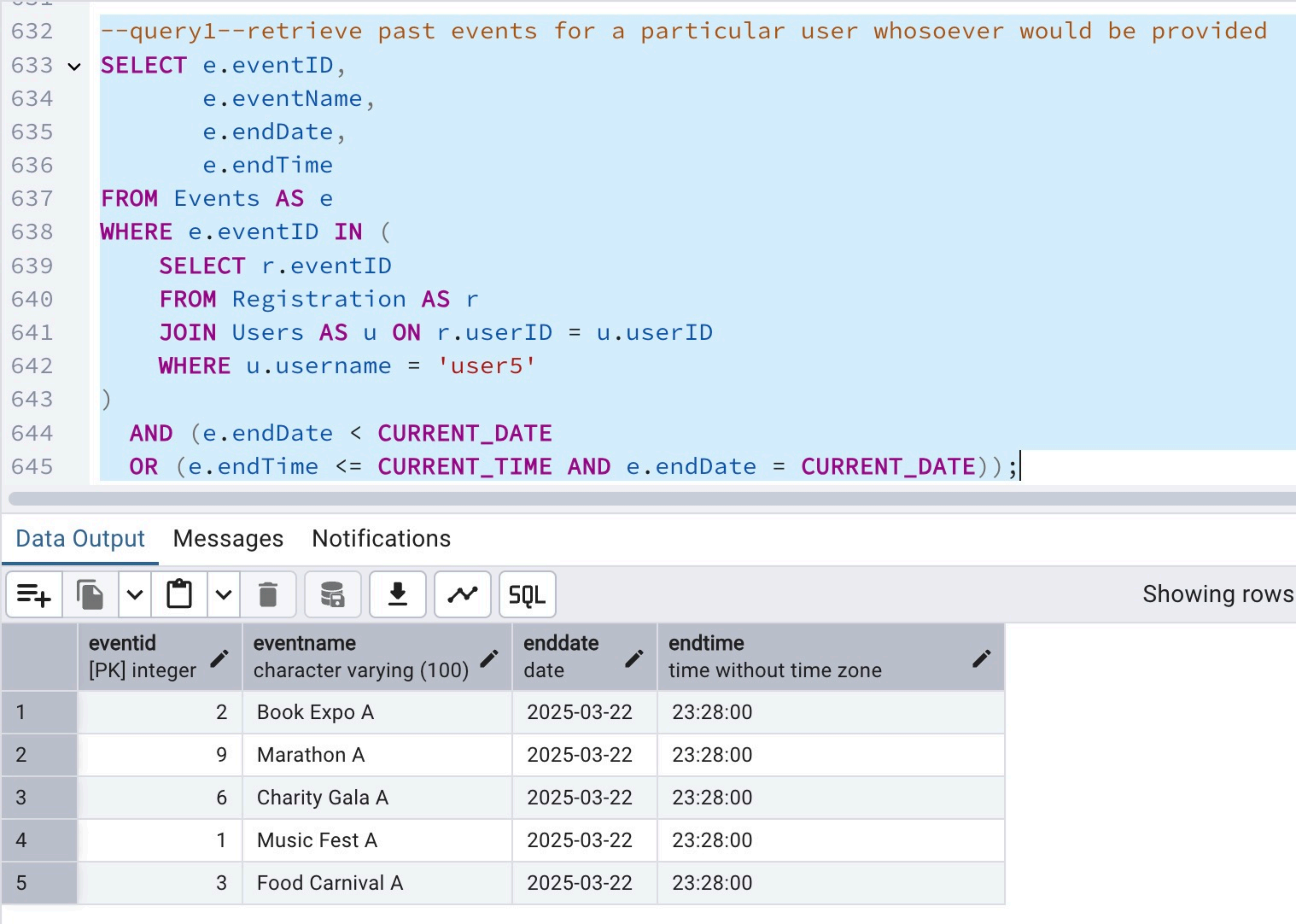
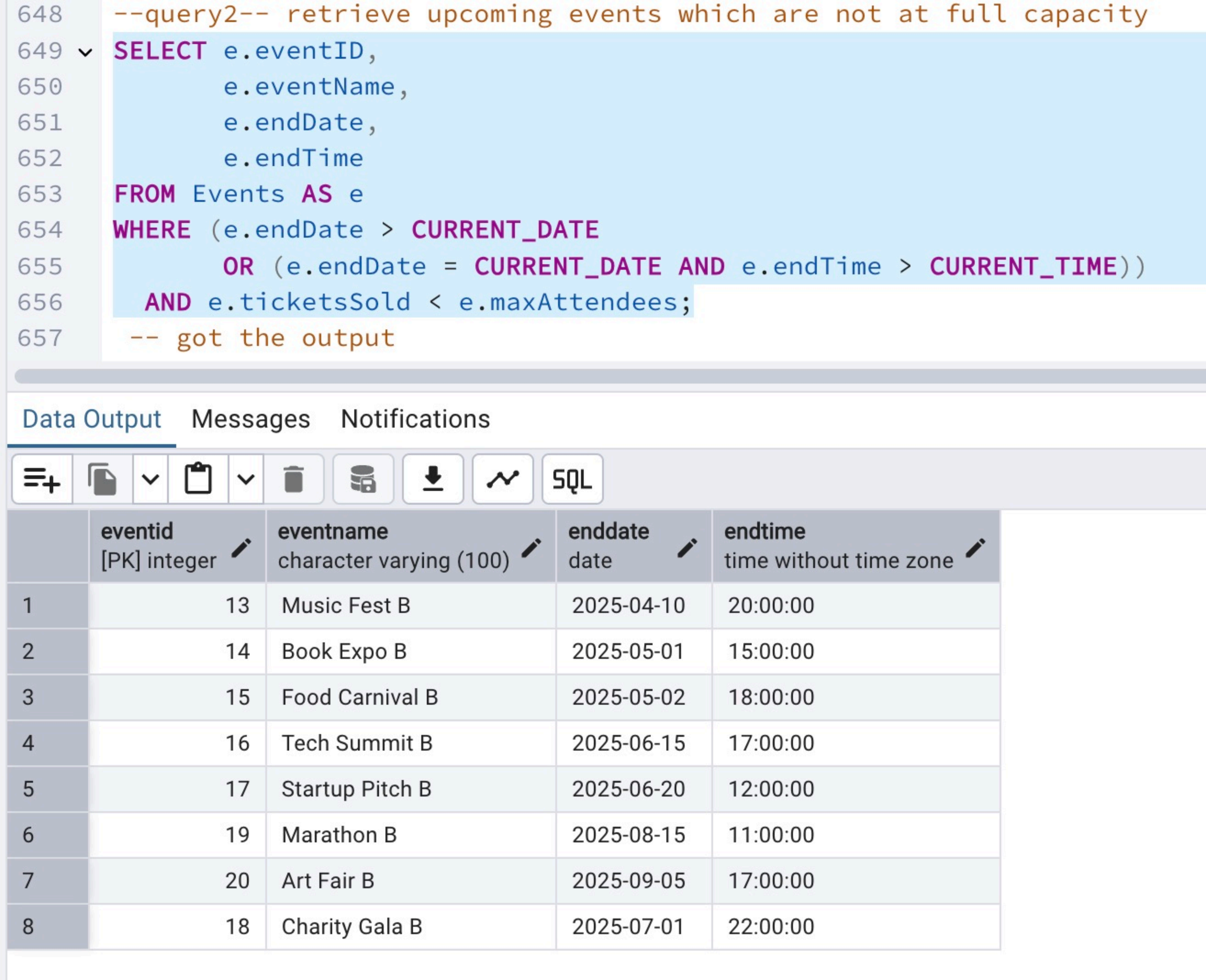


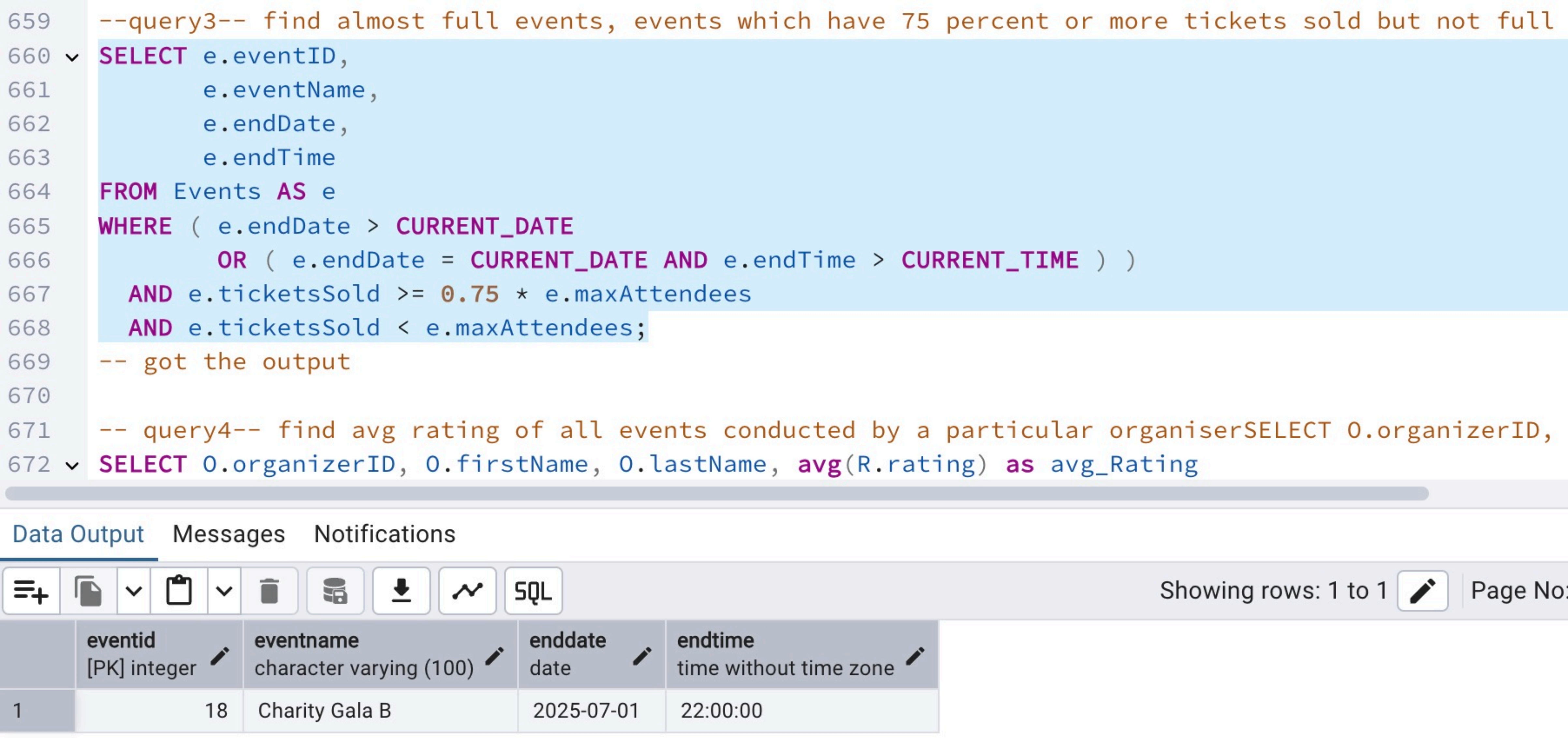
```sql
--query1--retrieve past events for a particular user whosoever would be provided
SELECT e.eventID,
       e.eventName,
       e.endDate,
       e.endTime
FROM Events AS e
WHERE e.eventID IN (
    SELECT r.eventID
    FROM Registration AS r
    JOIN Users AS u ON r.userID = u.userID
    WHERE u.username = 'user5'
)
  AND (e.endDate < CURRENT_DATE
  OR (e.endTime <= CURRENT_TIME AND e.endDate = CURRENT_DATE));
```

	eventid [PK] integer	eventname character varying (100)	enddate date	endtime time without time zone
1	2	Book Expo A	2025-03-22	23:28:00
2	9	Marathon A	2025-03-22	23:28:00
3	6	Charity Gala A	2025-03-22	23:28:00
4	1	Music Fest A	2025-03-22	23:28:00
5	3	Food Carnival A	2025-03-22	23:28:00



```sql
--query2-- retrieve upcoming events which are not at full capacity
SELECT e.eventID,
       e.eventName,
       e.endDate,
       e.endTime
FROM Events AS e
WHERE (e.endDate > CURRENT_DATE
       OR (e.endDate = CURRENT_DATE AND e.endTime > CURRENT_TIME))
  AND e.ticketsSold < e.maxAttendees;
 -- got the output
```

	eventid [PK] integer	eventname character varying (100)	enddate date	endtime time without time zone
1	13	Music Fest B	2025-04-10	20:00:00
2	14	Book Expo B	2025-05-01	15:00:00
3	15	Food Carnival B	2025-05-02	18:00:00
4	16	Tech Summit B	2025-06-15	17:00:00
5	17	Startup Pitch B	2025-06-20	12:00:00
6	19	Marathon B	2025-08-15	11:00:00
7	20	Art Fair B	2025-09-05	17:00:00
8	18	Charity Gala B	2025-07-01	22:00:00



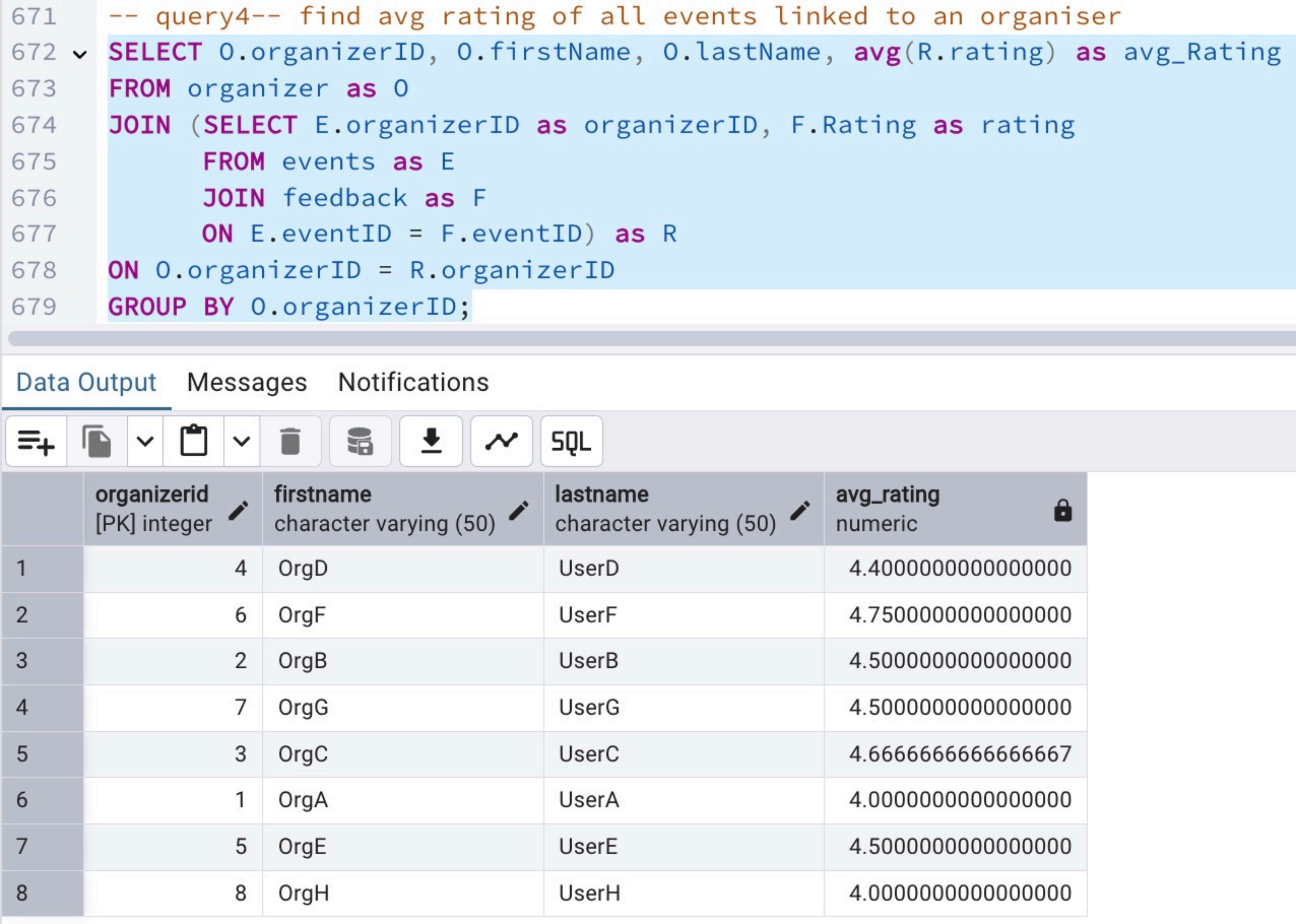
```
--query3-- find almost full events, events which have 75 percent or more tickets sold but not full
SELECT e.eventID,
       e.eventName,
       e.endDate,
       e.endTime
FROM Events AS e
WHERE ( e.endDate > CURRENT_DATE
        OR ( e.endDate = CURRENT_DATE AND e.endTime > CURRENT_TIME ) )
  AND e.ticketsSold >= 0.75 * e.maxAttendees
  AND e.ticketsSold < e.maxAttendees;
-- got the output

-- query4-- find avg rating of all events conducted by a particular organiserSELECT O.organizerID,
SELECT O.organizerID, O.firstName, O.lastName, avg(R.rating) as avg_Rating
```

Data Output   Messages   Notifications

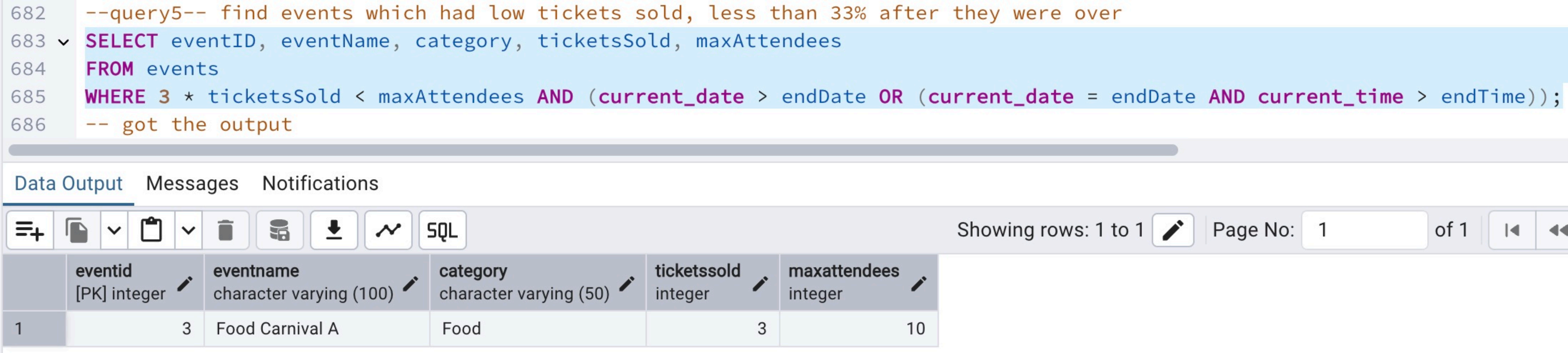
Showing rows: 1 to 1   Page No:

	eventid [PK] integer	eventname character varying (100)	enddate date	endtime time without time zone
1	18	Charity Gala B	2025-07-01	22:00:00



```
-- query4-- find avg rating of all events linked to an organiser
SELECT O.organizerID, O.firstName, O.lastName, avg(R.rating) as avg_Rating
FROM organizer as O
JOIN (SELECT E.organizerID as organizerID, F.Rating as rating
      FROM events as E
      JOIN feedback as F
      ON E.eventID = F.eventID) as R
ON O.organizerID = R.organizerID
GROUP BY O.organizerID;
```

	organizerid [PK] integer	firstname character varying (50)	lastname character varying (50)	avg_rating numeric
1	4	OrgD	UserD	4.4000000000000000
2	6	OrgF	UserF	4.7500000000000000
3	2	OrgB	UserB	4.5000000000000000
4	7	OrgG	UserG	4.5000000000000000
5	3	OrgC	UserC	4.6666666666666667
6	1	OrgA	UserA	4.0000000000000000
7	5	OrgE	UserE	4.5000000000000000
8	8	OrgH	UserH	4.0000000000000000

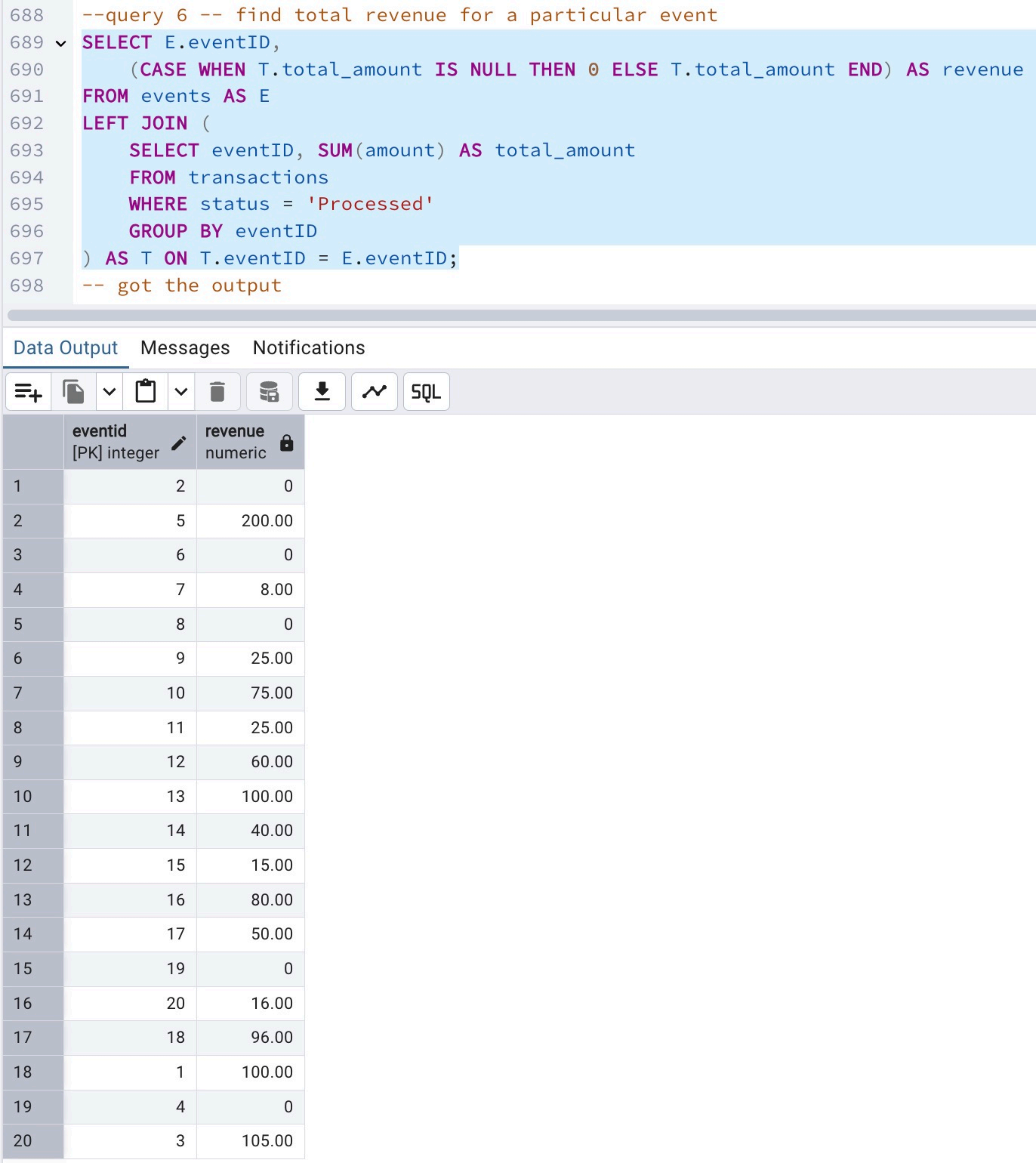


```sql
--query5-- find events which had low tickets sold, less than 33% after they were over
SELECT eventID, eventName, category, ticketsSold, maxAttendees
FROM events
WHERE 3 * ticketsSold < maxAttendees AND (current_date > endDate OR (current_date = endDate AND current_time > endTime));
-- got the output
```

Data Output | Messages | Notifications

Showing rows: 1 to 1 | Page No: 1 of 1

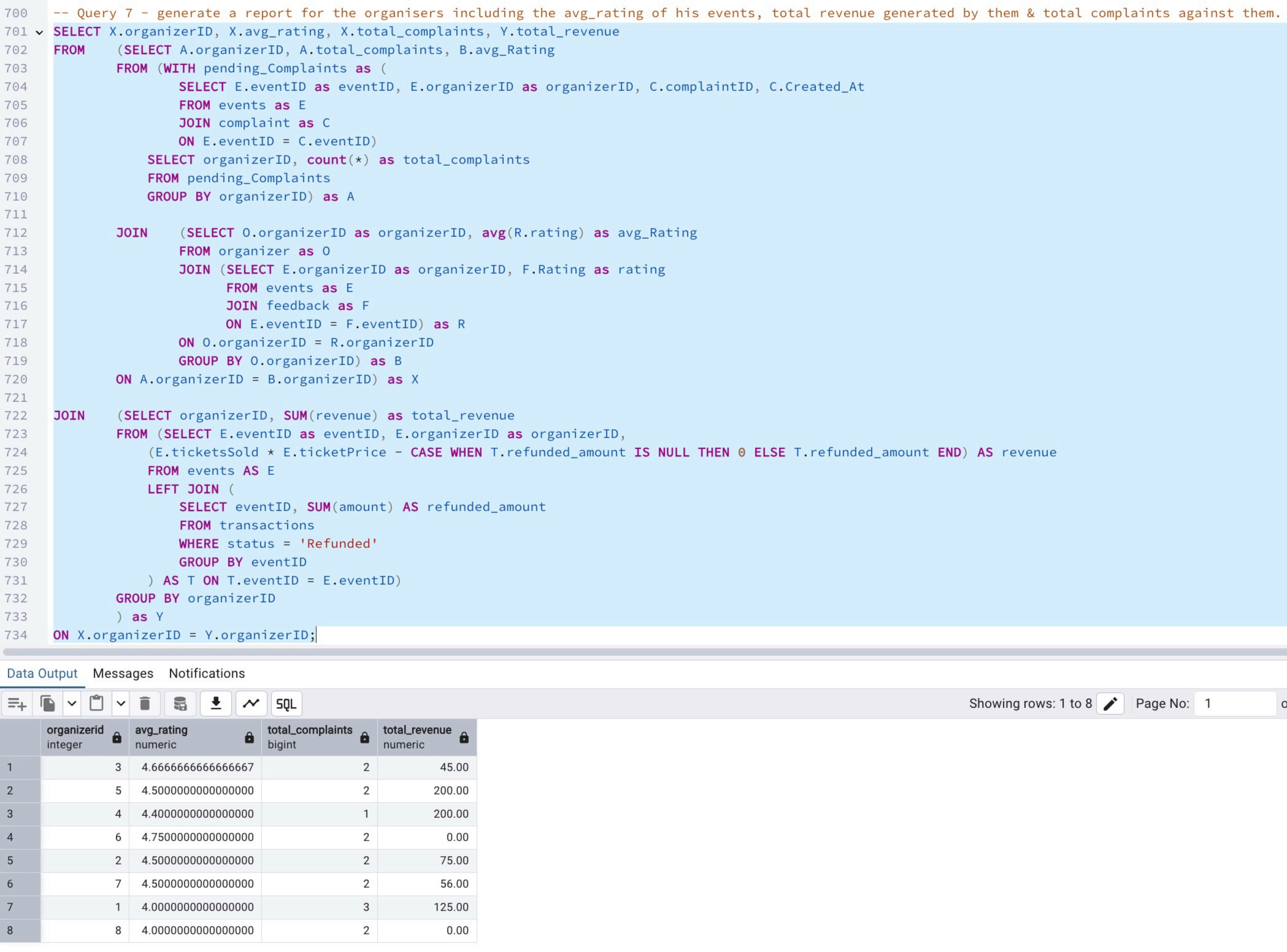
	eventid [PK] integer	eventname character varying (100)	category character varying (50)	ticketssold integer	maxattendees integer
1	3	Food Carnival A	Food	3	10



```
--query 6 -- find total revenue for a particular event
SELECT E.eventID,
    (CASE WHEN T.total_amount IS NULL THEN 0 ELSE T.total_amount END) AS revenue
FROM events AS E
LEFT JOIN (
    SELECT eventID, SUM(amount) AS total_amount
    FROM transactions
    WHERE status = 'Processed'
    GROUP BY eventID
) AS T ON T.eventID = E.eventID;
-- got the output
```

Data Output Messages Notifications

	eventid [PK] integer	revenue numeric
1	2	0
2	5	200.00
3	6	0
4	7	8.00
5	8	0
6	9	25.00
7	10	75.00
8	11	25.00
9	12	60.00
10	13	100.00
11	14	40.00
12	15	15.00
13	16	80.00
14	17	50.00
15	19	0
16	20	16.00
17	18	96.00
18	1	100.00
19	4	0
20	3	105.00

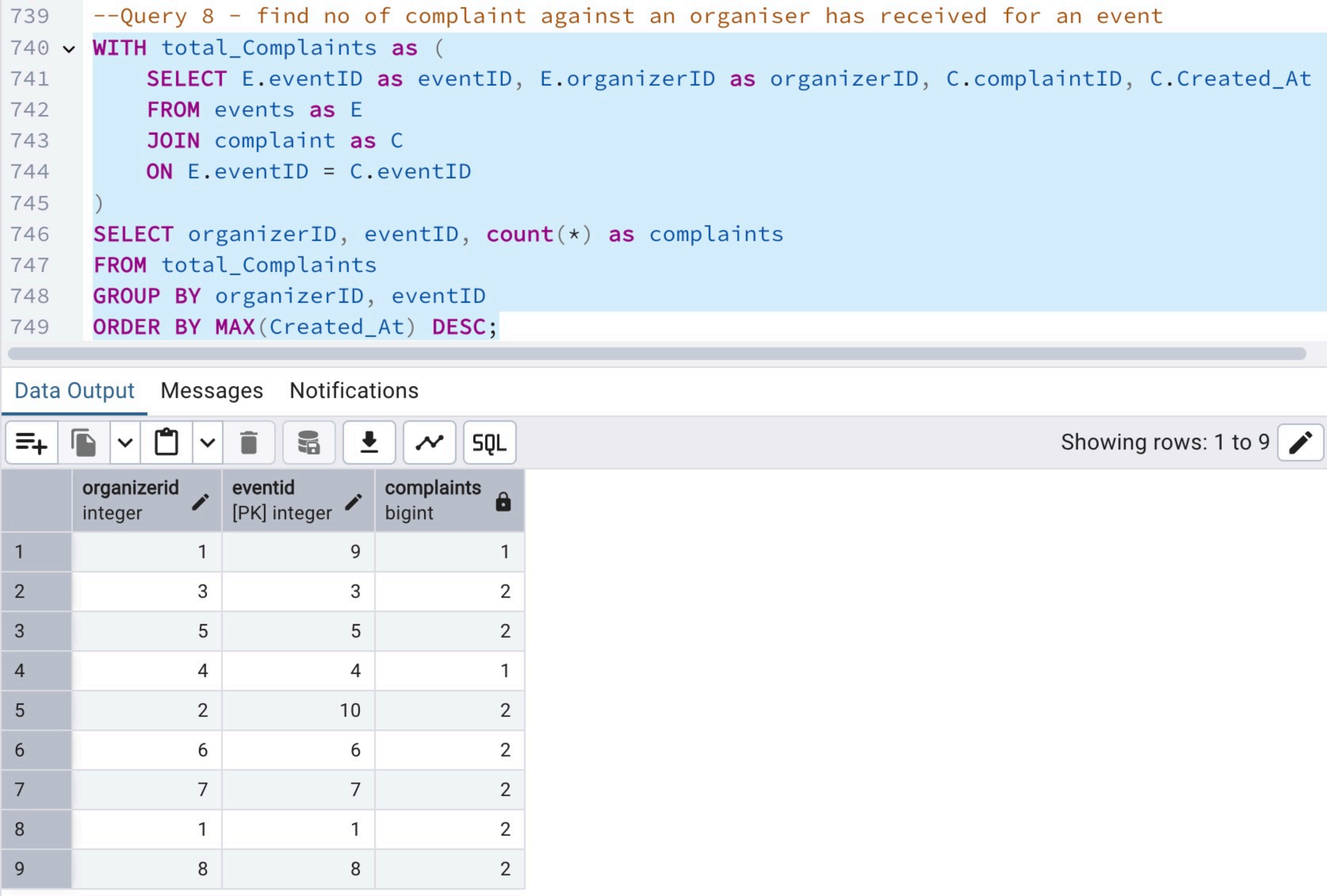


```sql
-- Query 7 - generate a report for the organisers including the avg_rating of his events, total revenue generated by them & total complaints against them.
SELECT X.organizerID, X.avg_rating, X.total_complaints, Y.total_revenue
FROM    (SELECT A.organizerID, A.total_complaints, B.avg_Rating
        FROM (WITH pending_Complaints as (
                SELECT E.eventID as eventID, E.organizerID as organizerID, C.complaintID, C.Created_At
                FROM events as E
                JOIN complaint as C
                ON E.eventID = C.eventID)
              SELECT organizerID, count(*) as total_complaints
              FROM pending_Complaints
              GROUP BY organizerID) as A

        JOIN    (SELECT O.organizerID as organizerID, avg(R.rating) as avg_Rating
                FROM organizer as O
                JOIN (SELECT E.organizerID as organizerID, F.Rating as rating
                      FROM events as E
                      JOIN feedback as F
                      ON E.eventID = F.eventID) as R
                ON O.organizerID = R.organizerID
                GROUP BY O.organizerID) as B
        ON A.organizerID = B.organizerID) as X

JOIN    (SELECT organizerID, SUM(revenue) as total_revenue
        FROM (SELECT E.eventID as eventID, E.organizerID as organizerID,
            (E.ticketsSold * E.ticketPrice - CASE WHEN T.refunded_amount IS NULL THEN 0 ELSE T.refunded_amount END) AS revenue
            FROM events AS E
            LEFT JOIN (
                SELECT eventID, SUM(amount) AS refunded_amount
                FROM transactions
                WHERE status = 'Refunded'
                GROUP BY eventID
            ) AS T ON T.eventID = E.eventID)
        GROUP BY organizerID
        ) as Y
ON X.organizerID = Y.organizerID;
```

	organizerid integer	avg_rating numeric	total_complaints bigint	total_revenue numeric
1	3	4.6666666666666667	2	45.00
2	5	4.5000000000000000	2	200.00
3	4	4.4000000000000000	1	200.00
4	6	4.7500000000000000	2	0.00
5	2	4.5000000000000000	2	75.00
6	7	4.5000000000000000	2	56.00
7	1	4.0000000000000000	3	125.00
8	8	4.0000000000000000	2	0.00

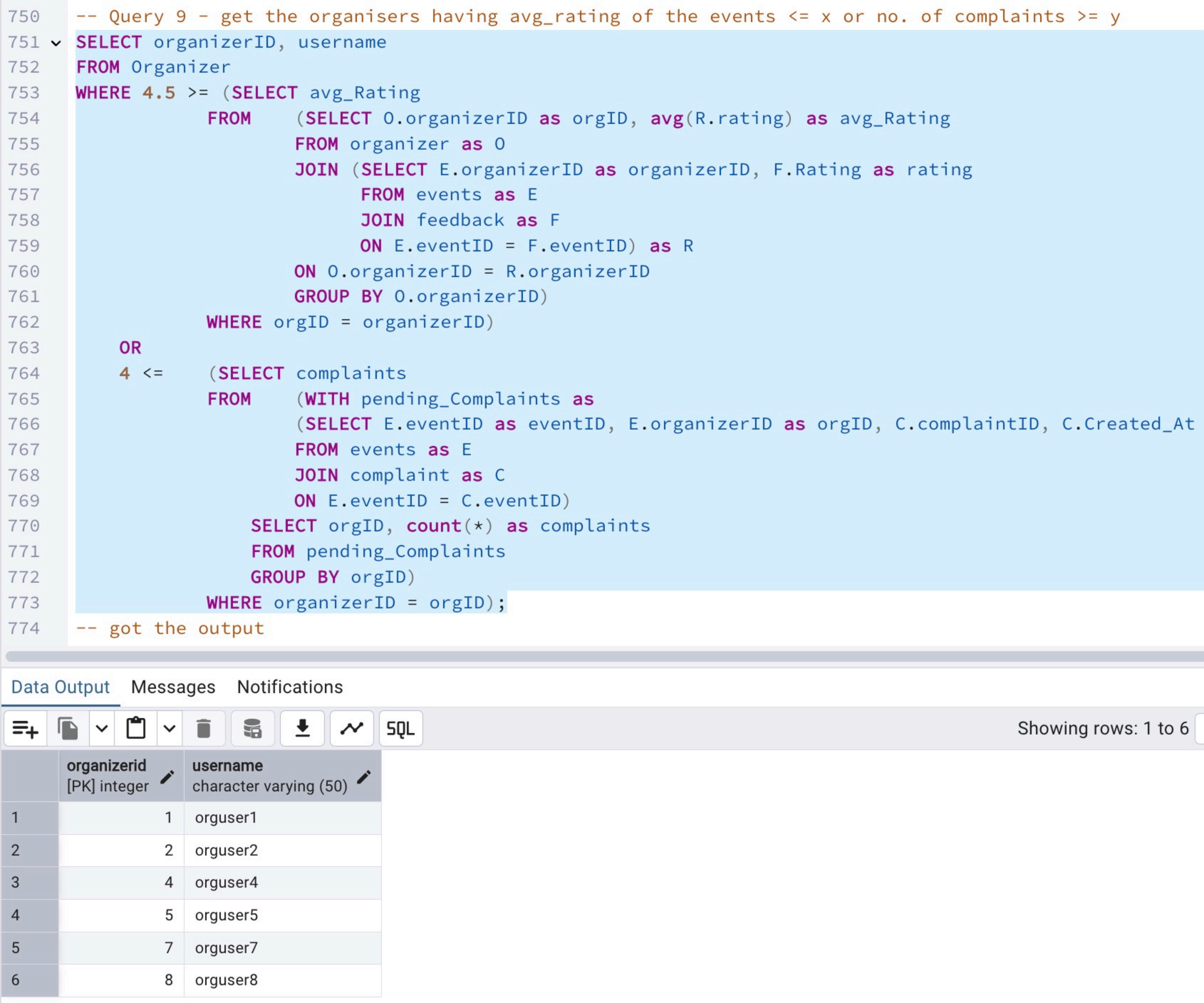


```sql
--Query 8 - find no of complaint against an organiser has received for an event
WITH total_Complaints as (
    SELECT E.eventID as eventID, E.organizerID as organizerID, C.complaintID, C.Created_At
    FROM events as E
    JOIN complaint as C
    ON E.eventID = C.eventID
)
SELECT organizerID, eventID, count(*) as complaints
FROM total_Complaints
GROUP BY organizerID, eventID
ORDER BY MAX(Created_At) DESC;
```

Data Output Messages Notifications

Showing rows: 1 to 9

	organizerid integer	eventid [PK] integer	complaints bigint
1	1	9	1
2	3	3	2
3	5	5	2
4	4	4	1
5	2	10	2
6	6	6	2
7	7	7	2
8	1	1	2
9	8	8	2

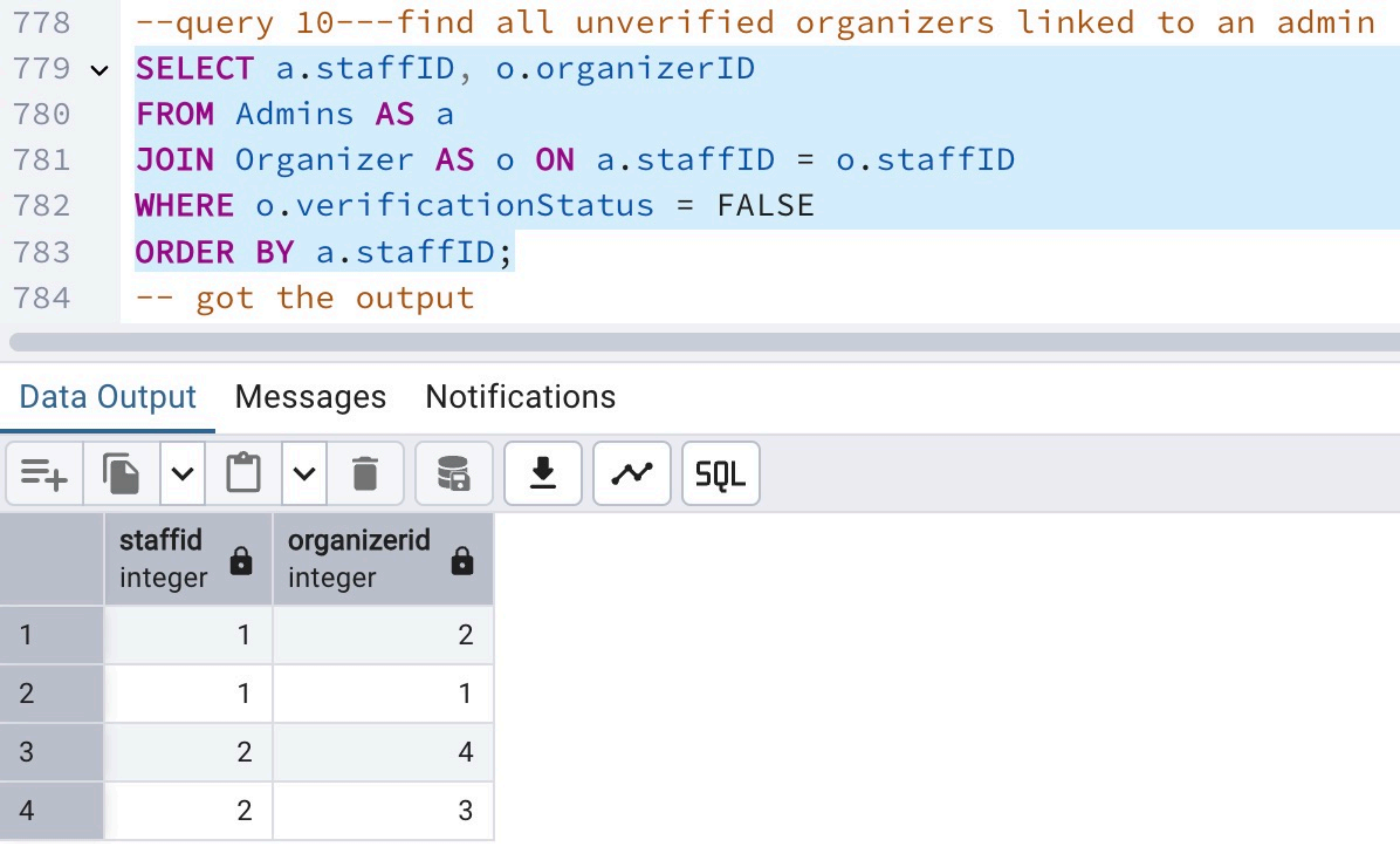


```
-- Query 9 - get the organisers having avg_rating of the events <= x or no. of complaints >= y
SELECT organizerID, username
FROM Organizer
WHERE 4.5 >= (SELECT avg_Rating
            FROM    (SELECT O.organizerID as orgID, avg(R.rating) as avg_Rating
                    FROM organizer as O
                    JOIN (SELECT E.organizerID as organizerID, F.Rating as rating
                          FROM events as E
                          JOIN feedback as F
                          ON E.eventID = F.eventID) as R
                    ON O.organizerID = R.organizerID
                    GROUP BY O.organizerID)
            WHERE orgID = organizerID)
    OR
    4 <=    (SELECT complaints
            FROM    (WITH pending_Complaints as
                    (SELECT E.eventID as eventID, E.organizerID as orgID, C.complaintID, C.Created_At
                    FROM events as E
                    JOIN complaint as C
                    ON E.eventID = C.eventID)
                SELECT orgID, count(*) as complaints
                FROM pending_Complaints
                GROUP BY orgID)
            WHERE organizerID = orgID);
-- got the output
```

Data Output Messages Notifications

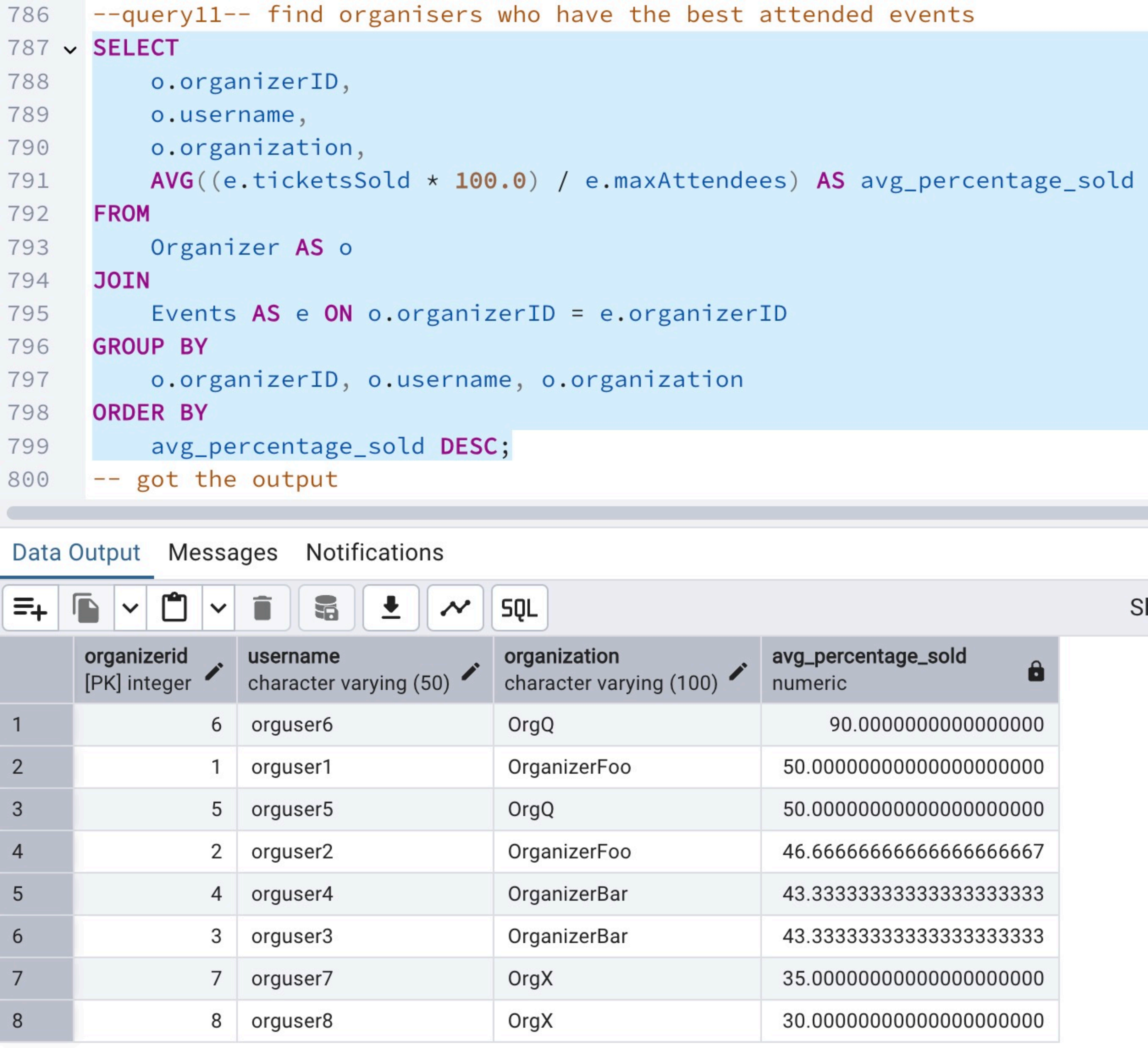
Showing rows: 1 to 6

	organizerid [PK] integer	username character varying (50)
1	1	orguser1
2	2	orguser2
3	4	orguser4
4	5	orguser5
5	7	orguser7
6	8	orguser8



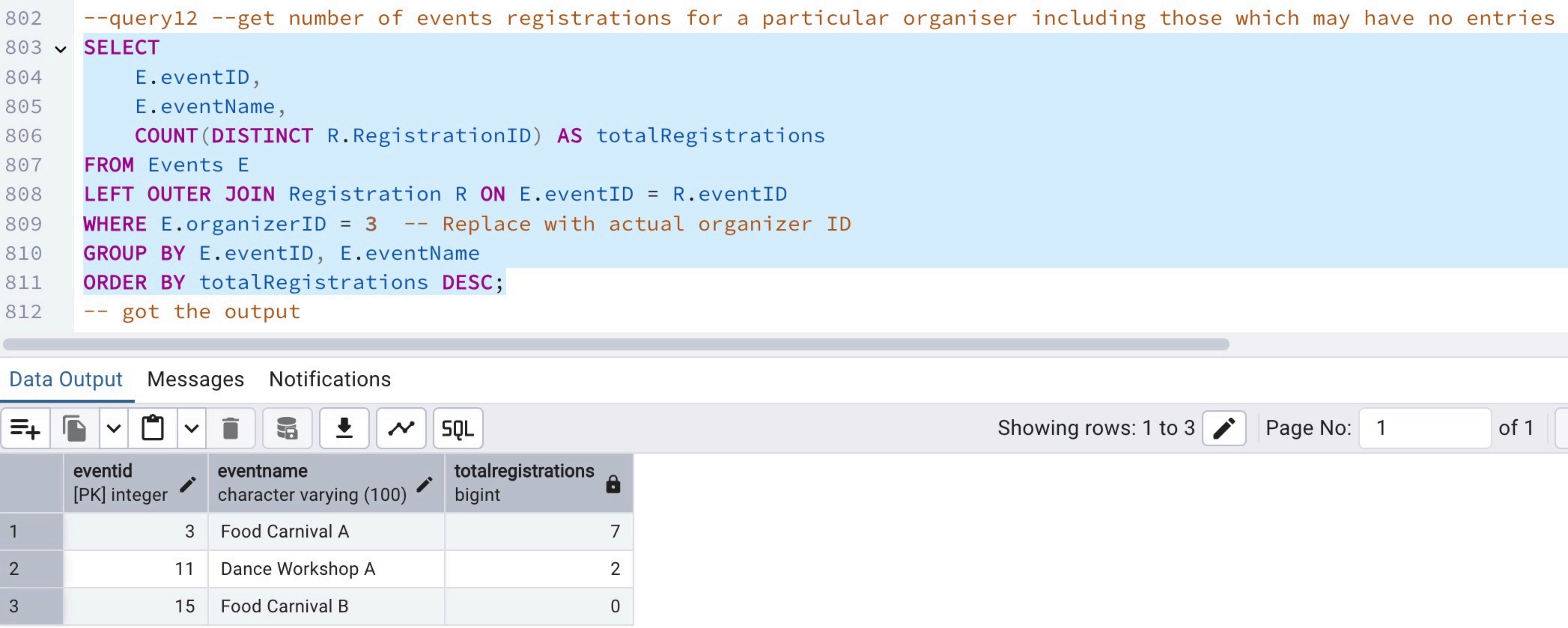
```sql
--query 10---find all unverified organizers linked to an admin
SELECT a.staffID, o.organizerID
FROM Admins AS a
JOIN Organizer AS o ON a.staffID = o.staffID
WHERE o.verificationStatus = FALSE
ORDER BY a.staffID;
-- got the output
```

	staffid integer	organizerid integer
1	1	2
2	1	1
3	2	4
4	2	3



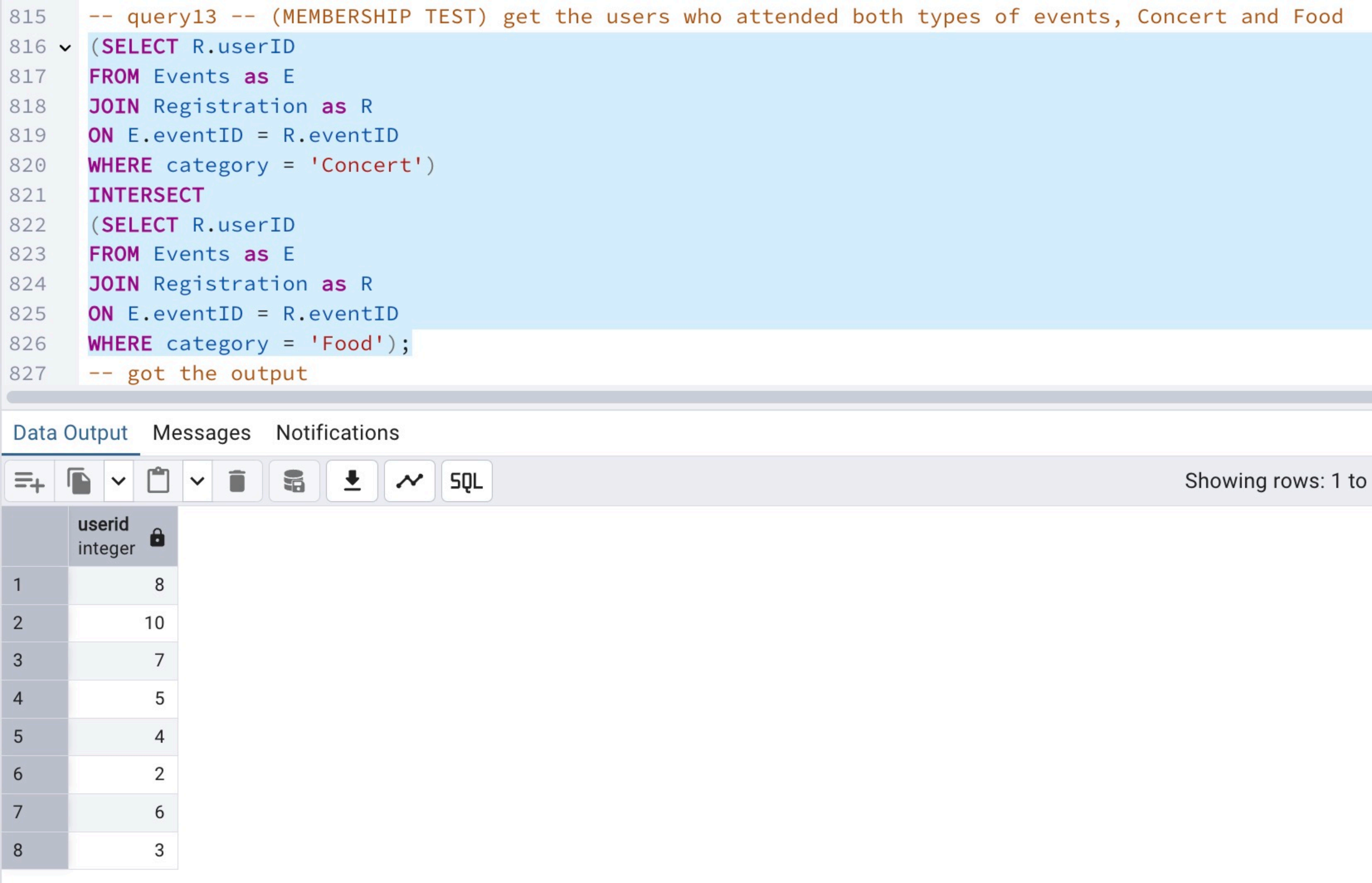
```sql
--query11-- find organisers who have the best attended events
SELECT
    o.organizerID,
    o.username,
    o.organization,
    AVG((e.ticketsSold * 100.0) / e.maxAttendees) AS avg_percentage_sold
FROM
    Organizer AS o
JOIN
    Events AS e ON o.organizerID = e.organizerID
GROUP BY
    o.organizerID, o.username, o.organization
ORDER BY
    avg_percentage_sold DESC;
-- got the output
```

	organizerid [PK] integer	username character varying (50)	organization character varying (100)	avg_percentage_sold numeric
1	6	orguser6	OrgQ	90.0000000000000000
2	1	orguser1	OrganizerFoo	50.0000000000000000000
3	5	orguser5	OrgQ	50.0000000000000000000
4	2	orguser2	OrganizerFoo	46.6666666666666666667
5	4	orguser4	OrganizerBar	43.3333333333333333333
6	3	orguser3	OrganizerBar	43.3333333333333333333
7	7	orguser7	OrgX	35.0000000000000000000
8	8	orguser8	OrgX	30.0000000000000000000



```
--query12 --get number of events registrations for a particular organiser including those which may have no entries
SELECT
    E.eventID,
    E.eventName,
    COUNT(DISTINCT R.RegistrationID) AS totalRegistrations
FROM Events E
LEFT OUTER JOIN Registration R ON E.eventID = R.eventID
WHERE E.organizerID = 3  -- Replace with actual organizer ID
GROUP BY E.eventID, E.eventName
ORDER BY totalRegistrations DESC;
-- got the output
```

	eventid [PK] integer	eventname character varying (100)	totalregistrations bigint
1	3	Food Carnival A	7
2	11	Dance Workshop A	2
3	15	Food Carnival B	0

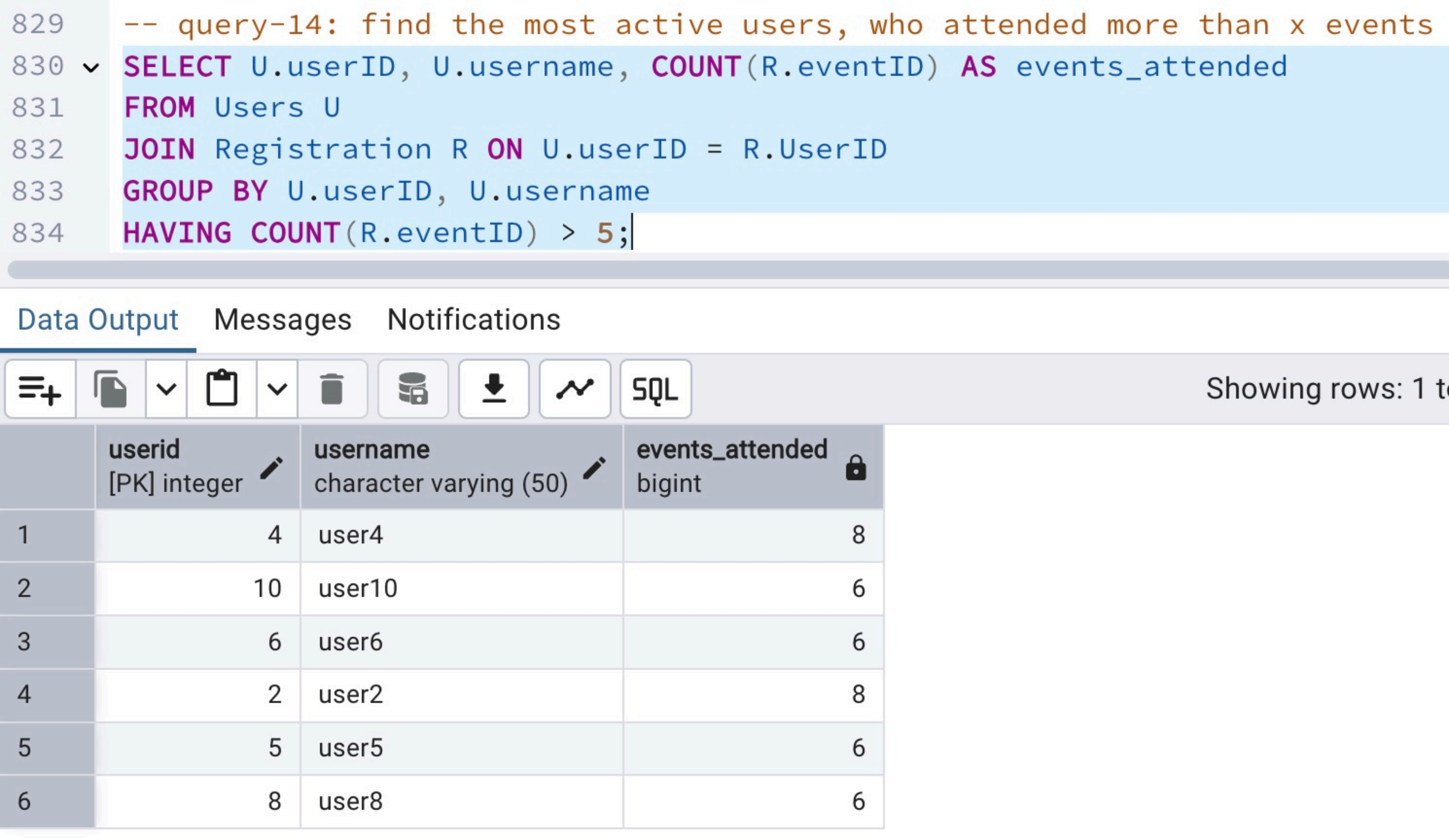


```
-- query13 -- (MEMBERSHIP TEST) get the users who attended both types of events, Concert and Food
(SELECT R.userID
FROM Events as E
JOIN Registration as R
ON E.eventID = R.eventID
WHERE category = 'Concert')
INTERSECT
(SELECT R.userID
FROM Events as E
JOIN Registration as R
ON E.eventID = R.eventID
WHERE category = 'Food');
-- got the output
```

Data Output Messages Notifications

Showing rows: 1 to

```
   userid
   integer
1        8
2       10
3        7
4        5
5        4
6        2
7        6
8        3
```



```
-- query-14: find the most active users, who attended more than x events
SELECT U.userID, U.username, COUNT(R.eventID) AS events_attended
FROM Users U
JOIN Registration R ON U.userID = R.UserID
GROUP BY U.userID, U.username
HAVING COUNT(R.eventID) > 5;
```

	userid [PK] integer	username character varying (50)	events_attended bigint
1	4	user4	8
2	10	user10	6
3	6	user6	6
4	2	user2	8
5	5	user5	6
6	8	user8	6