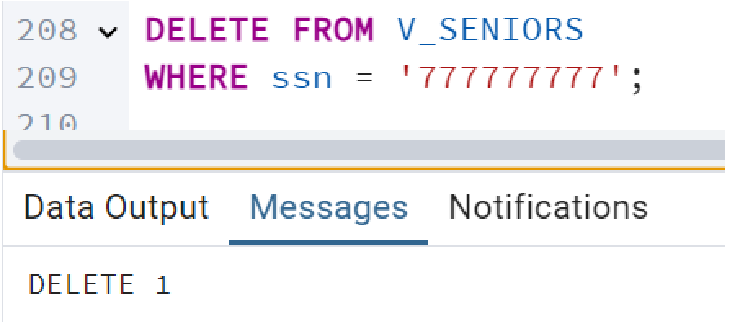
## After updating & inserting into the base table:



```sql
SELECT * FROM EMPLOYEE WHERE ssn = '777777777' OR ssn = '888665555';
```

| | fname<br>character varying (15) | minit<br>character (1) | lname<br>character varying (15) | ssn<br>[PK] character (9) | bdate<br>date | address<br>character varying (100) | sex<br>character (1) | salary<br>integer | super_ssn<br>character (9) | dno<br>integer |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | James | E | Borg | 888665555 | 1937-11-10 | 450 Stone, Houston, TX | M | 46000 | [null] | 1 |
| 2 | Alice | M | Wong | 777777777 | 1990-06-15 | 123 Apple St, NY | F | 47000 | [null] | 1 |

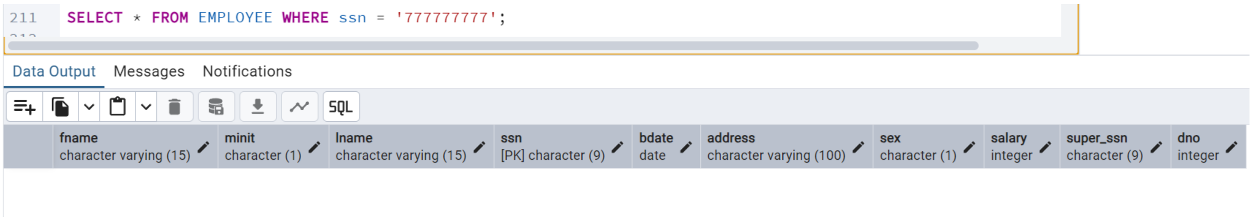
## Delete:



```sql
DELETE FROM V_SENIORS
WHERE ssn = '777777777';
```

```
DELETE 1
```

## After deleting tuple using View, the base table:



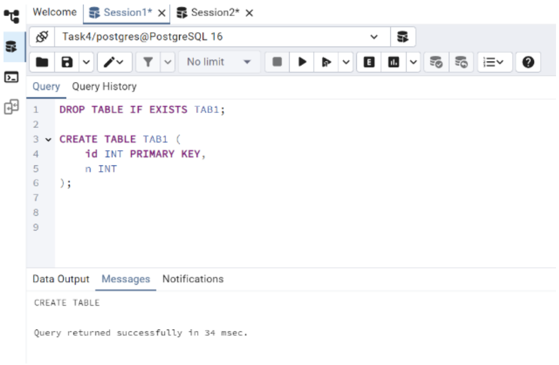
```sql
SELECT * FROM EMPLOYEE WHERE ssn = '777777777';
```

| | fname<br>character varying (15) | minit<br>character (1) | lname<br>character varying (15) | ssn<br>[PK] character (9) | bdate<br>date | address<br>character varying (100) | sex<br>character (1) | salary<br>integer | super_ssn<br>character (9) | dno<br>integer |
|---|---|---|---|---|---|---|---|---|---|---|

# Assignment 4: Transactions

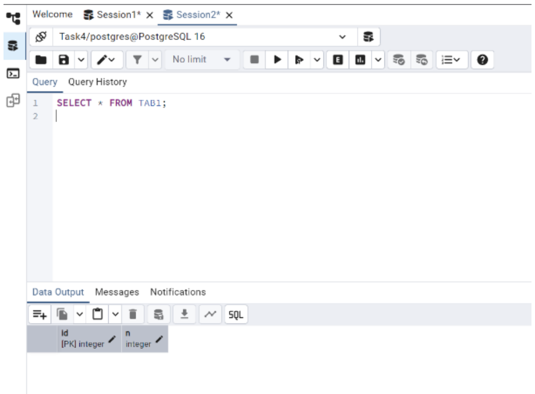
1. In session 1 run a command creating a table named TAB1 with two attributes:
   - id with data type integer, primary key
   - n with data type integer

   When is table TAB1 visible in session 2?



```sql
DROP TABLE IF EXISTS TAB1;

CREATE TABLE TAB1 (
    id INT PRIMARY KEY,
    n INT
);
```

```
CREATE TABLE

Query returned successfully in 34 msec.
```



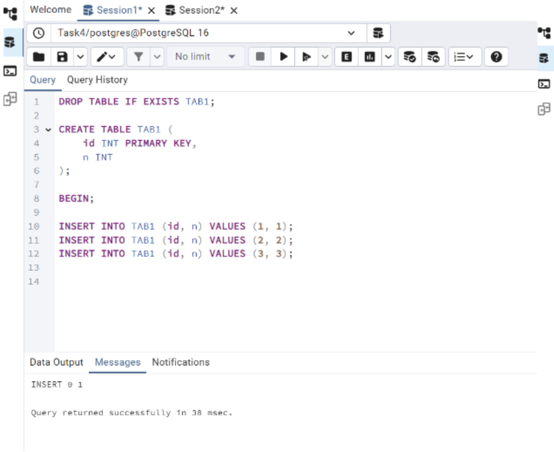
```sql
SELECT * FROM TAB1;
```

| id<br>[PK] integer | n<br>integer |
|---|---|

Once the table is created in session 1, it is also reflected in table 2 immediately.

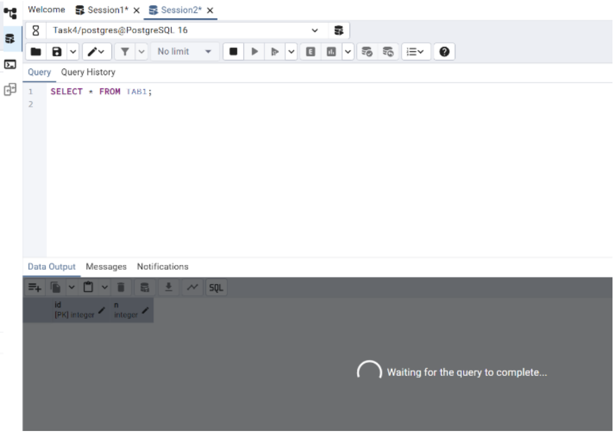
2. In session 1 insert the following tuples into TAB1 within one transaction: 1 (1,1), (2,2), (3,3)

Before committing changes: The values are not updated in session 2



```sql
DROP TABLE IF EXISTS TAB1;

CREATE TABLE TAB1 (
    id INT PRIMARY KEY,
    n INT
);

BEGIN;

INSERT INTO TAB1 (id, n) VALUES (1, 1);
INSERT INTO TAB1 (id, n) VALUES (2, 2);
INSERT INTO TAB1 (id, n) VALUES (3, 3);
```

```
INSERT 0 1

Query returned successfully in 38 msec.
```

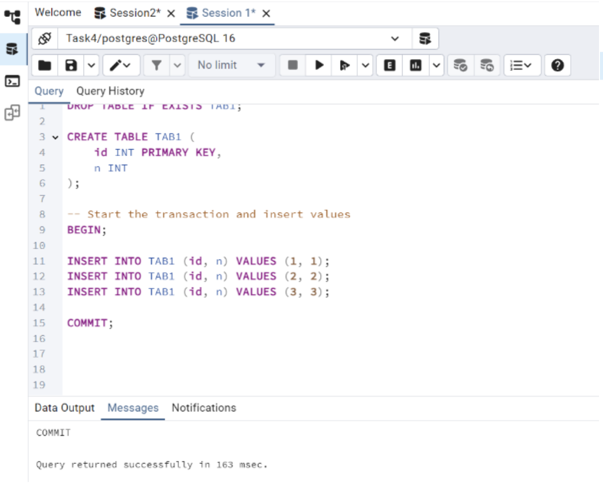


```sql
SELECT * FROM TAB1;
```

| id<br>[PK] integer | n<br>integer |
|---|---|

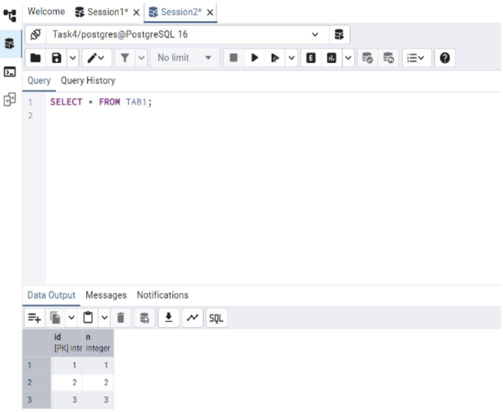
Waiting for the query to complete...

After committing changes: The values are updated in session 2



```
DROP TABLE IF EXISTS TAB1;

CREATE TABLE TAB1 (
    id INT PRIMARY KEY,
    n INT
);

-- Start the transaction and insert values
BEGIN;

INSERT INTO TAB1 (id, n) VALUES (1, 1);
INSERT INTO TAB1 (id, n) VALUES (2, 2);
INSERT INTO TAB1 (id, n) VALUES (3, 3);

COMMIT;
```

```
COMMIT

Query returned successfully in 163 msec.
```

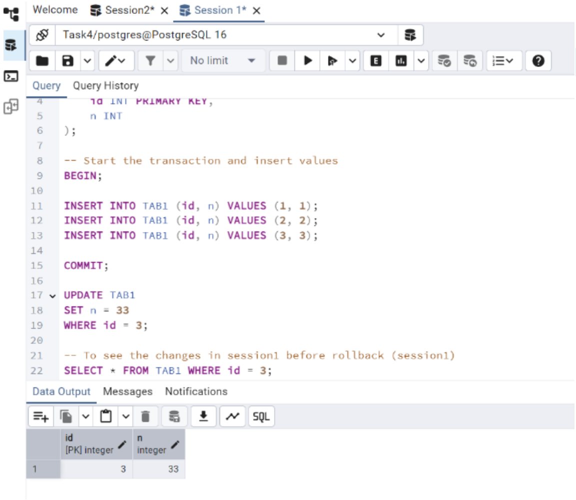


```
SELECT * FROM TAB1;
```

|  | id<br>[PK] inte | n<br>integer |
|---|---|---|
| 1 | 1 | 1 |
| 2 | 2 | 2 |
| 3 | 3 | 3 |

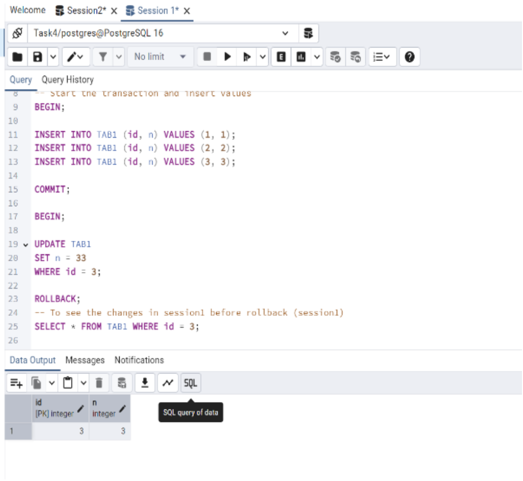
3. In session1 update the value of n to 33 for the tuple with id 3 (without committing). Afterwards rollback that transaction.

   - What value of n (id=3) is displayed in session 1 before and after the rollback?



```
    id INT PRIMARY KEY,
    n INT
);

-- Start the transaction and insert values
BEGIN;

INSERT INTO TAB1 (id, n) VALUES (1, 1);
INSERT INTO TAB1 (id, n) VALUES (2, 2);
INSERT INTO TAB1 (id, n) VALUES (3, 3);

COMMIT;

UPDATE TAB1
SET n = 33
WHERE id = 3;

-- To see the changes in session1 before rollback (session1)
SELECT * FROM TAB1 WHERE id = 3;
```

|  | id<br>[PK] integer | n<br>integer |
|---|---|---|
| 1 | 3 | 33 |

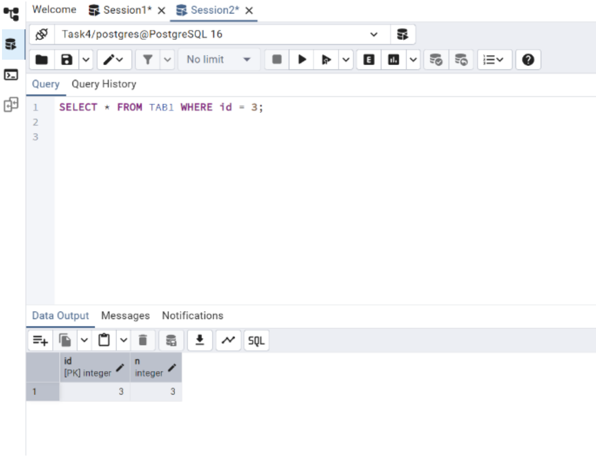


```
-- Start the transaction and insert values
BEGIN;

INSERT INTO TAB1 (id, n) VALUES (1, 1);
INSERT INTO TAB1 (id, n) VALUES (2, 2);
INSERT INTO TAB1 (id, n) VALUES (3, 3);

COMMIT;

BEGIN;

UPDATE TAB1
SET n = 33
WHERE id = 3;

ROLLBACK;
-- To see the changes in session1 before rollback (session1)
SELECT * FROM TAB1 WHERE id = 3;
```

|  | id<br>[PK] integer | n<br>integer |
|---|---|---|
| 1 | 3 | 3 |

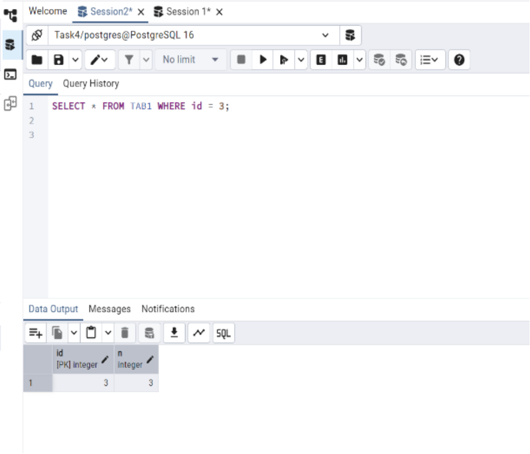
The value changes to 33 after updating in session 1 but changes back to 3 after the rollback

- What value of n (id=3) is displayed in session 2 before and after the rollback?



```
SELECT * FROM TAB1 WHERE id = 3;
```

|  | id<br>[PK] integer | n<br>integer |
|---|---|---|
| 1 | 3 | 3 |



```
SELECT * FROM TAB1 WHERE id = 3;
```

|  | id<br>[PK] integer | n<br>integer |
|---|---|---|
| 1 | 3 | 3 |

The value before the rollback in session 2 is  3 because we did not commit changes, after the rollback it is still 3 because we reverted the changes we made.