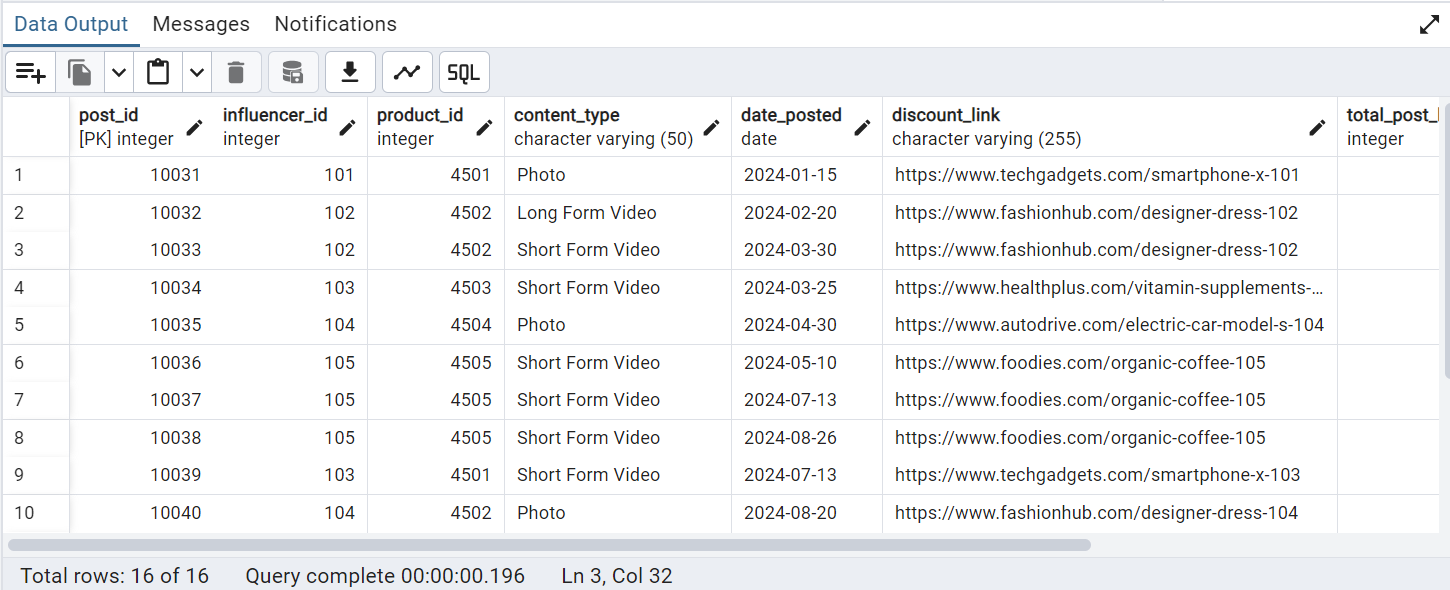
**SQL QUERIES OUTPUT SCREENSHOTS**

set search\_path to influencer\_partnership

**Q1: Select all columns and all rows from one table (5 points)**

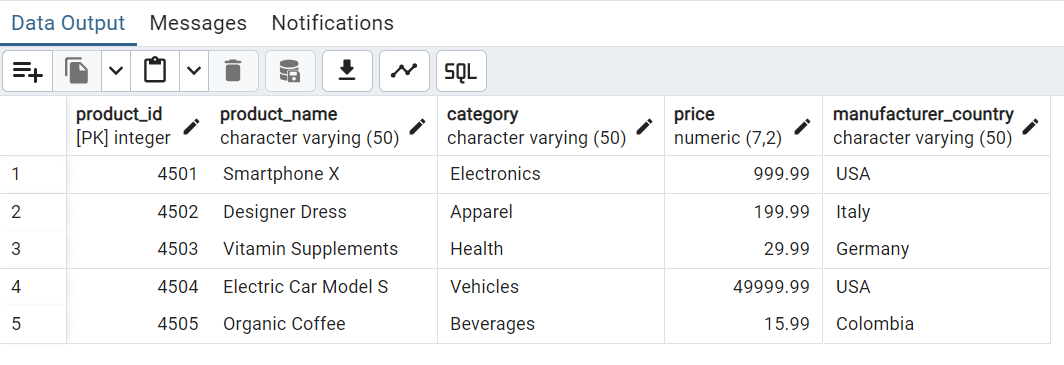
SELECT \* FROM content\_creation;



**Q2: Select five columns and all rows from one table (5 points)**

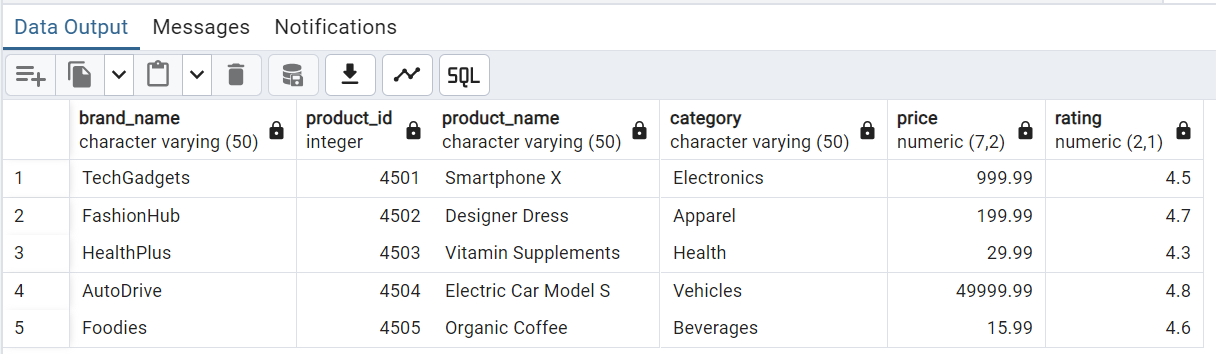
SELECT product\_id, product\_name, category, price, manufacturer\_country

FROM product;



**Q3: Select all columns from all rows from one view (5 points)**

SELECT \* FROM ProductInfo;

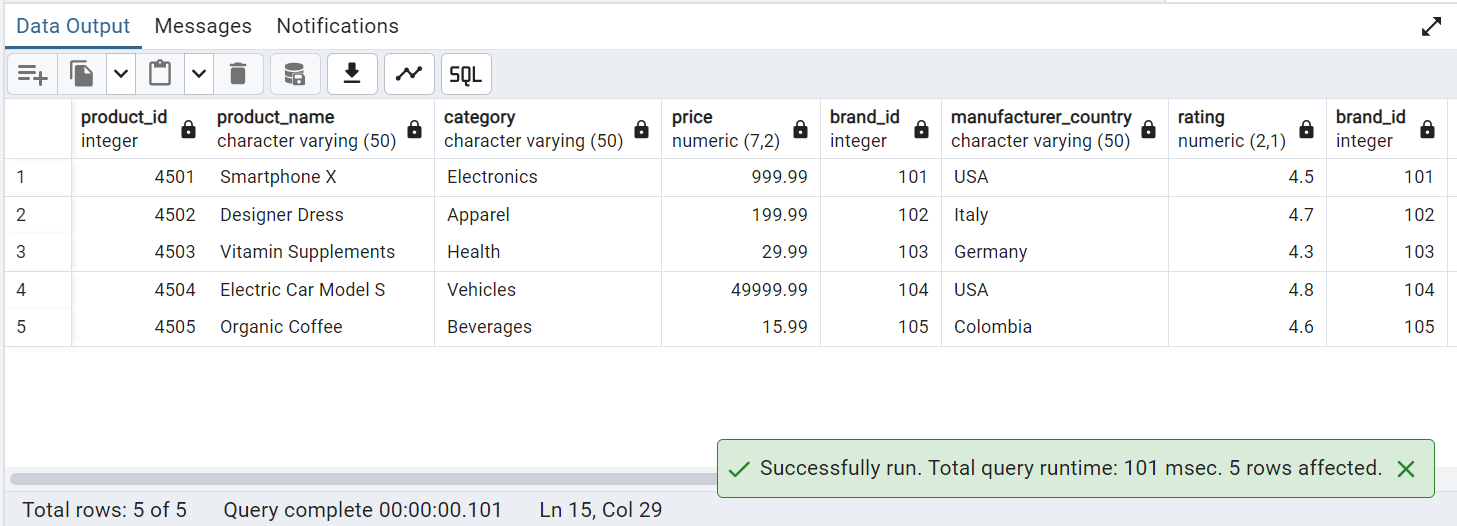


**Q4: Using a join on 2 tables, select all columns and all rows from the tables without the use of a Cartesian product (5 points)**

SELECT \* FROM product p

JOIN brand b

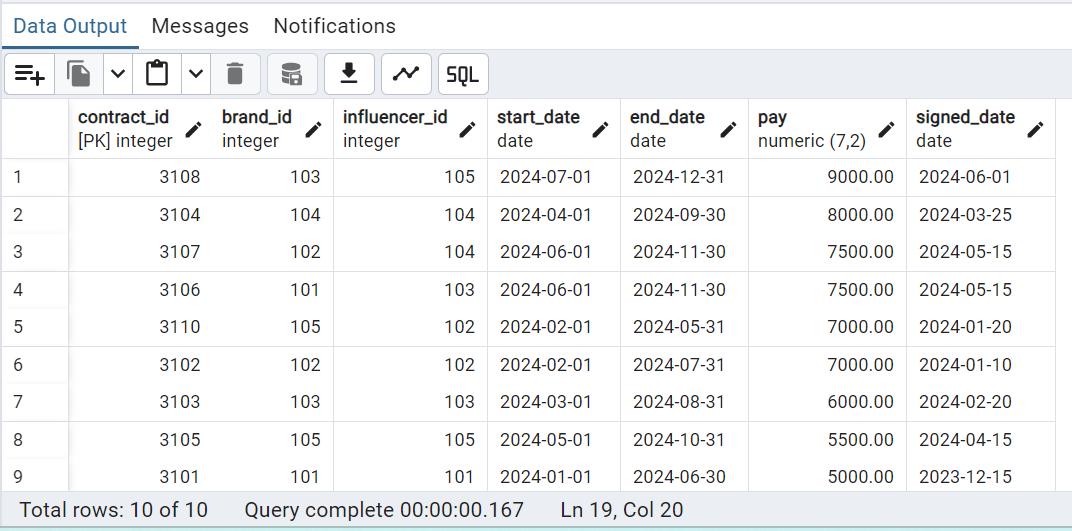
ON p.brand\_id = b.brand\_id ;



**Q5: Select and order data retrieved from one table (5 points)**

SELECT \* FROM contract

ORDER BY pay DESC ;



**Q6: Using a join on 3 tables, select 5 columns from the 3 tables. Use syntax that would limit the output to 3 rows (5 points)**

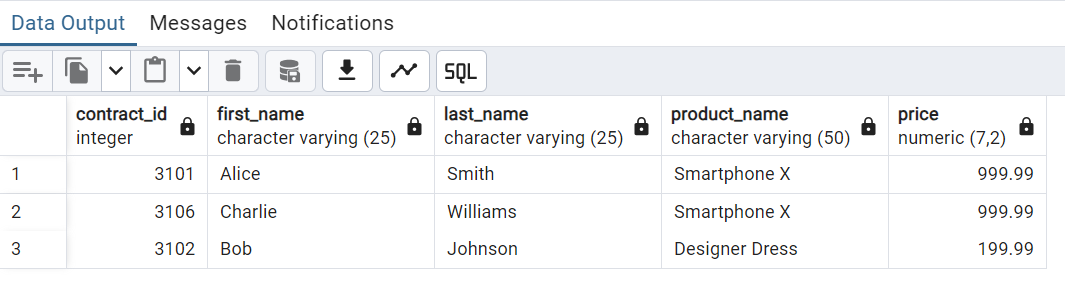
SELECT c.contract\_id, i.first\_name, i.last\_name, p.product\_name, p.price

FROM contract c

JOIN influencer i ON c.influencer\_id = i.influencer\_id

JOIN product p ON c.brand\_id = p.brand\_id

LIMIT 3;



**Q7: Select distinct rows using joins on 3 tables (5 points)**

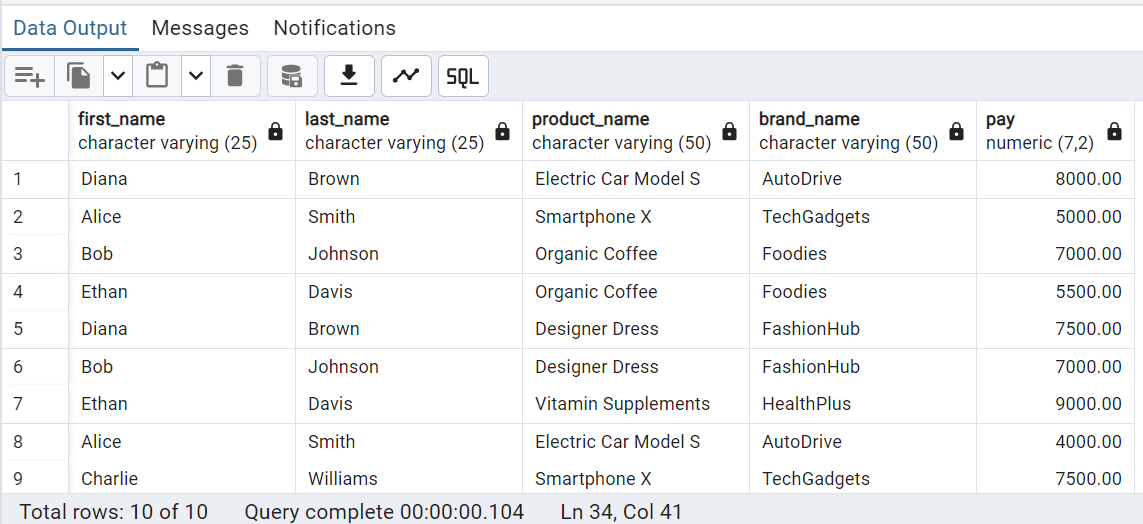
SELECT DISTINCT i.first\_name, i.last\_name, p.product\_name, b.brand\_name, c.pay

FROM contract c

JOIN influencer i ON c.influencer\_id = i.influencer\_id

JOIN product p ON c.brand\_id = p.brand\_id

JOIN brand b ON p.brand\_id = b.brand\_id;



**Q8: Use GROUP BY and HAVING in a select statement using one or more tables (5 points)**

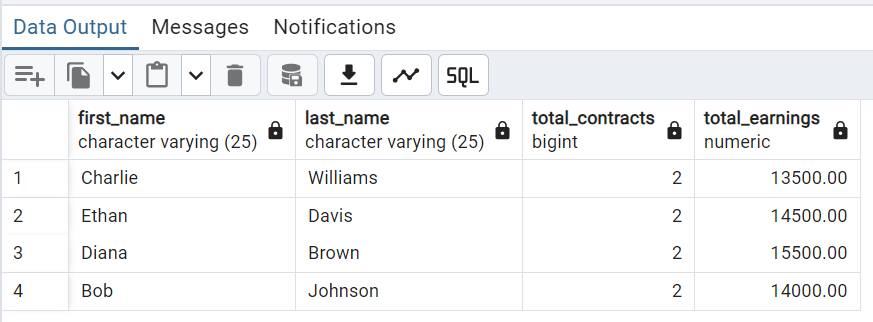
SELECT i.first\_name, i.last\_name, COUNT(c.contract\_id) AS total\_contracts, SUM(c.pay) AS total\_earnings

FROM influencer i

JOIN contract c ON i.influencer\_id = c.influencer\_id

GROUP BY i.first\_name, i.last\_name

HAVING SUM(c.pay) > 10000;



**Q9: Use IN clause to select data from one or more tables (5 points)**

SELECT p.product\_name, p.category, p.price

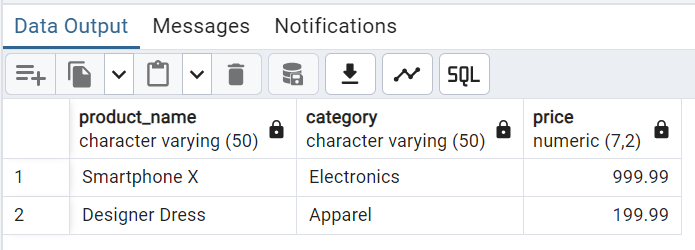
FROM product p

WHERE p.brand\_id IN (

SELECT b.brand\_id FROM brand b

WHERE b.industry IN ('Technology', 'Fashion')

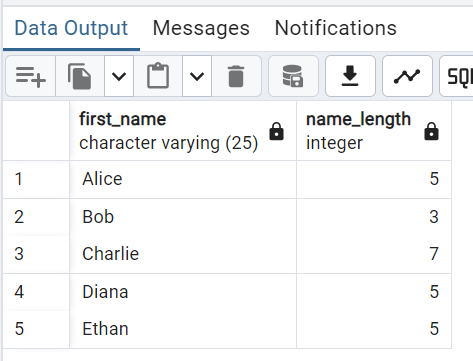
);



**Q10: Select length of one column from one table (use LENGTH function) (5 points)**

SELECT first\_name, LENGTH(first\_name) AS name\_length

FROM influencer;



**Q11: Delete one record from one table. Use select statements to demonstrate the table contents before and after the DELETE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed (5 points)**

-- View Table Contents Before Delete

SELECT \* FROM content\_creation;

BEGIN;

DELETE FROM content\_creation

WHERE post\_id = 10031;

-- View Table Contents After Delete

SELECT \* FROM content\_creation;

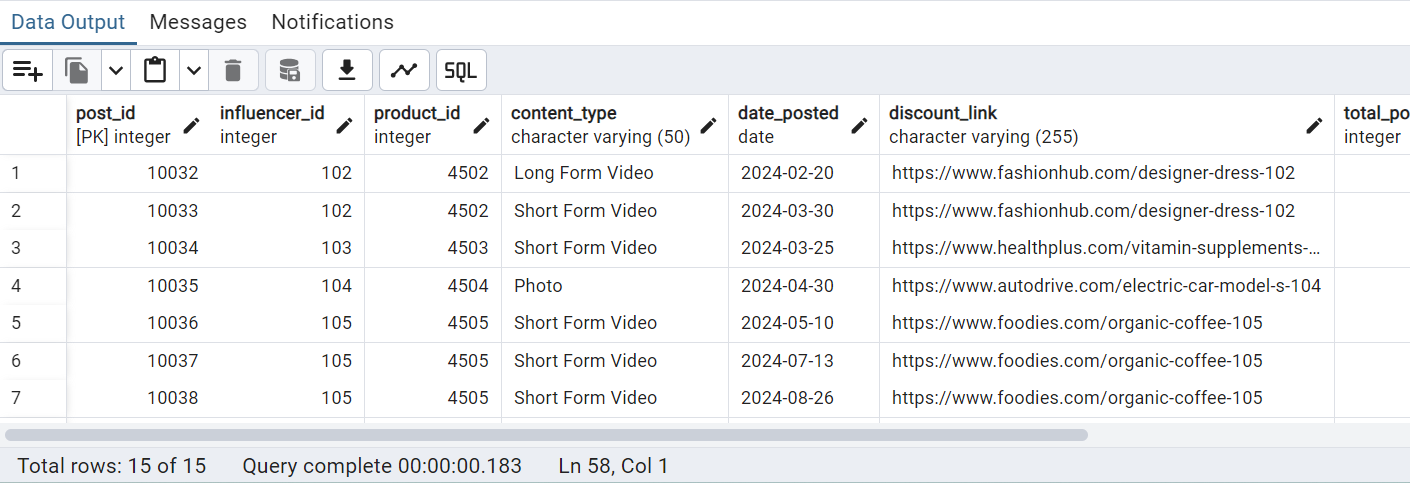
-- Rollback the Transaction

ROLLBACK;

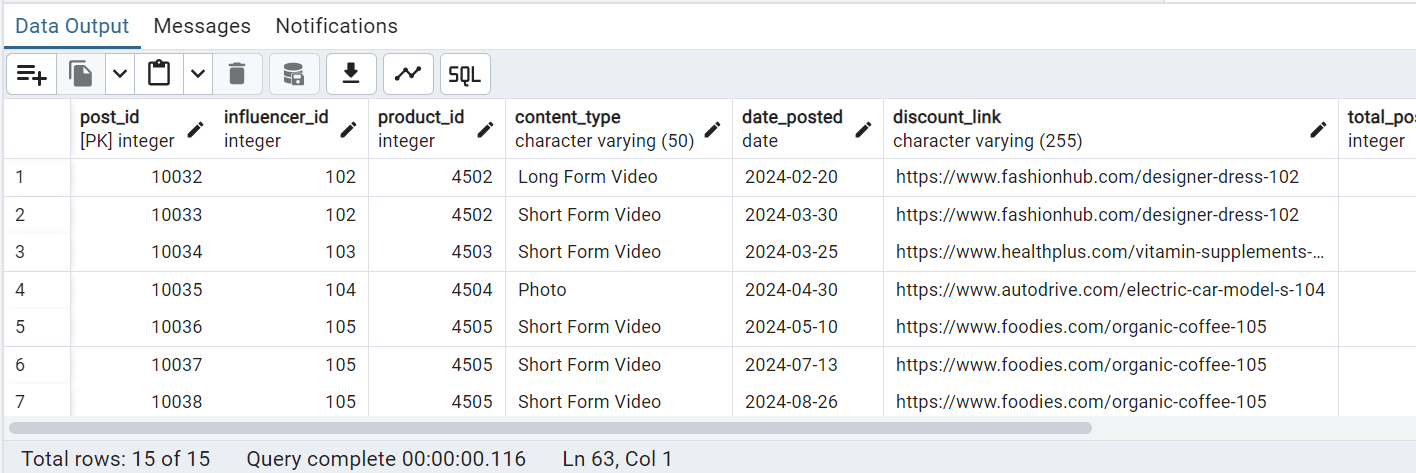
-- Verify Table Contents After Rollback

SELECT \* FROM content\_creation;

##BEFORE



##AFTER



**Q12: Update one record from one table. Use select statements to demonstrate the table contents before and after the UPDATE statement. Make sure you use ROLLBACK afterwards so that the data will not be physically removed (5 points) Update a record and demonstrate using ROLLBACK**

-- View Table Contents Before Update

SELECT \* FROM influencer;

-- Start a Transaction

BEGIN;

-- Update a Record

UPDATE influencer

SET platform = 'TikTok'

WHERE influencer\_id = 101;

-- View Table Contents After Update

SELECT \* FROM influencer;

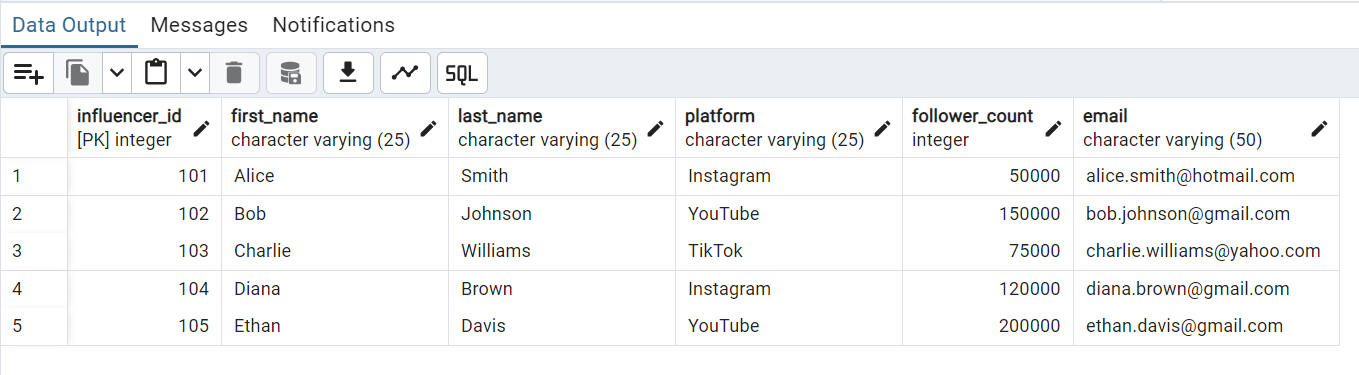
-- Rollback the Transaction

ROLLBACK;

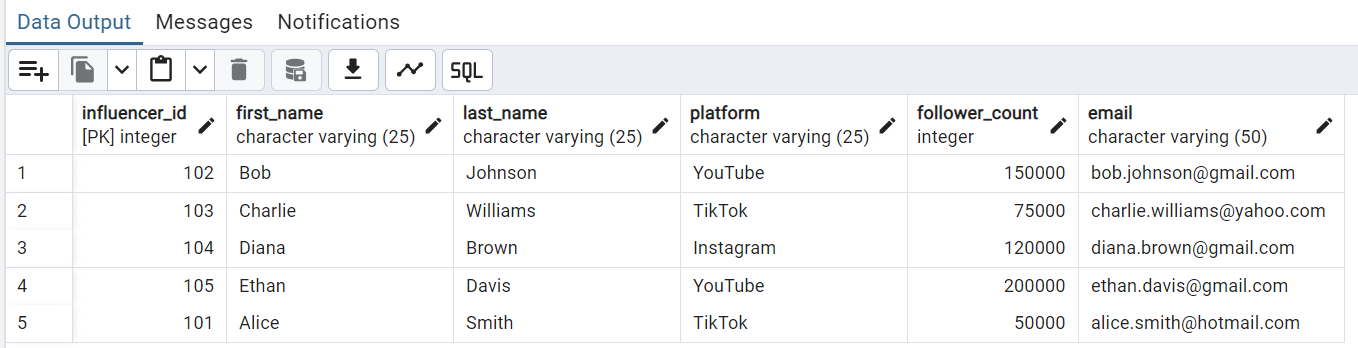
-- Verify Table Contents After Rollback

SELECT \* FROM influencer;

##BEFORE



##AFTER



**Q13: Classify each influencer by type (nano, micro, macro, or celebrity) based on their follower count.**

SELECT

i.influencer\_id,

i.first\_name,

i.last\_name,

CASE

WHEN follower\_count >= 190000 THEN 'Celebrity'

WHEN follower\_count >= 100000 THEN 'Macro'

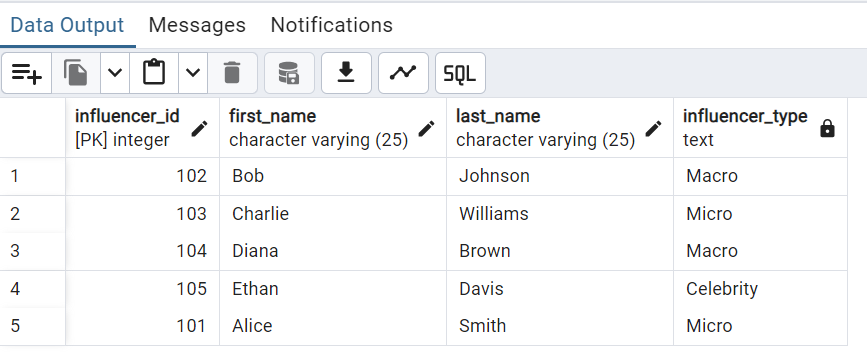
WHEN follower\_count >= 10000 THEN 'Micro'

ELSE 'Nano'

END AS influencer\_type

FROM

influencer i ;



**Q14: Identify the Top 3 Most Profitable Products per Brand**

SELECT

b.brand\_name,

p.product\_name,

SUM(pm.revenue) AS total\_revenue

FROM

product p

JOIN

performance\_metrics pm ON p.product\_id = pm.product\_id

JOIN

brand b ON p.brand\_id = b.brand\_id

GROUP BY

b.brand\_name, p.product\_name

HAVING

SUM(pm.revenue) > 0

ORDER BY

b.brand\_name, total\_revenue DESC

LIMIT 3;

