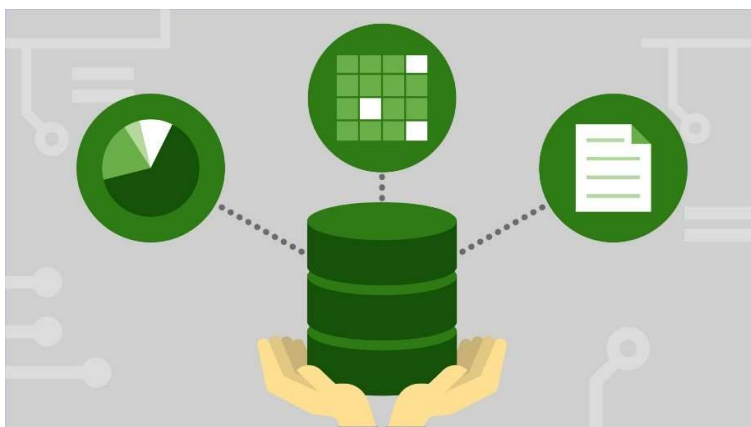


به نام خدا



دانشگاه تهران
پردیس دانشکده‌های فنی
دانشکده برق و کامپیوتر



آزمایشگاه پایگاه داده

دستور کار شماره ۷

پرنیان فاضل

۸۱۰۱۹۸۵۱۶

بهار ۱۴۰۲

مقدمه

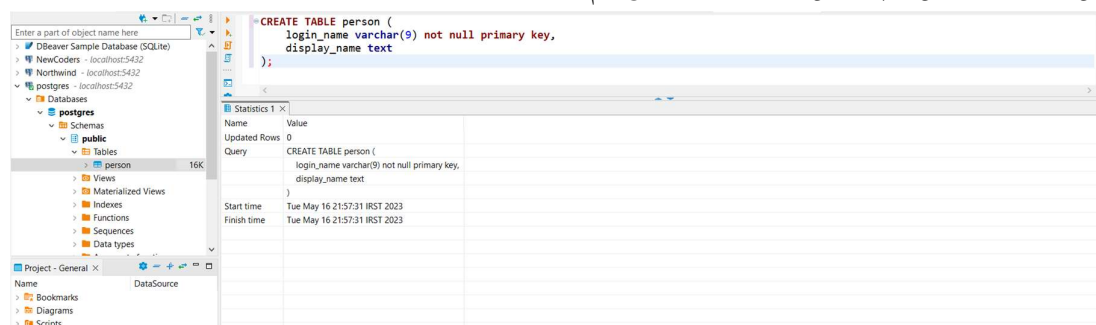
در این دستورکار به بررسی دستورات [این سایت](#)^۱ می‌پردازیم. عکس خروجی و توضیحی کوتاه درباره دستورات ارائه شده است.

توابع در SQL بلوک‌های کد قابل استفاده مجدد هستند که عملیات خاصی را انجام می‌دهند و مقداری را برمی‌گردانند. می‌توان از آنها برای محاسبات پیچیده یا دستکاری داده‌ها استفاده کرد و راهی مختصر برای انجام کارهای تکراری ارائه کرد. توابع می‌توانند پارامترهای ورودی را بپذیرند و می‌توانند یک مقدار یا جدولی از مقادیر را برگردانند. آنها ماژولار بودن و قابلیت استفاده مجدد کد SQL را افزایش می‌دهند و نگهداری و درک آن را آسان تر می‌کنند. توابع را می‌توان در دستورات SQL فراخوانی کرد که امکان پردازش کارآمد و کارآمد داده را فراهم می‌کند.

تریگرها در SQL اشیاء پایگاه داده هستند که به طور خودکار در پاسخ به رویدادها یا اقدامات خاصی روی یک جدول مانند درج، به روز رسانی یا حذف داده‌ها اجرا می‌شوند. آنها راهی برای اعمال یکپارچگی داده‌ها، انجام اعتبارسنجی های اضافی، یا راه اندازی سایر اقدامات بر اساس وقوع رویدادهای از پیش تعریف شده ارائه می‌دهند. تریگرها شامل رویداد ماشه‌ای هستند که مشخص می‌کند تریگر چه زمانی باید شلیک شود و اقدام ماشه‌ای که کدی را که باید هنگام فعال شدن تریگر اجرا شود را مشخص می‌کند. تریگرها می‌توانند برای اجرای قوانین تجاری، حفظ مسیرهای حسابرسی، یا اجرای منطق پیچیده مرتبط با داده‌ها در خود پایگاه داده مفید باشند.

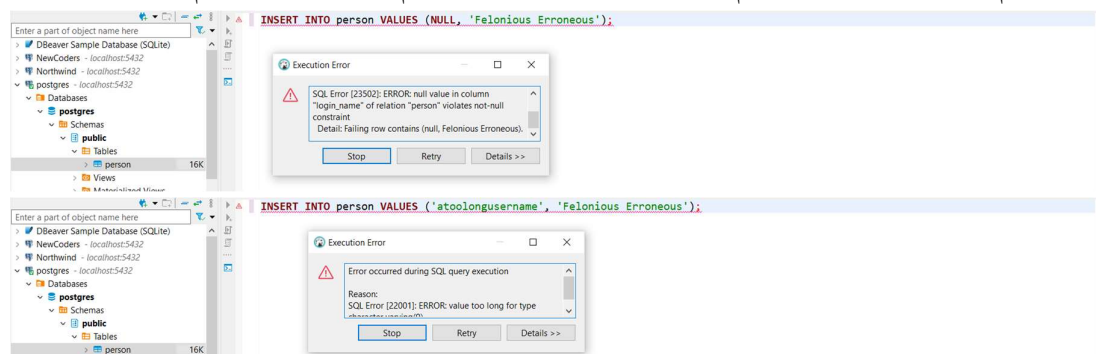
بخش Show Me Some Code Already

پس از ساخت دیتابیس در پستگرس، دستورات را اجرا می‌کنیم:



با استفاده از این دستور یک جدول به نام Person ایجاد می‌کنیم.

حال یک اسم با مقدار NULL و یک بار هم با طول بیشتر از ۹ وارد دیتابیس می‌کنیم و با ارور مواجه می‌شویم:



^۱ <https://severalnines.com/blog/postgresql-triggers-and-stored-function-basics>

حال به صورت constraint هایی را برای این جدول می گذاریم:

The screenshot shows the DBeaver SQL editor with two queries. The first query adds a non-null constraint and a length check to the 'login_name' column. The second query adds a check constraint to ensure no spaces are present in the 'login_name' column.

```

ALTER TABLE person
ADD CONSTRAINT PERSON_LOGIN_NAME_NON_NULL
CHECK (LENGTH(login_name) > 0);

ALTER TABLE person
ADD CONSTRAINT person_login_name_no_space
CHECK (POSITION(' ' IN login_name) = 0);

```

Below each query, the 'Statistics 1' tab shows the execution details, including the query text, updated rows (0), start time, and finish time.

حال با وارد کردن مقادیر اشتباه در دیتابیس از ورهای مربوط به این constraint ها را مشاهده می کنیم:

The screenshot shows two failed INSERT queries. The first query attempts to insert a value with a space, and the second query attempts to insert a value with a space and a comma. Both queries fail due to the constraints defined in the previous steps.

```

INSERT INTO person VALUES ('', 'Felonious Eroneous');

INSERT INTO person VALUES ('space man', 'Major Tom');

```

Below each query, the 'Execution Error' dialog box displays the error message: "SQL Error [23514]: ERROR: new row for relation 'person' violates check constraint 'person_login_name_non_null' (or 'person_login_name_no_space'). Detail: Failing row contains ('', 'Felonious Eroneous')." (or "space man", "Major Tom")."

حال constraint هایی که تعریف کردیم را حذف می کنیم:

The screenshot shows the DBeaver SQL editor with two queries to drop the constraints. The first query drops the 'person_login_name_no_space' constraint, and the second query drops the 'person_login_name_non_null' constraint.

```

ALTER TABLE PERSON DROP CONSTRAINT person_login_name_no_space;

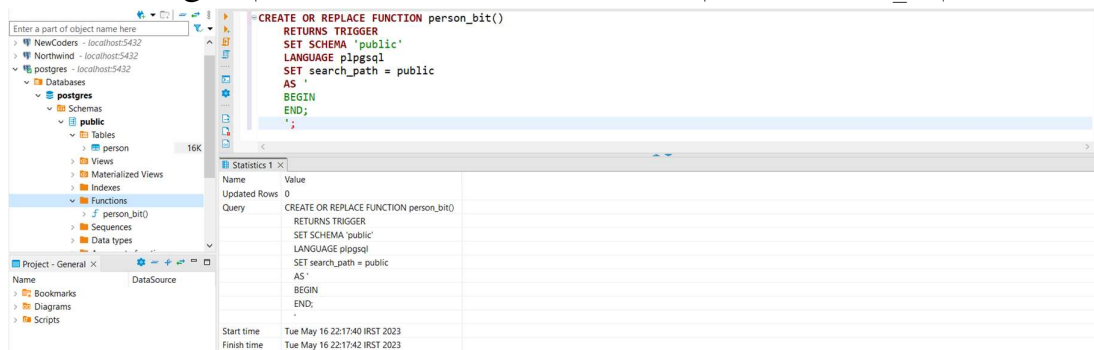
ALTER TABLE PERSON DROP CONSTRAINT person_login_name_non_null;

```

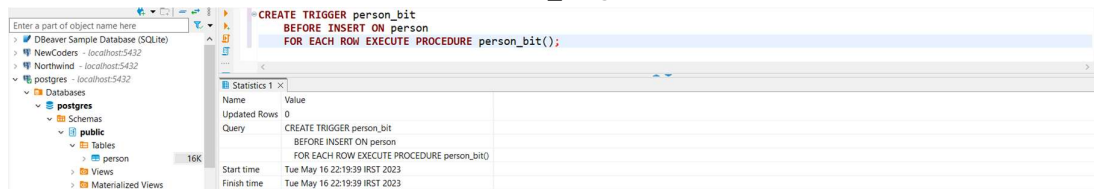
Below each query, the 'Statistics 1' tab shows the execution details, including the query text, updated rows (0), start time, and finish time.

بخش Show Me Some More Code

در اینجا تابعی به نام `person_bit` ایجاد می‌کنیم و همانطور که دیده می‌شود در قسمت Function هم این تابع قابل مشاهده است:

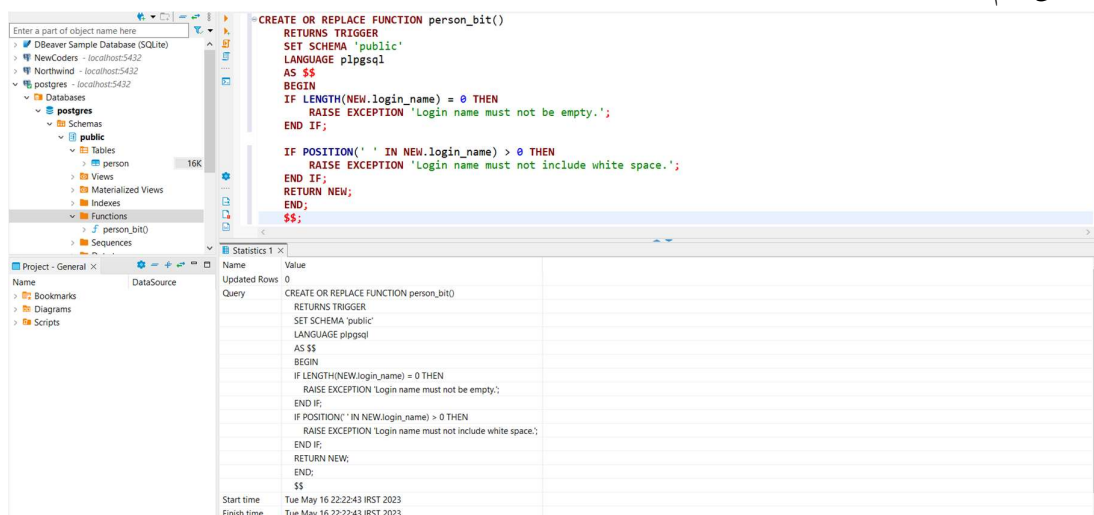


در اینجا یک trigger ایجاد می‌کنیم که جدول `person` و تابع `person_bit` که تعریف کردیم را به همدیگر مرتبط می‌کند:



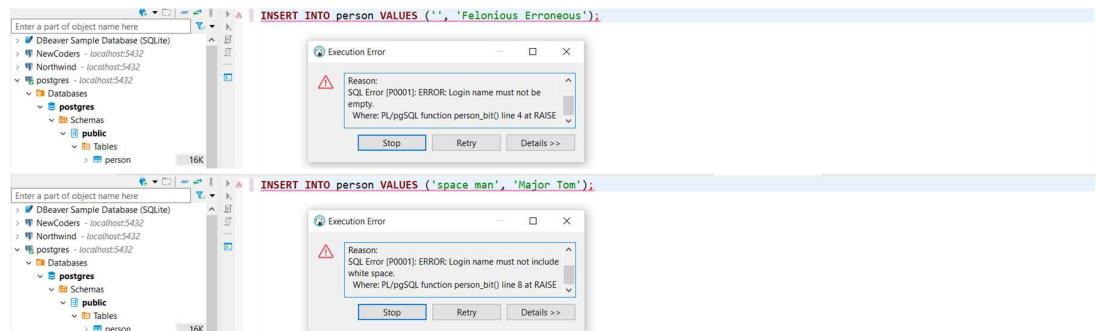
بخش EXAMPLE 0 – Data Validation

در اینجا، مواردی که در قسمت‌های قبل به عنوان constraint تعریف کردیم که پیام‌های با محتوای کاملی نبودند را با متنی خواناتر نشان می‌دهیم:



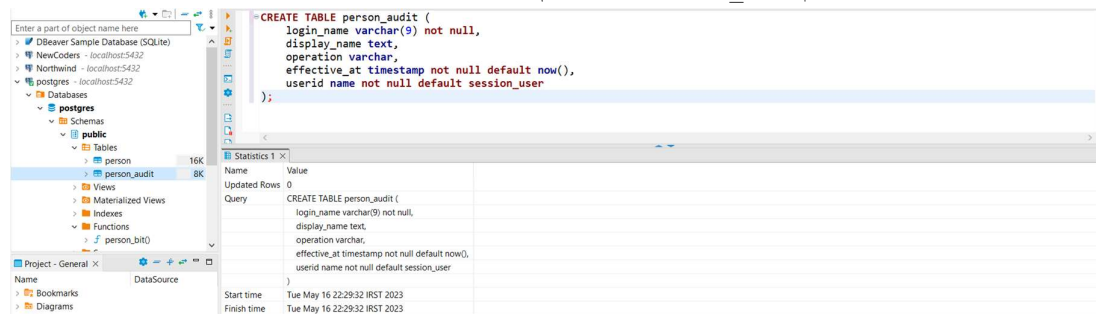
در این کد، "NEW" اشاره‌گری به ردیف داده‌هایی است که قرار است insert شوند. این یکی از تعدادی از متغیرهای ویژه موجود در یک تابع trigger است. در ادامه تعدادی دیگر را معرفی می‌کنیم.

حال به درج مواردی که در محدودیت‌های ما هستند می‌پردازیم. مشاهده می‌کنیم که متن خطاها بسیار خواناتر شده است و مطابق انتظار ما است:

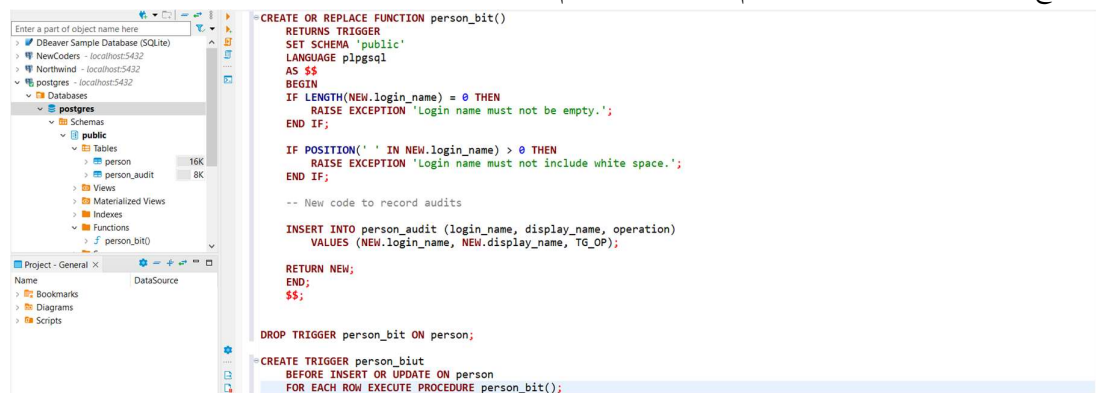


بخش Audit Logging – EXAMPLE 1

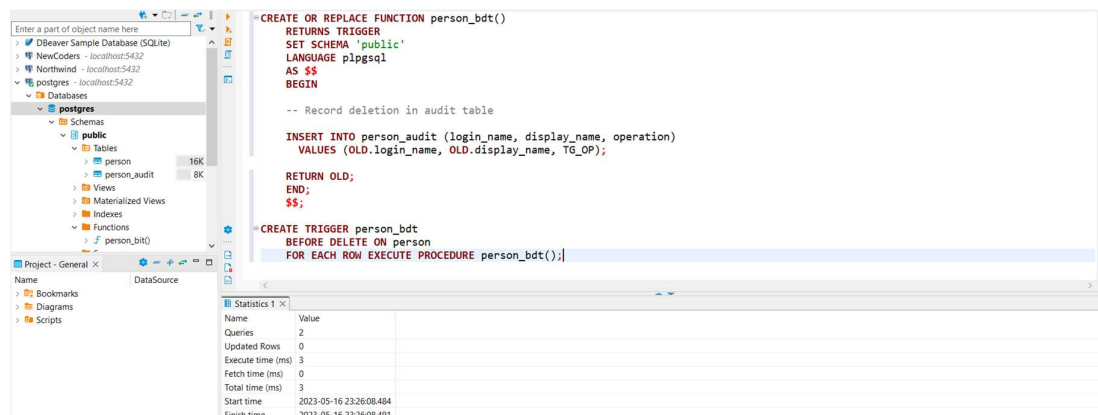
در ابتدای این بخش یک جدول به نام `person_audit` ایجاد می‌کنیم:



حال تابع `trigger` که قبلاً تعریف کرده بودیم را کمی تغییر می‌دهیم:



توجه شود که از متغیر "TG_OP" برای شناسایی عملیات DML که تریگر را به ترتیب به عنوان "INSERT"، "UPDATE"، "DELETE" و "TRUNCATE" فعال می‌کند، تنظیم می‌کند.



```

CREATE OR REPLACE FUNCTION person_bdt()
RETURNS TRIGGER
SET SCHEMA 'public'
LANGUAGE plpgsql
AS $$
BEGIN
    -- Record deletion in audit table
    INSERT INTO person_audit (login_name, display_name, operation)
    VALUES (OLD.login_name, OLD.display_name, TG_OP);

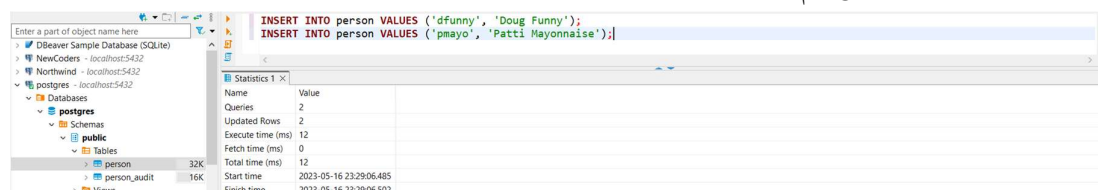
    RETURN OLD;
END;
$$;

CREATE TRIGGER person_bdt
BEFORE DELETE ON person
FOR EACH ROW EXECUTE PROCEDURE person_bdt();

```

در اینجا یک تابع trigger به نام "person_bdt" ایجاد می‌کنیم که قبل از حذف یک ردیف از جدول "person" اجرا می‌شود. تابع trigger یک رکورد را با اطلاعات مربوط به ردیف حذف شده در جدول "person_audit" وارد می‌کند و سپس ردیف حذف شده را برمی‌گرداند.

حال در اینجا ۲ رکورد اضافه می‌کنیم:



```

INSERT INTO person VALUES ('dfunny', 'Doug Funny');
INSERT INTO person VALUES ('pmayo', 'Patti Mayonnaise');

```

مشاهده می‌کنیم که داده‌ها به درستی در دیتابیس قرار گرفته‌اند:



login_name	display_name
dfunny	Doug Funny
pmayo	Patti Mayonnaise

حال یکی از ردیف‌ها را update می‌کنیم:

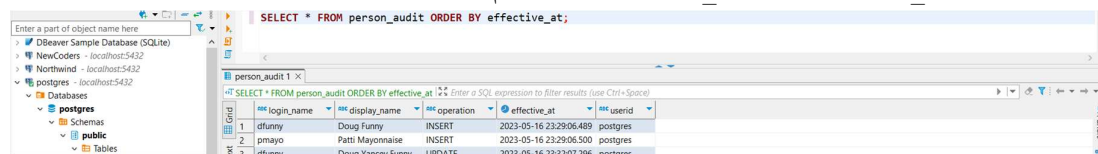


```

UPDATE person SET display_name = 'Doug Yancey Funny' WHERE login_name = 'dfunny';

```

داده‌های جدول person_audit را به ترتیب effective_at نشان می‌دهیم:



```

SELECT * FROM person_audit ORDER BY effective_at;

```

حال یکی از رکوردها را حذف می‌کنیم:



```

DELETE FROM person WHERE login_name = 'pmayo';

```


داده‌های جدول `person_audit` را به ترتیب `effective_at` نشان می‌دهیم و می‌بینیم که به درستی نشان داده می‌شود:

id	login_name	display_name	operation	effective_at	ec_userid
1	dfunny	Doug Funny	INSERT	2023-05-16 23:29:06.489	postgres
2	pmayo	Patti Mayonnaise	INSERT	2023-05-16 23:29:06.500	postgres
3	dfunny	Doug Yancey Funny	UPDATE	2023-05-16 23:32:07.296	postgres
4	pmayo	Patti Mayonnaise	DELETE	2023-05-16 23:37:47.658	postgres

بخش EXAMPLE 2 – Derived Values

در ابتدا دو ویژگی برای پشتیبانی از storage به جدول اصلی اضافه می‌کنیم:

```
ALTER TABLE person ADD COLUMN abstract TEXT;
ALTER TABLE person ADD COLUMN ts_abstract TSVECTOR;

ALTER TABLE person_audit ADD COLUMN abstract TEXT;
```

Name	Value
Queries	3
Updated Rows	0
Execute time (ms)	3
Fetch time (ms)	0
Total time (ms)	3
Start time	2023-05-16 23:40:10.001
Finish time	2023-05-16 23:40:10.008

حال تابع trigger را طوری تغییر می‌دهیم که attribute‌های جدیدی که اضافه کردیم هم پردازش شوند:

```
CREATE OR REPLACE FUNCTION person_bit()
RETURNS TRIGGER
LANGUAGE plpgsql
SET SCHEMA 'public'
AS $$
BEGIN
    IF LENGTH(NEW.login_name) = 0 THEN
        RAISE EXCEPTION 'Login name must not be empty.';
    END IF;

    IF POSITION(' ' IN NEW.login_name) > 0 THEN
        RAISE EXCEPTION 'Login name must not include white space.';
    END IF;

    -- Modified audit code to include text abstract
    INSERT INTO person_audit (login_name, display_name, operation, abstract)
    VALUES (NEW.login_name, NEW.display_name, TG_OP, NEW.abstract);

    -- New code to reduce text to text-search vector
    SELECT to_tsvector(NEW.abstract) INTO NEW.ts_abstract;

    RETURN NEW;
END;
$$;
```

Name	Value
Query	CREATE OR REPLACE FUNCTION person_bit() RETURNS TRIGGER LANGUAGE plpgsql SET SCHEMA 'public' AS \$\$ BEGIN IF LENGTH(NEW.login_name) = 0 THEN
Updated Rows	0

در اینجا به عنوان تست یک ردیف را به روز رسانی می‌کنیم:

```
UPDATE person
SET abstract = 'Doug is depicted as an introverted, quiet, insecure and gullible 11 (later 12) year old boy who wants to fit in v'
WHERE login_name = 'dfunny';
```

Name	Value
Query	UPDATE person SET abstract = 'Doug is depicted as an introverted, quiet, insecure and gullible 11 (later 12) year old boy who wants to fit in with the crowd.' WHERE login_name = 'dfunny'
Updated Rows	1
Start time	Tue May 16 23:46:48 IRST 2023
Finish time	Tue May 16 23:46:48 IRST 2023

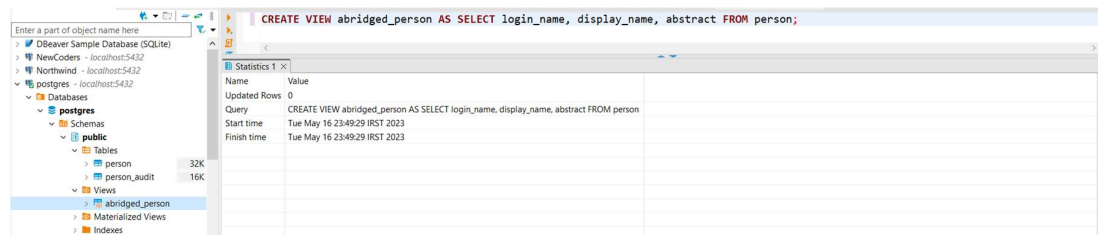
مشاهده می‌کنیم که اطلاعات به درستی ست شده اند:

```
SELECT login_name, ts_abstract FROM person;
```

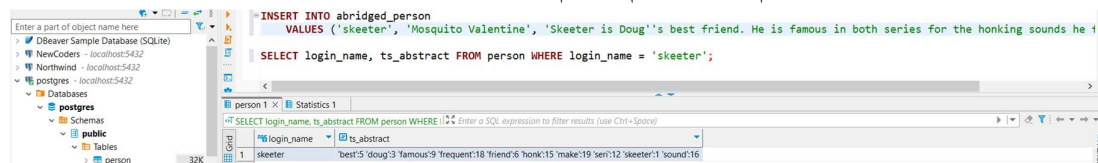
id	login_name	ts_abstract
1	dfunny	'11:11 12:13 boy:16 crowd:24 depict:3 doug:1 fit:20 gullible:10 insecure:8 introvert:6 later:12 old:15 quiet:7 want:18 year:14

بخش EXAMPLE 3 – Triggers & Views

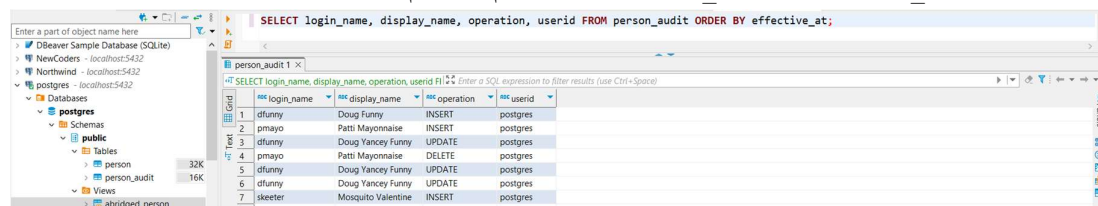
برای پنهان کردن کامل ستون، یک view ایجاد می‌کنیم که ستون ts_abstract را hide می‌کند، در حالی که همچنان از trigger بهره می‌برد:



حال یک رکورد به view که تعریف کردیم، اضافه می‌کنیم و میبینیم که خروجی به درستی اضافه شده است:

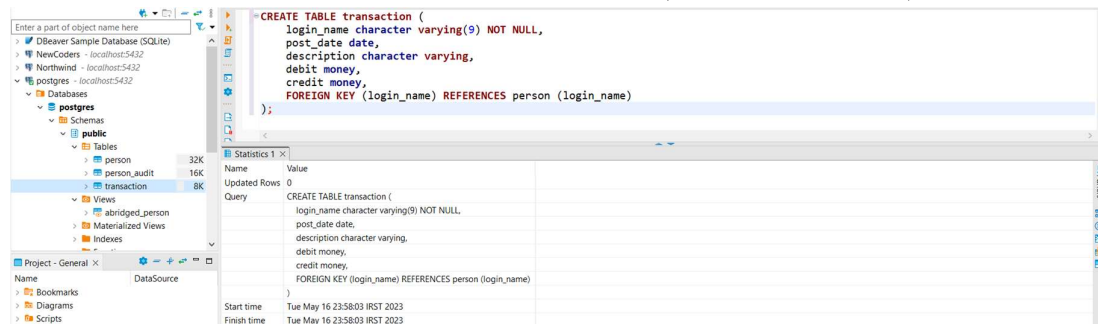


داده‌های جدول person_audit نشان می‌دهیم و می‌بینیم که به درستی نشان داده می‌شود:



بخش EXAMPLE 4 – Summary Values

در ابتدا یک جدول به نام transaction ایجاد می‌کنیم:



برای جلوگیری از نیاز به محاسبه مجدد balance با جمع کردن همه تراکنش‌ها در هر زمان که نیاز به موجودی است، می‌توانیم با اضافه کردن یک ستون جدید و استفاده از یک trigger و تابع ذخیره شده، یک مقدار موجودی فعلی را در جدول شخص، غیرعادی کرده و نگه داریم. مانده خالص به عنوان معاملات درج می‌شود:


```

ALTER TABLE person ADD COLUMN balance MONEY DEFAULT 0;

CREATE FUNCTION transaction_bit() RETURNS trigger
LANGUAGE plpgsql
SET SCHEMA 'public'
AS $$
DECLARE
    newbalance money;
BEGIN
    -- Update person account balance
    UPDATE person
    SET balance =
        balance +
        COALESCE(NEW.debit, 0::money) -
        COALESCE(NEW.credit, 0::money)
    WHERE login_name = NEW.login_name
    RETURNING balance INTO newbalance;
    -- Data validation
    IF COALESCE(NEW.debit, 0::money) < 0::money THEN
        RAISE EXCEPTION 'Debit value must be non-negative';
    END IF;
    IF COALESCE(NEW.credit, 0::money) < 0::money THEN
        RAISE EXCEPTION 'Credit value must be non-negative';
    END IF;
    IF newbalance < 0::money THEN
        RAISE EXCEPTION 'Insufficient funds: %', NEW;
    END IF;
    RETURN NEW;
END;
$$;

CREATE TRIGGER transaction_bit
BEFORE INSERT ON transaction
FOR EACH ROW EXECUTE PROCEDURE transaction_bit();

```

Name	Value
Queries	3
Updated Rows	0
Execute time (ms)	1800
Fetch time (ms)	0
Total time (ms)	1800
Start time	2023-05-17 00:02:00.780

برای تست کردن از دستورات زیر استفاده می‌کنیم و مشاهده می‌کنیم که خروجی‌ها درست هستند:

```

SELECT login_name, balance FROM person WHERE login_name = 'dfunny';

INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny', '2018-01-11', 'ACH CREDIT FROM: FINANCE AND ACCO ALLOTMENT : Direct Deposit', NULL, '$2,000.00');

```

Name	Value
Queries	2
Updated Rows	1
Execute time (ms)	6
Fetch time (ms)	0
Total time (ms)	6
Start time	2023-05-17 00:04:26.510
Finish time	2023-05-17 00:04:26.692

login_name	balance
dfunny	\$2,000.00

Execution Error

Reason:
SQL Error (P0001): ERROR: insufficient funds: (dfunny,2018-01-17,'FOR:BGE PAYMENT ACH Withdrawal','\$2780.52', NULL);

حال تعدادی دستور جهت تست اعمال می‌کنیم:

```

SELECT login_name, balance FROM person WHERE login_name = 'dfunny';

INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny', '2018-01-17', 'FOR:BGE PAYMENT ACH Withdrawal', '$2780.52', NULL);

SELECT login_name, balance FROM person WHERE login_name = 'dfunny';

```

Name	Value
Queries	3
Updated Rows	1
Execute time (ms)	6
Fetch time (ms)	0
Total time (ms)	6
Start time	2023-05-17 00:04:26.510
Finish time	2023-05-17 00:04:26.692

login_name	balance
dfunny	\$1,721.48

```

INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny', '2018-01-23', 'FOR: ANNE ARUNDEL ONLINE PMT ACH Withdrawal', '$35.29', NULL);

SELECT login_name, balance FROM person WHERE login_name = 'dfunny';

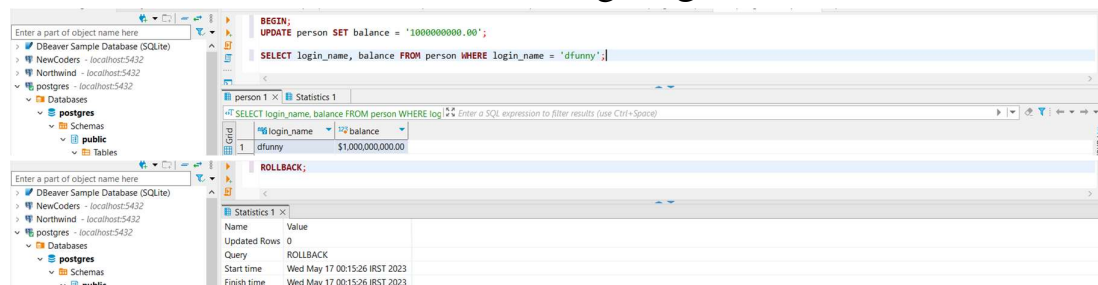
```

Name	Value
Queries	3
Updated Rows	1
Execute time (ms)	6
Fetch time (ms)	0
Total time (ms)	6
Start time	2023-05-17 00:04:26.510
Finish time	2023-05-17 00:04:26.692

login_name	balance
dfunny	\$1,686.19

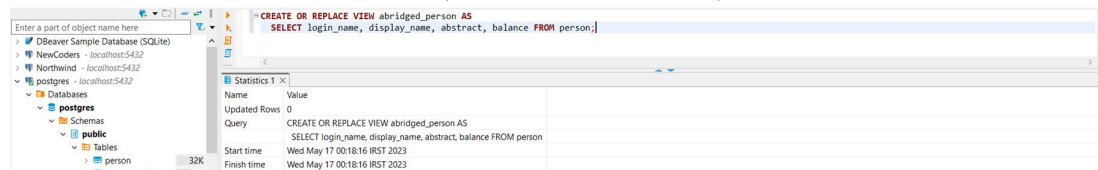
بخش 5 - Triggers and Views Redux

اجرای بخش قبل مشکلی دارد و آن این است که هیچ چیز مانع از چاپ پول توسط کاربر مخرب نمی شود:

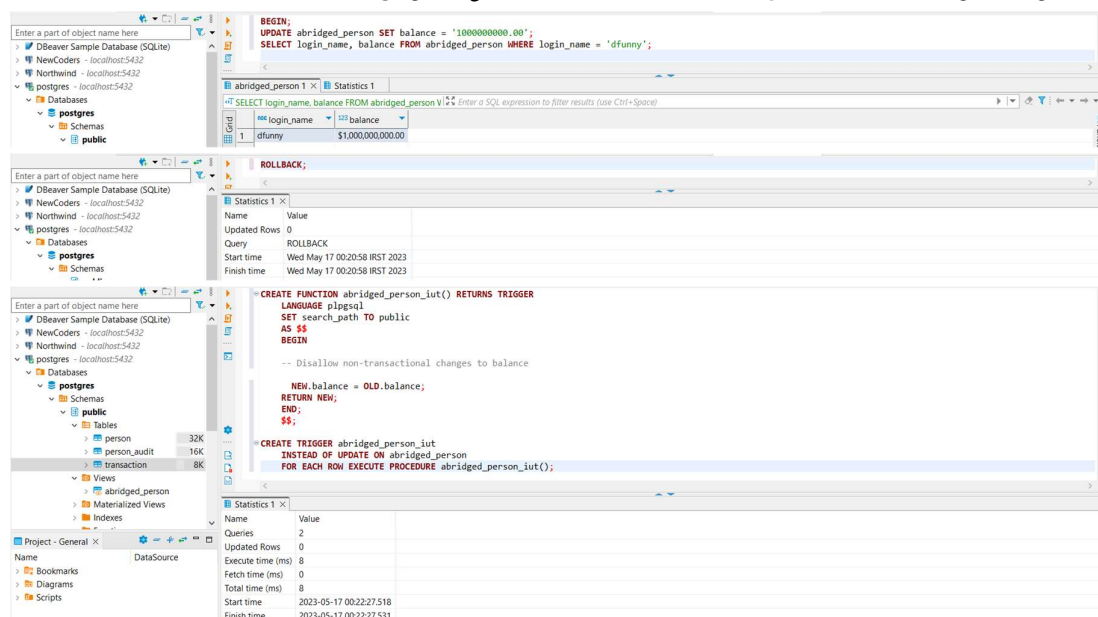


ما فعلاً سرقت بالا را لغو کرده‌ایم و راهی برای ایجاد محافظت در برابر با استفاده از یک trigger در view برای جلوگیری از به‌روزرسانی مقدار موجودی نشان خواهیم داد.

ابتدا view بخش قبلی را به روز رسانی می‌کنیم تا ستون balance را نشان دهیم:



بدیهی است که این امکان دسترسی خواندن به balance را فراهم می‌کند، اما همچنان مشکل را حل نمی‌کند زیرا برای view های ساده مانند این بر اساس یک جدول، PostgreSQL به طور خودکار view را قابل نوشتن می‌کند:



در این حالت به جای به‌روزرسانی trigger و stored procedure، هرگونه به‌روزرسانی را برای مقدار موجودی نادیده می‌گیرد و در عوض استفاده از مقدار موجود در پایگاه داده را قبل از trigger شدن دستور به‌روزرسانی قرار می‌دهد.

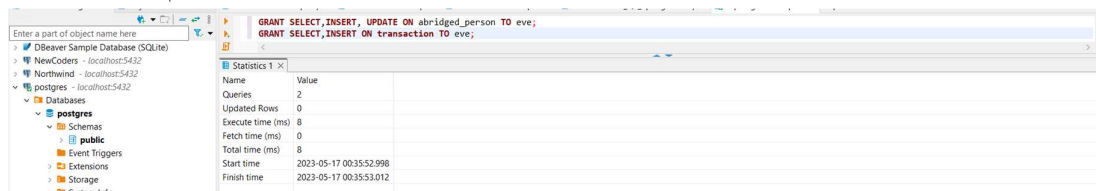
حال یک تست می نویسیم:



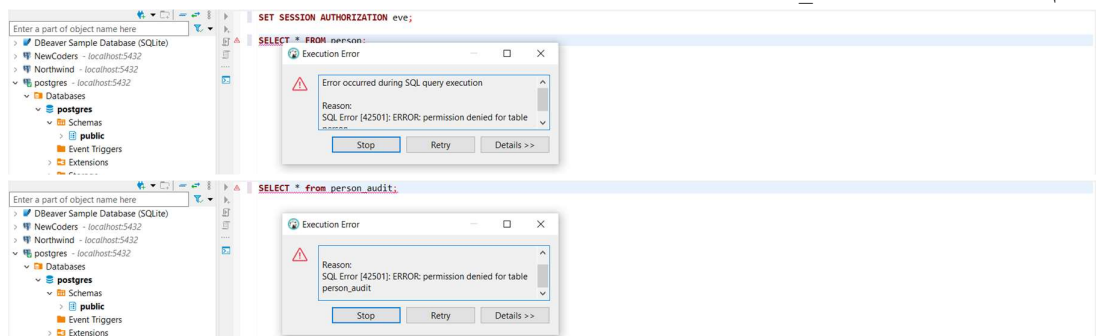
بخش EXAMPLE 6 - Elevated Privileges

ابتدا یک role non-privileged تعریف می کنیم.

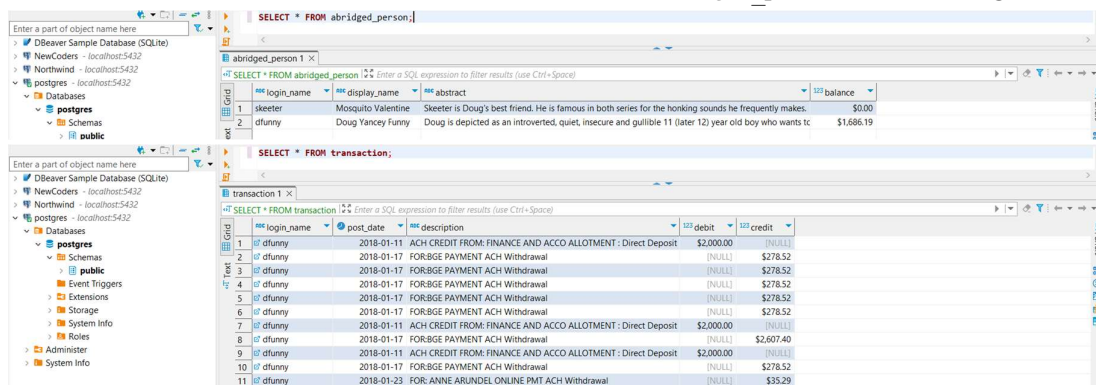
ما امتیاز خواندن، به روزرسانی، و ایجاد امتیاز در view شخص و خواندن و ایجاد در جدول تراکنش ها را می دهیم:



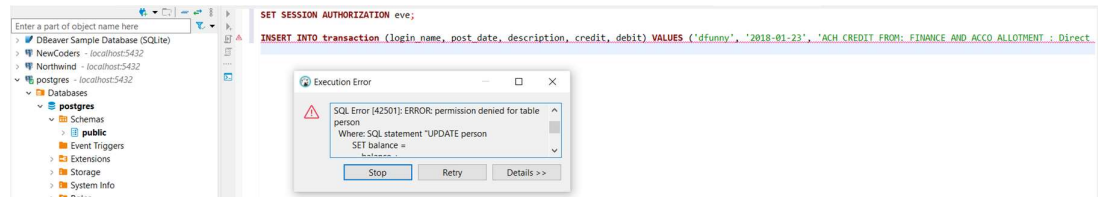
می بینیم که eve به person_audit و person دسترسی ندارد:



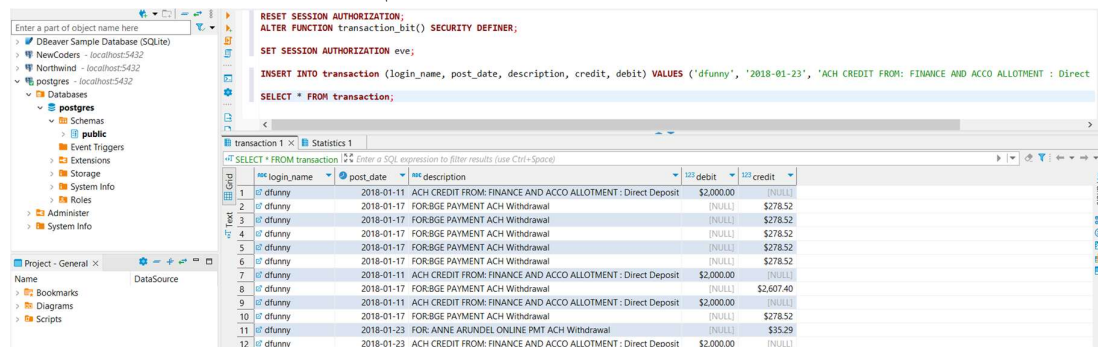
eve دسترسی خواندن به جداول abridged_person و transaction دارد:



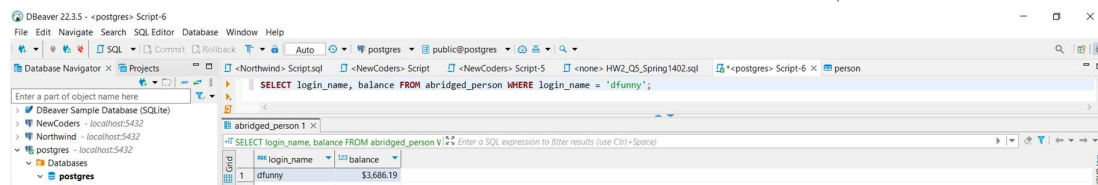
با این حال، حتی اگر eve دارای امتیاز نوشتن در جدول transaction است، insert تراکنش به دلیل نداشتن امتیاز در جدول person با شکست مواجه می شود.



در اینجا ویژگی SECURITY DEFINER را به stored function اضافه می کنیم:

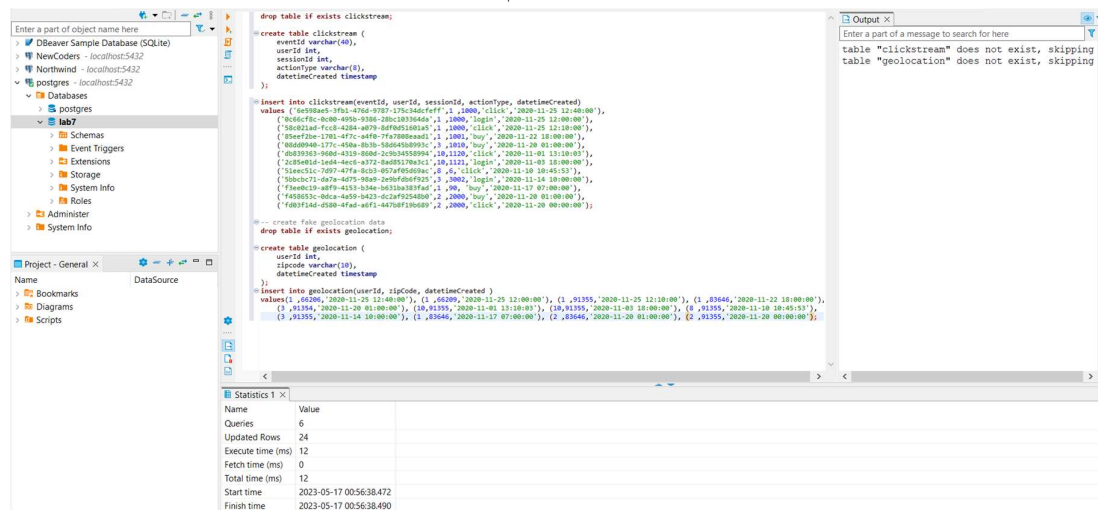


(*) توجه شود که در اینجا تعداد ستون‌ها طبق خروجی سایت نیست چون من خارج از دستورات سایت، دستوراتی را اعمال کردم اما طوری این دستورات را وارد کردم که در نهایت خروجی یکسان باشد).



بخش امتیازی

ابتدا یک دیتابیس در پستگرس ایجاد کرده، این دستورات را اجرا می کنیم تا جدول مربوطه ایجاد شود:



خروجی تکه کد داده شده به صورت زیر است:

The screenshot shows the DBeaver SQL editor interface. On the left, the 'Schemas' tree is expanded, showing the 'public' schema with tables like 'person', 'person_audit', 'transaction', 'views', 'materialized_views', 'indexes', and 'functions'. The main editor displays a SQL query using Common Table Expressions (CTEs) to calculate session metrics and purchase counts. The query is as follows:

```
with purchasingUsers as (
    select user_id,
    (sum(
        case
            when actionType = 'buy' then 1
            else 0
        end
    )) as numPurchases
    from clickstream
    group by user_id
    having count(distinct sessionId) > 1
),
userSessionMetrics as (
    select user_id,
    numclicks,
    numlogins,
    (sum(
        case
            when actionType = 'click' then 1
            else 0
        end
    )) as numclicks,
    (sum(
        case
            when actionType = 'login' then 1
            else 0
        end
    )) as numlogins,
    (sum(
        case
            when actionType = 'buy' then 1
            else 0
        end
    )) as numPurchases
    from clickstream
    group by user_id, sessionId
),
select sum.*
from userSessionMetrics as um
join purchasingUsers as pu on um.user_id = pu.user_id
join clickstream as cs on um.user_id = pu.user_id
```

The 'Output' tab at the bottom shows the results of the query, which is a table with 6 columns: 'user_id', 'sessionId', 'numclicks', 'numlogins', 'numPurchases', and 'numPurchases'. The data is as follows:

user_id	sessionId	numclicks	numlogins	numPurchases	numPurchases
1	1,000	2	1	0	0
1	1,001	0	0	0	1
1	90	0	0	0	1
2	2,000	1	0	0	1
3	1,010	0	0	0	1
3	3,002	0	1	1	0

عبارت "case" در کد SQL ارائه شده یک عبارت شرطی است که برای ارزیابی یک شرط یا معیار خاص و برگرداندن مقادیر مختلف بر اساس درست یا نادرست بودن شرط استفاده می‌شود. در این کد برای تعریف سه معیار مجموع مختلف استفاده می‌شود: «numPurchases» در «purchasingUsers»، «numclicks» و «numlogins» در «userSessionMetrics». عبارت "case" مقدار ستون "actionType" را در جدول "clickstream" ارزیابی می‌کند. اگر "actionType" برابر با "buy" در "purchasingUsers" یا "login" / "click" در "userSessionMetrics" باشد، یک را برمی‌گرداند، در غیر این صورت، صفر را برمی‌گرداند. این مقادیر سپس با استفاده از تابع "sum" برای aggregate metrics برای هر کاربر محاسبه می‌شود.

مشکلات و توضیحات تکمیلی

مشکلی در قسمت اصلی پروژه نبود؛ فقط در قسمت امتیازی یک خط به من ارور میداد که با سعی و خطا ارور را برطرف کردم.

آنچه آموختم / پیشنهادات

با توابع و تریگرها در بانک‌های اطلاعاتی رابطه‌ای آشنا شدم.