به‌نام آفریننده بیت‌ها

دانشکده‌ی برق و کامپیوتر

دانشگاه صنعتی اصفهان

نیم‌سال اول ۱۴۰۱ - ۱۴۰۲

پایگاه داده ۱

تمرین سوم

نام:
دانیال خراسانی‌زاده

شماره دانشجویی:
۹۹۲۲۳۹۳

استاد درس:
دکتر بصیری

فهرست مطالب

۱ ۲
۲ ۲
۱.۲ ۲
۲.۲ ۲
۳ ۳
۱.۳ ۳
۲.۳ ۴
۴ ۵
۵ ۶
۶ ۷
۷ ۸
۸ ۹
۹ ۱۳
۱۰ ۱۴

۱

حالت بیشینه زمانی رخ می‌دهد که یک کاربر ۱۰۰ شماره تلفن داشته باشد و برای بقیه کاربرها شماره تلفنی ثبت نشده باشد و حالت کمینه زمانی رخ می‌دهد که هر کاربر یک شماره تلفن داشته باشد.

	Minimum	Maximum
Inner Join	100	100
Left outer join	100	199
Right outer join	100	100
Full outer join	100	199

۲

۱.۲

این کوئری تمام گره‌هایی که حداقل یک یال به آن‌ها وارد شده اما هیچ یالی از آن‌ها خارج نشده است را باز می‌گرداند.

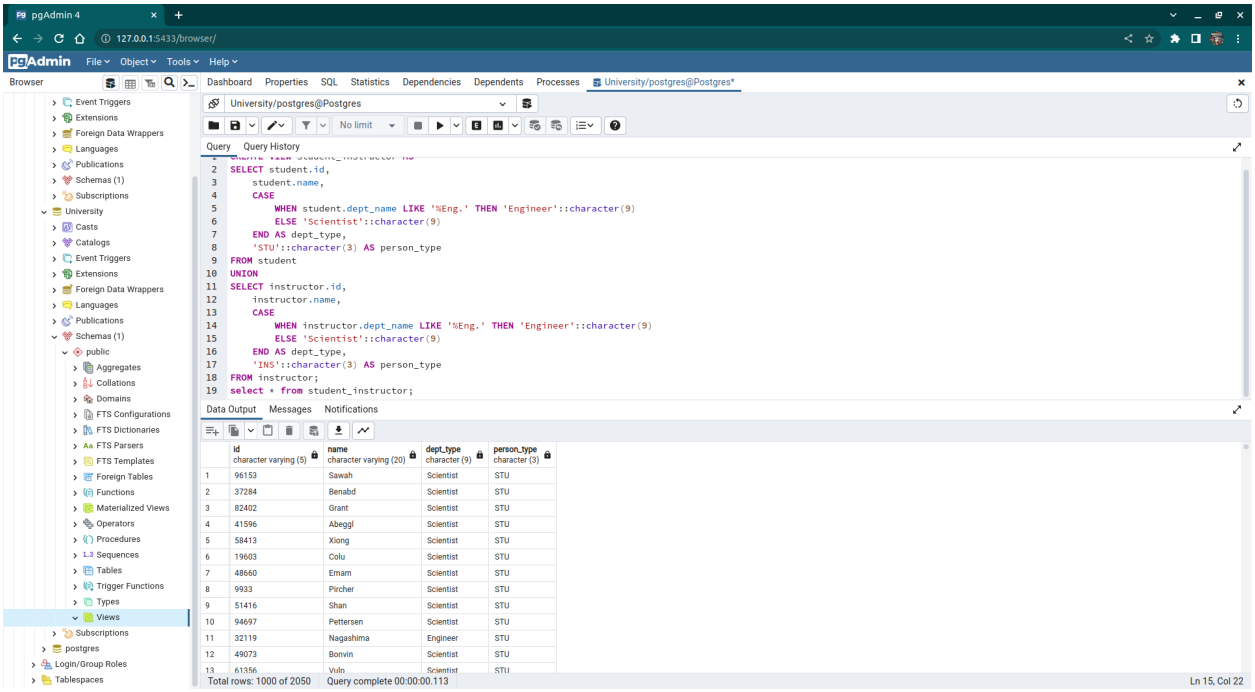
۲.۲

ساب کوئری اول گره‌های حاضر به همراه تعداد یال‌های خروجی از آن‌ها را پیدا می‌کند و ساب کوئری دوم گره‌های حاضر به همراه تعداد یال‌های ورودی به آن‌ها را باز می‌گرداند. با جوین این دو کوئری با شروط داده شده در نهایت لیست گره‌هایی که تعداد یال‌های خروجی از آن‌ها از تعداد یال‌های ورودی به آن‌ها کمتر است را در نتیجه خواهیم داشت.

۳

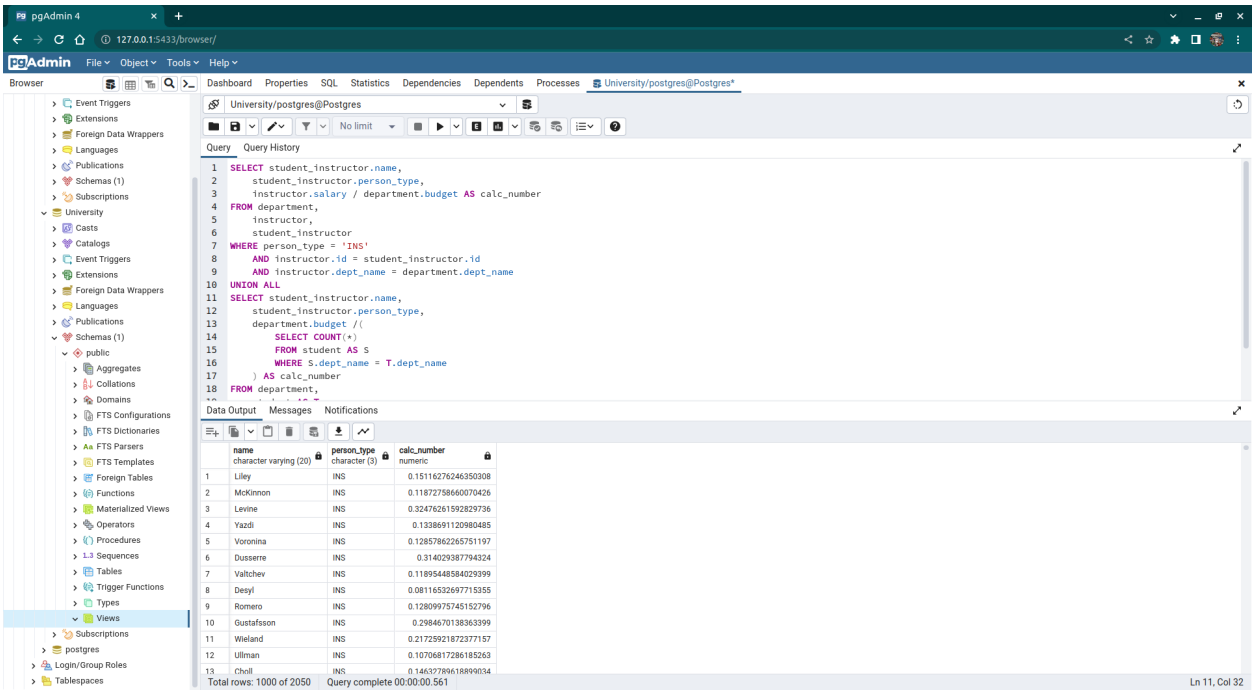
۱.۳

```sql
CREATE VIEW student_instructor AS
SELECT student.id,
   student.name,
   CASE
      WHEN student.dept_name LIKE '%Eng.' THEN 'Engineer'::character(9)
      ELSE 'Scientist'::character(9)
   END AS dept_type,
   'STU'::character(3) AS person_type
FROM student
UNION
SELECT instructor.id,
   instructor.name,
   CASE
      WHEN instructor.dept_name LIKE '%Eng.' THEN 'Engineer'::character(9)
      ELSE 'Scientist'::character(9)
   END AS dept_type,
   'INS'::character(3) AS person_type
FROM instructor;
```



۲.۳

```sql
SELECT student_instructor.name,
   student_instructor.person_type,
   instructor.salary / department.budget AS calc_number
FROM department,
   instructor,
   student_instructor
WHERE person_type = 'INS'
   AND instructor.id = student_instructor.id
   AND instructor.dept_name = department.dept_name
UNION ALL
SELECT student_instructor.name,
   student_instructor.person_type,
   department.budget /(
      SELECT COUNT(*)
      FROM student AS S
      WHERE S.dept_name = T.dept_name
   ) AS calc_number
FROM department,
   student AS T,
   student_instructor
WHERE student_instructor.person_type = 'STU'
   AND T.id = student_instructor.id
   AND T.dept_name = department.dept_name;
```

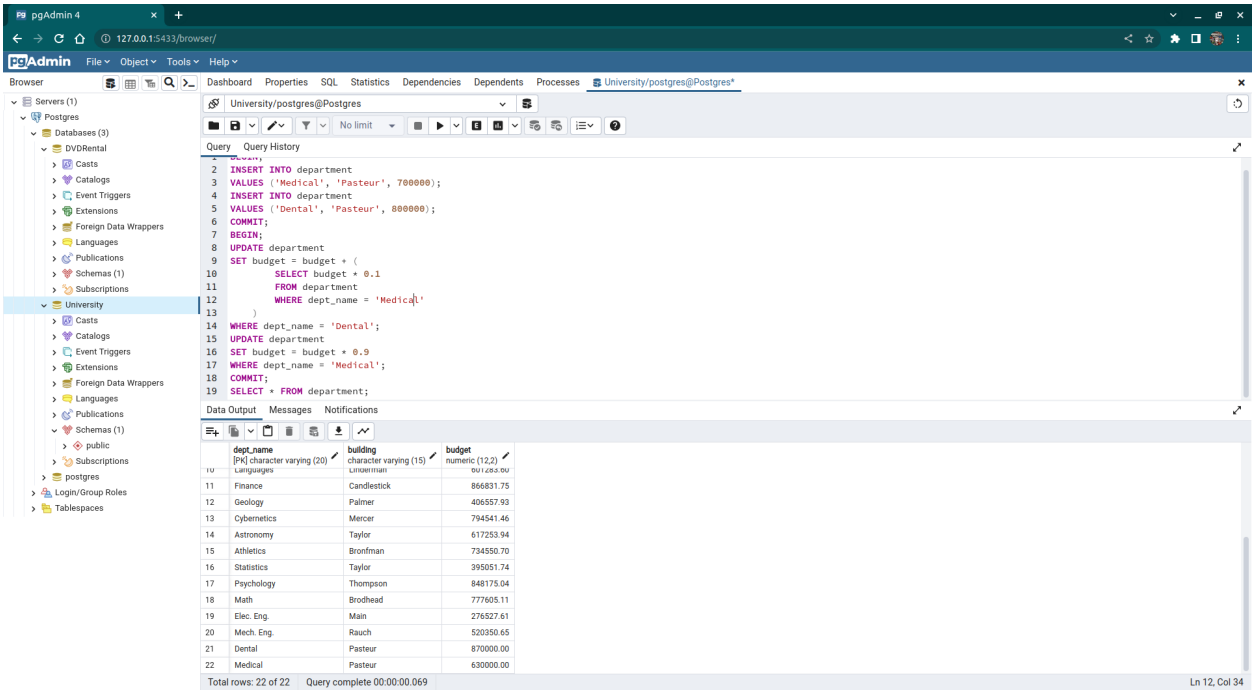


۴

```sql
ALTER TABLE film
ADD CHECK (length >= 50) NOT VALID;
ALTER TABLE payment
ADD pay_type VARCHAR(11) CHECK (pay_type in ('credit_card', 'cash', 'online'));
```

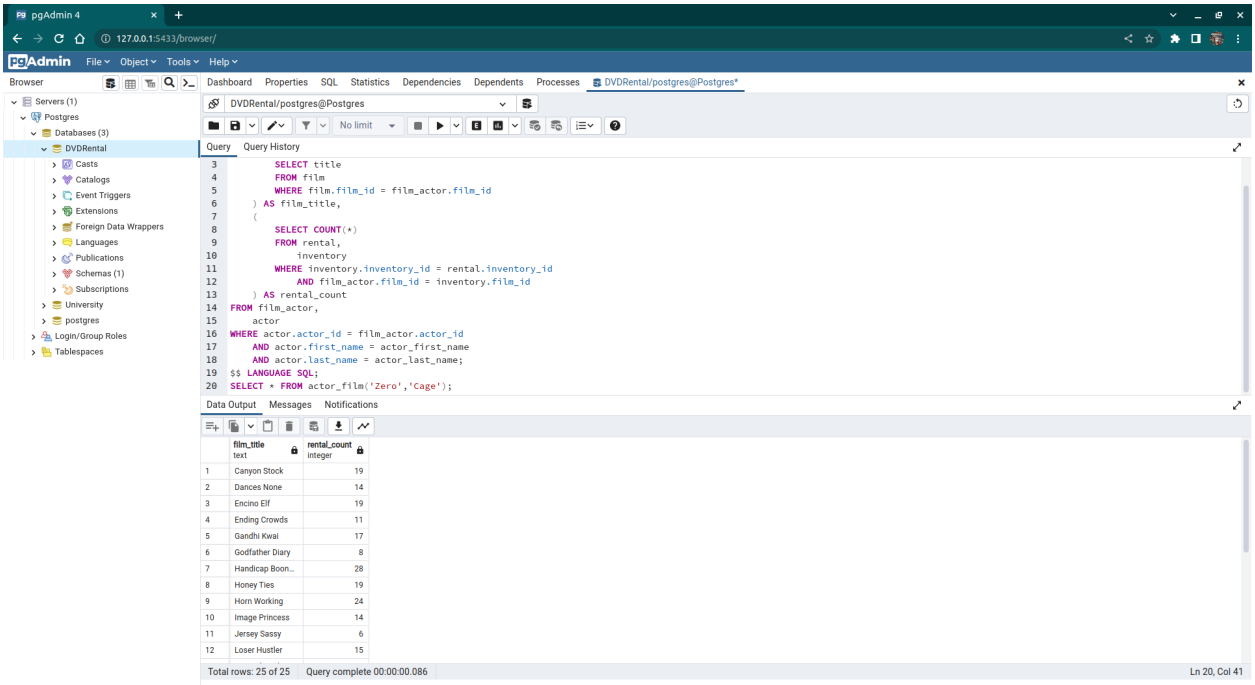
۵

```sql
BEGIN;
INSERT INTO department
VALUES ('Medical', 'Pasteur', 700000);
INSERT INTO department
VALUES ('Dental', 'Pasteur', 800000);
COMMIT;
BEGIN;
UPDATE department
SET budget = budget + (
        SELECT budget * 0.1
        FROM department
        WHERE dept_name = 'Medical'
    )
WHERE dept_name = 'Dental';
UPDATE department
SET budget = budget * 0.9
WHERE dept_name = 'Medical';
COMMIT;
```



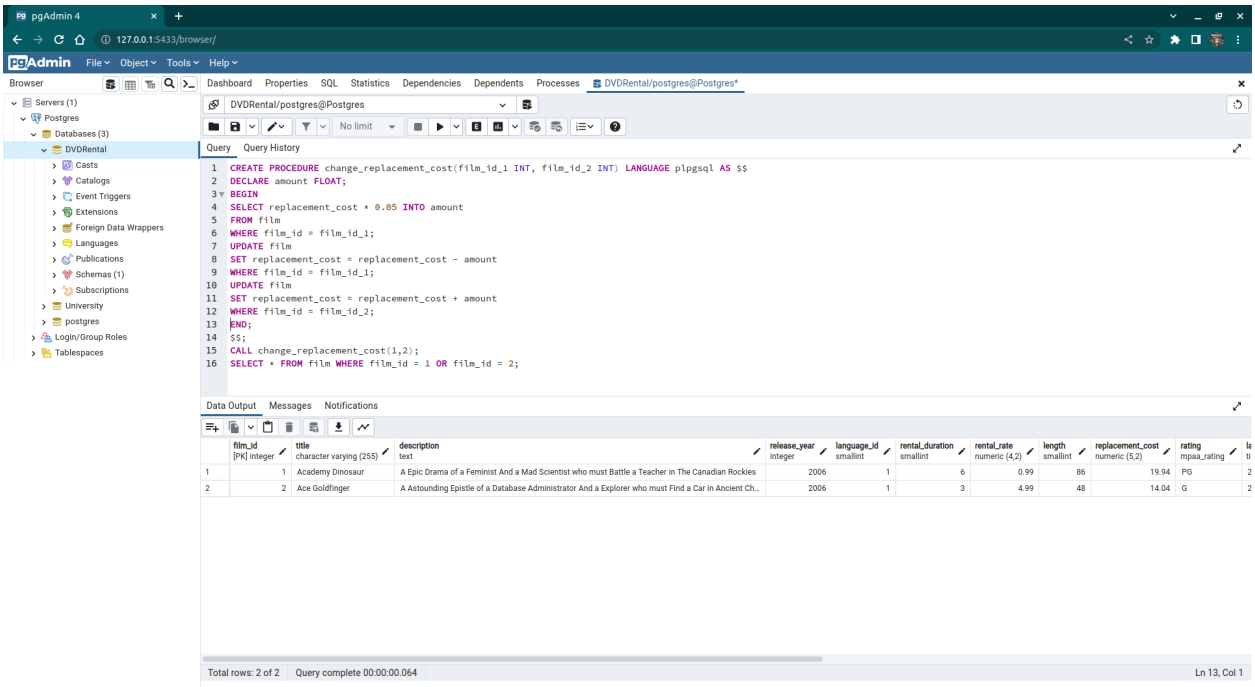
۶

```sql
CREATE FUNCTION actor_film(actor_first_name TEXT, actor_last_name TEXT) RETURNS
    TABLE(film_title TEXT, rental_count int) AS $$
SELECT (
      SELECT title
      FROM film
      WHERE film.film_id = film_actor.film_id
   ) AS film_title,
   (
      SELECT COUNT(*)
      FROM rental,
         inventory
      WHERE inventory.inventory_id = rental.inventory_id
         AND film_actor.film_id = inventory.film_id
   ) AS rental_count
FROM film_actor,
   actor
WHERE actor.actor_id = film_actor.actor_id
   AND actor.first_name = actor_first_name
   AND actor.last_name = actor_last_name;
$$ LANGUAGE SQL;
```



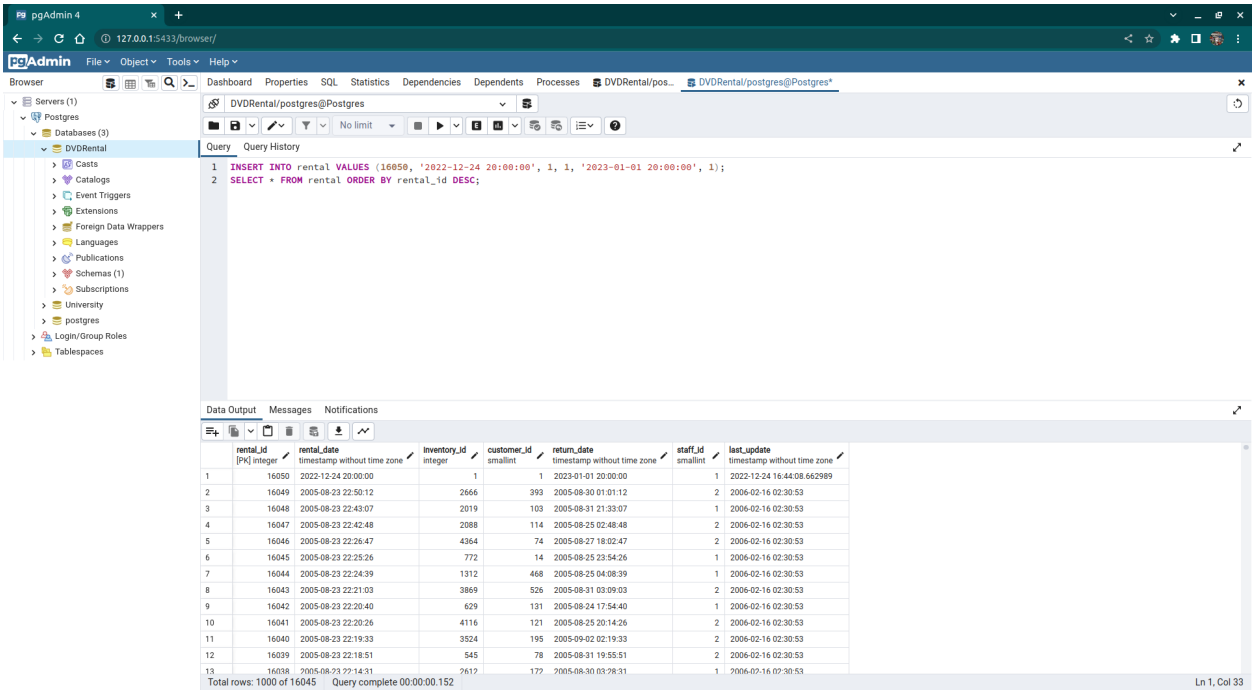
۷

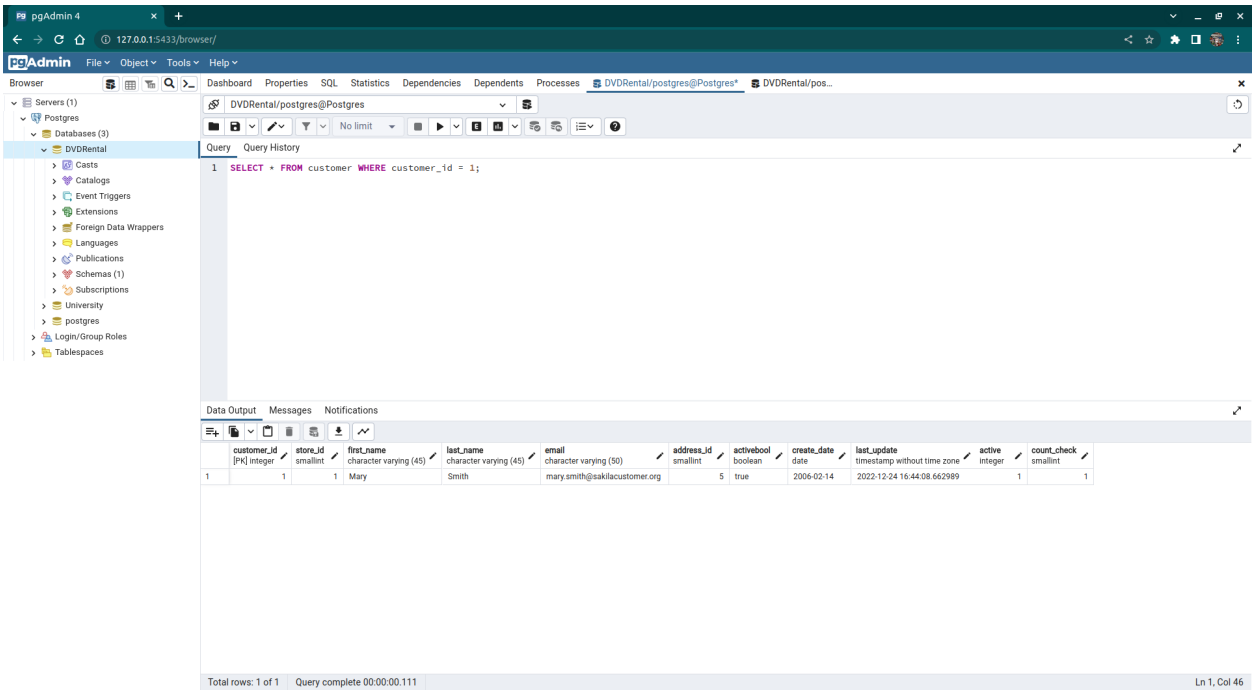
```sql
CREATE PROCEDURE change_replacement_cost(film_id_1 INT, film_id_2 INT) LANGUAGE plpgsql AS
    $$
DECLARE amount FLOAT;
BEGIN
SELECT replacement_cost * 0.05 INTO amount
FROM film
WHERE film_id = film_id_1;
UPDATE film
SET replacement_cost = replacement_cost - amount
WHERE film_id = film_id_1;
UPDATE film
SET replacement_cost = replacement_cost + amount
WHERE film_id = film_id_2;
END;
$$;
```

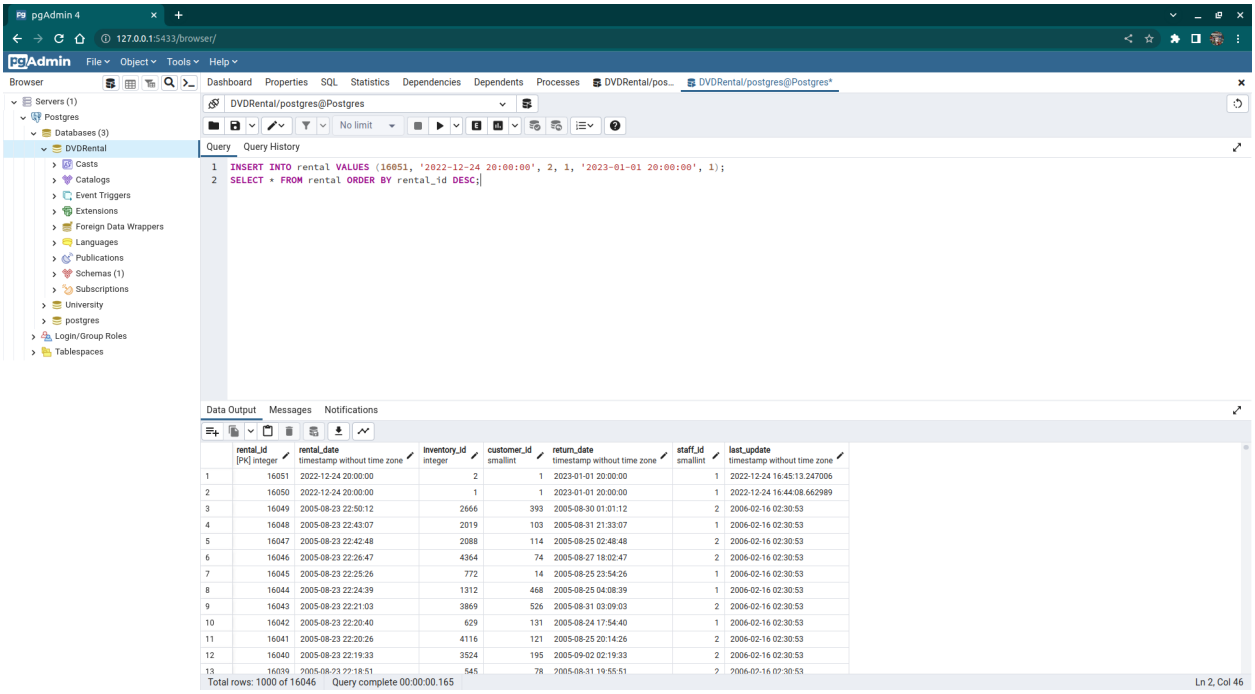


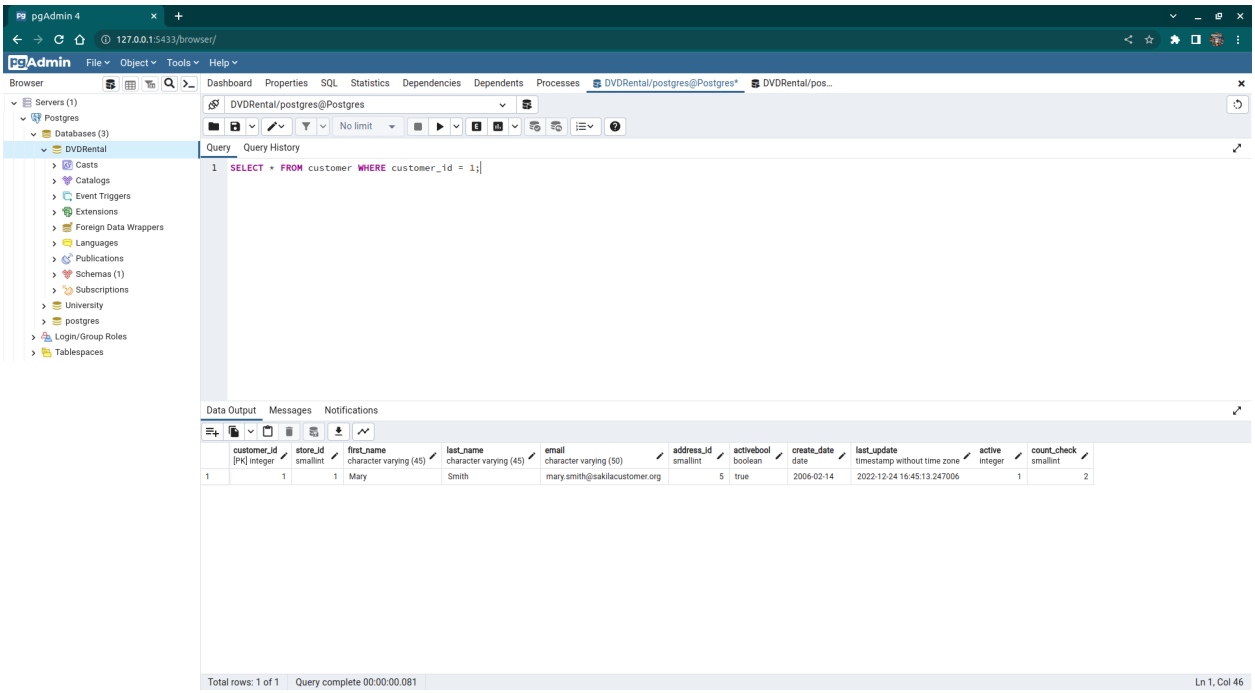
۸

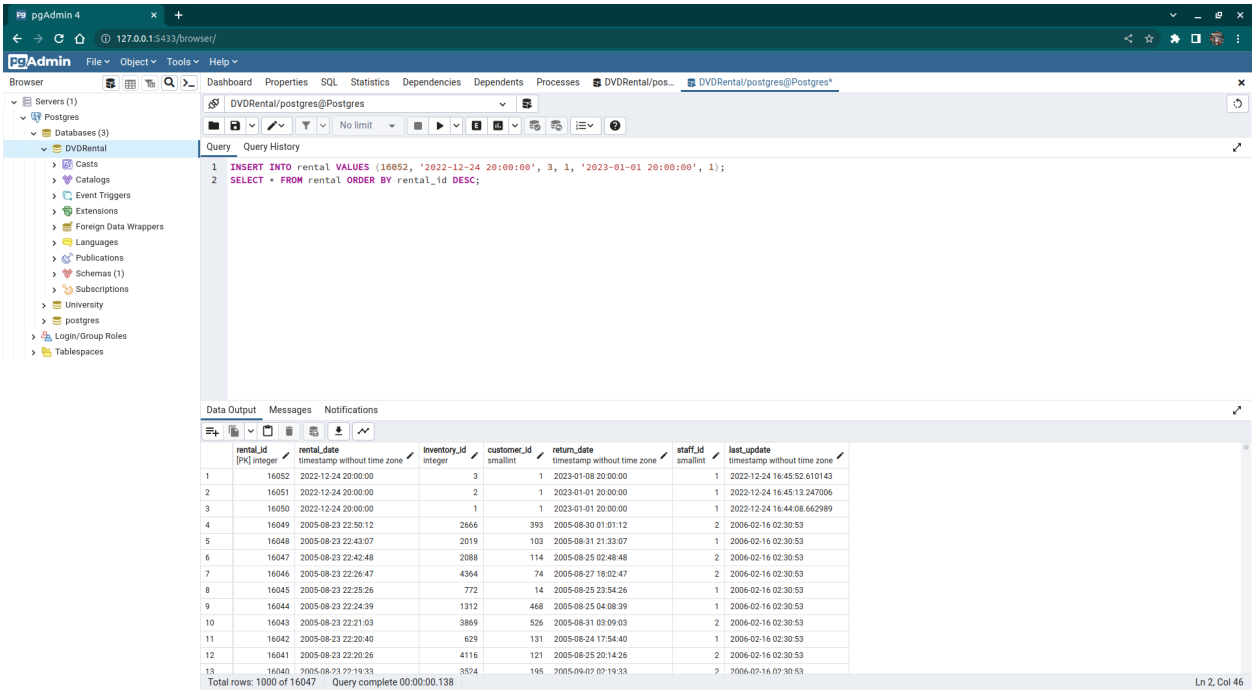
```
ALTER TABLE customer
ADD count_check SMALLINT;
UPDATE customer
SET count_check = 0;
CREATE OR Replace FUNCTION bonus_return_date() RETURNS trigger AS $$
DECLARE count smallint;
BEGIN
SELECT count_check INTO count
FROM customer
WHERE customer_id = NEW.customer_id;
IF count = 2 THEN
NEW.return_date := NEW.return_date + interval '7 day';
UPDATE customer
SET count_check = 0
WHERE customer_id = NEW.customer_id;
ELSE
UPDATE customer
SET count_check = count_check + 1
WHERE customer_id = NEW.customer_id;
END IF;
RETURN NEW;
END;
$$ LANGUAGE plpgsql;
CREATE OR REPLACE TRIGGER check_return_date BEFORE
INSERT ON rental FOR EACH ROW EXECUTE FUNCTION bonus_return_date();
```

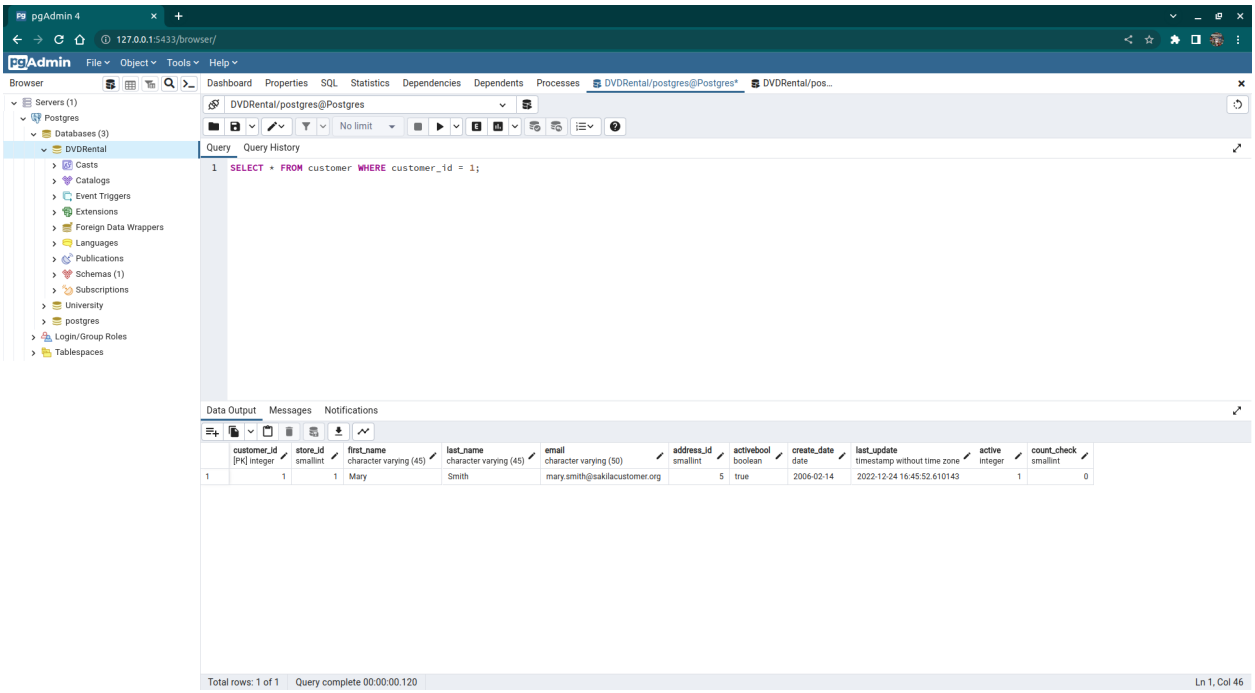

pgAdmin 4
127.0.0.1:5433/browser/
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents Processes DVDRental/pos... DVDRental/postgres@Postgres*
DVDRental/postgres@Postgres
No limit
Query Query History
1 INSERT INTO rental VALUES (16050, '2022-12-24 20:00:00', 1, 1, '2023-01-01 20:00:00', 1);
2 SELECT * FROM rental ORDER BY rental_id DESC;
Data Output Messages Notifications
rental_id [PK] integer
rental_date timestamp without time zone
inventory_id integer
customer_id smallint
return_date timestamp without time zone
staff_id smallint
last_update timestamp without time zone
1 16050 2022-12-24 20:00:00 1 1 2023-01-01 20:00:00 1 2022-12-24 16:44:08.662989
2 16049 2005-08-23 22:50:12 2666 393 2005-08-30 01:01:12 2 2006-02-16 02:30:53
3 16048 2005-08-23 22:43:07 2019 103 2005-08-31 21:33:07 1 2006-02-16 02:30:53
4 16047 2005-08-23 22:42:48 2088 114 2005-08-25 02:48:48 2 2006-02-16 02:30:53
5 16046 2005-08-23 22:26:47 4364 74 2005-08-27 18:02:47 2 2006-02-16 02:30:53
6 16045 2005-08-23 22:25:26 772 14 2005-08-25 23:54:26 1 2006-02-16 02:30:53
7 16044 2005-08-23 22:24:39 1312 468 2005-08-25 04:08:39 1 2006-02-16 02:30:53
8 16043 2005-08-23 22:21:03 3869 526 2005-08-31 03:09:03 2 2006-02-16 02:30:53
9 16042 2005-08-23 22:20:40 629 131 2005-08-24 17:54:40 1 2006-02-16 02:30:53
10 16041 2005-08-23 22:20:26 4116 121 2005-08-25 20:14:26 2 2006-02-16 02:30:53
11 16040 2005-08-23 22:19:33 3524 195 2005-09-02 02:19:33 2 2006-02-16 02:30:53
12 16039 2005-08-23 22:18:51 545 78 2005-08-31 19:55:51 2 2006-02-16 02:30:53
Total rows: 1000 of 16045
Query complete 00:00:00.152
Ln 1, Col 33


pgAdmin 4
127.0.0.1:5433/browser/
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents Processes DVDRental/postgres@Postgres* DVDRental/pos...
DVDRental/postgres@Postgres
No limit
Query Query History
1 SELECT * FROM customer WHERE customer_id = 1;
Data Output Messages Notifications
customer_id [PK] integer
store_id smallint
first_name character varying (45)
last_name character varying (45)
email character varying (50)
address_id smallint
activebool boolean
create_date date
last_update timestamp without time zone
active integer
count_check smallint
1 1 1 Mary Smith mary.smith@sakilacustomer.org 5 true 2006-02-14 2022-12-24 16:44:08.662989 1 1
Total rows: 1 of 1
Query complete 00:00:00.111
Ln 1, Col 46


pgAdmin 4
127.0.0.1:5433/browser/
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents Processes DVDRental/pos... DVDRental/postgres@Postgres*
DVDRental/postgres@Postgres
No limit
Query Query History
1 INSERT INTO rental VALUES (16051, '2022-12-24 20:00:00', 2, 1, '2023-01-01 20:00:00', 1);
2 SELECT * FROM rental ORDER BY rental_id DESC;
Data Output Messages Notifications
rental_id [PK] integer
rental_date timestamp without time zone
inventory_id integer
customer_id smallint
return_date timestamp without time zone
staff_id smallint
last_update timestamp without time zone
1 16051 2022-12-24 20:00:00 2 1 2023-01-01 20:00:00 1 2022-12-24 16:45:13.247006
2 16050 2022-12-24 20:00:00 1 1 2023-01-01 20:00:00 1 2022-12-24 16:44:08.662989
3 16049 2005-08-23 22:50:12 2666 393 2005-08-30 01:01:12 2 2006-02-16 02:30:53
4 16048 2005-08-23 22:43:07 2019 103 2005-08-31 21:33:07 1 2006-02-16 02:30:53
5 16047 2005-08-23 22:42:48 2088 114 2005-08-25 02:48:48 2 2006-02-16 02:30:53
6 16046 2005-08-23 22:26:47 4364 74 2005-08-27 18:02:47 2 2006-02-16 02:30:53
7 16045 2005-08-23 22:25:26 772 14 2005-08-25 23:54:26 1 2006-02-16 02:30:53
8 16044 2005-08-23 22:24:39 1312 468 2005-08-25 04:08:39 1 2006-02-16 02:30:53
9 16043 2005-08-23 22:21:03 3869 526 2005-08-31 03:09:03 2 2006-02-16 02:30:53
10 16042 2005-08-23 22:20:40 629 131 2005-08-24 17:54:40 1 2006-02-16 02:30:53
11 16041 2005-08-23 22:20:26 4116 121 2005-08-25 20:14:26 2 2006-02-16 02:30:53
12 16040 2005-08-23 22:19:33 3524 195 2005-09-02 02:19:33 2 2006-02-16 02:30:53
13 16039 2005-08-23 22:18:51 545 78 2005-08-31 19:55:51 2 2006-02-16 02:30:53
Total rows: 1000 of 16046 Query complete 00:00:00.165
Ln 2, Col 46

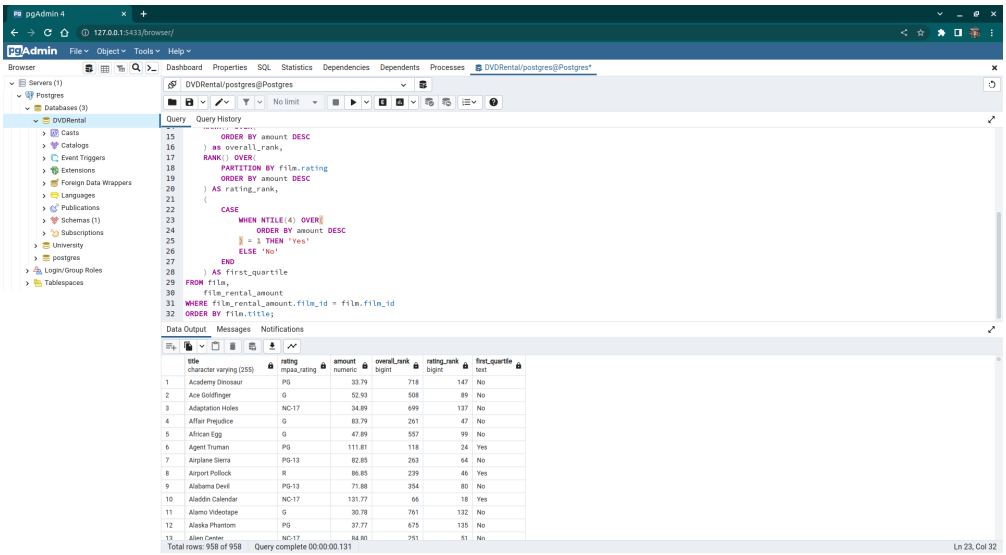

pgAdmin 4
127.0.0.1:5433/browser/
File Object Tools Help
Browser
Dashboard Properties SQL Statistics Dependencies Dependents Processes DVDRental/postgres@Postgres* DVDRental/pos...
DVDRental/postgres@Postgres
No limit
Query Query History
1 SELECT * FROM customer WHERE customer_id = 1;
Data Output Messages Notifications
customer_id [PK] integer
store_id smallint
first_name character varying (45)
last_name character varying (45)
email character varying (50)
address_id smallint
activebool boolean
create_date date
last_update timestamp without time zone
active integer
count_check smallint
1 1 1 Mary Smith mary.smith@sakilacustomer.org 5 true 2006-02-14 2022-12-24 16:45:13.247006 1 2
Total rows: 1 of 1 Query complete 00:00:00.081
Ln 1, Col 46


pgAdmin 4
127.0.0.1:5433/browser/
DVDRental/postgres@Postgres
INSERT INTO rental VALUES (16052, '2022-12-24 20:00:00', 3, 1, '2023-01-01 20:00:00', 1);
SELECT * FROM rental ORDER BY rental_id DESC;
Total rows: 1000 of 16047
Query complete 00:00:00.138
Ln 2, Col 46


pgAdmin 4
127.0.0.1:5433/browser/
DVDRental/postgres@Postgres
SELECT * FROM customer WHERE customer_id = 1;
Total rows: 1 of 1
Query complete 00:00:00.120
Ln 1, Col 46

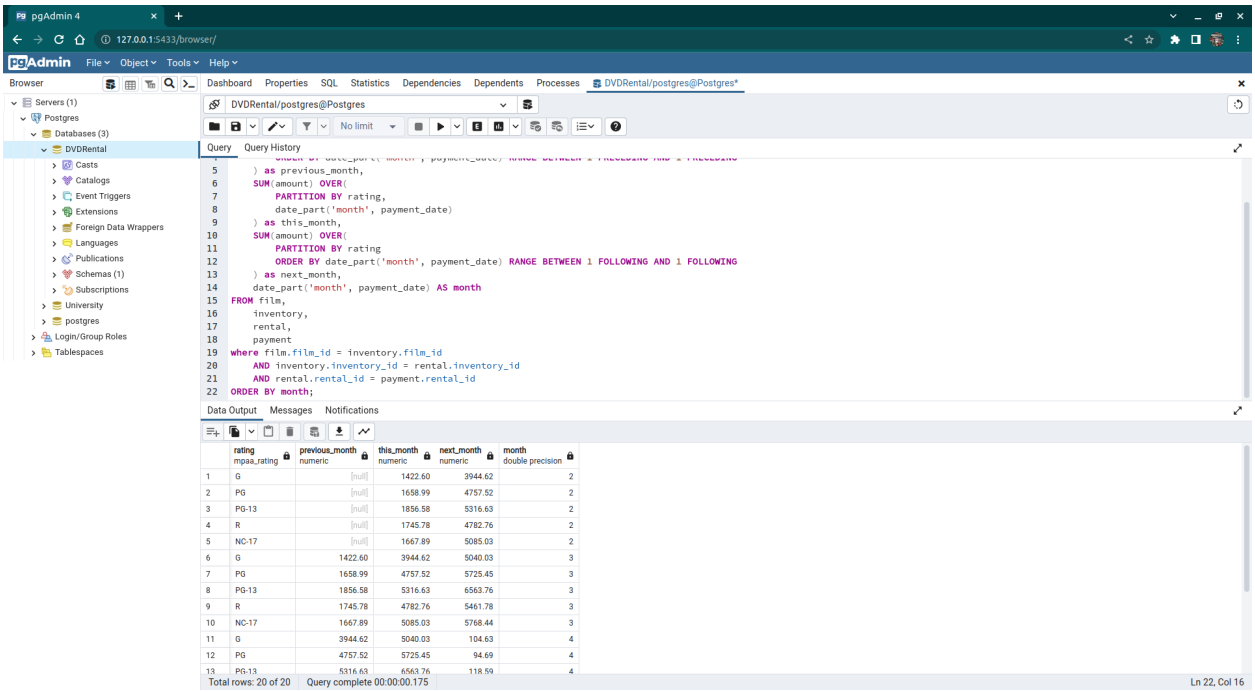
۹

```sql
WITH film_rental_amount AS (
    SELECT inventory.film_id AS film_id,
        SUM(amount) AS amount
    FROM payment,
        rental,
        inventory
    WHERE rental.rental_id = payment.rental_id
        AND inventory.inventory_id = rental.inventory_id
    GROUP BY inventory.film_id
)
SELECT title,
    rating,
    amount,
    RANK() OVER(ORDER BY amount DESC) AS overall_rank,
    RANK() OVER(PARTITION BY film.rating ORDER BY amount DESC) AS rating_rank,
    (
        CASE
            WHEN NTILE(4) OVER(
                ORDER BY amount DESC
            ) = 1 THEN 'Yes'
            ELSE 'No'
        END
    ) AS first_quartile
FROM film,
    film_rental_amount
WHERE film_rental_amount.film_id = film.film_id
ORDER BY film.title;
```



۱۰

```sql
SELECT DISTINCT rating,
    SUM(amount) OVER(
        PARTITION BY rating
        ORDER BY date_part('month', payment_date) RANGE BETWEEN 1 PRECEDING AND 1 PRECEDING
    ) as previous_month,
    SUM(amount) OVER(
        PARTITION BY rating,
        date_part('month', payment_date)
    ) as this_month,
    SUM(amount) OVER(
        PARTITION BY rating
        ORDER BY date_part('month', payment_date) RANGE BETWEEN 1 FOLLOWING AND 1 FOLLOWING
    ) as next_month,
    date_part('month', payment_date) AS month
FROM film,
    inventory,
    rental,
    payment
where film.film_id = inventory.film_id
    AND inventory.inventory_id = rental.inventory_id
    AND rental.rental_id = payment.rental_id
ORDER BY month;
```



	rating	previous_month	this_month	next_month	month
1	G	[null]	1422.60	3944.62	2
2	PG	[null]	1658.99	4757.52	2
3	PG-13	[null]	1856.58	5316.63	2
4	R	[null]	1745.78	4782.76	2
5	NC-17	[null]	1667.89	5085.03	2
6	G	1422.60	3944.62	5040.03	3
7	PG	1658.99	4757.52	5725.45	3
8	PG-13	1856.58	5316.63	6563.76	3
9	R	1745.78	4782.76	5461.78	3
10	NC-17	1667.89	5085.03	5768.44	3
11	G	3944.62	5040.03	104.63	4
12	PG	4757.52	5725.45	94.69	4
13	PG-13	5316.63	6563.76	118.59	4