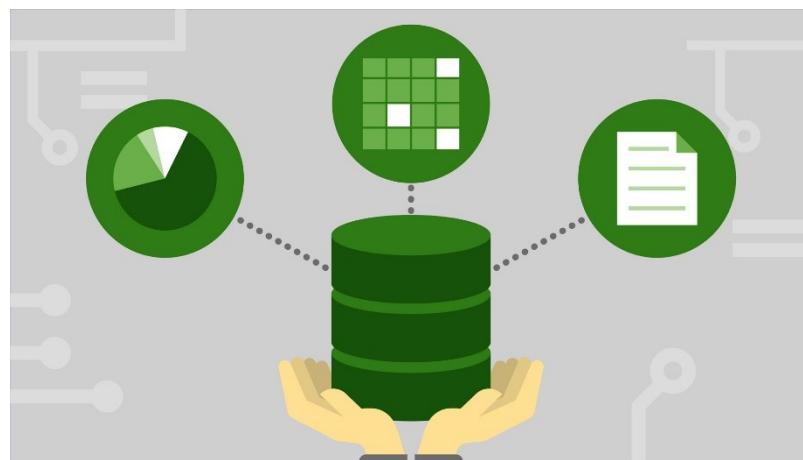


به نام خدا



دانشگاه تهران  
پردیس دانشکده‌های فنی  
دانشکده برق و کامپیوتر



## آزمایشگاه پایگاه داده

ستور کار شماره ۵

آتیه آرمین

۸۱۰۱۹۷۶۴۸

مهرماه ۱۴۰۰

## گزارش دستورکار انجام شده

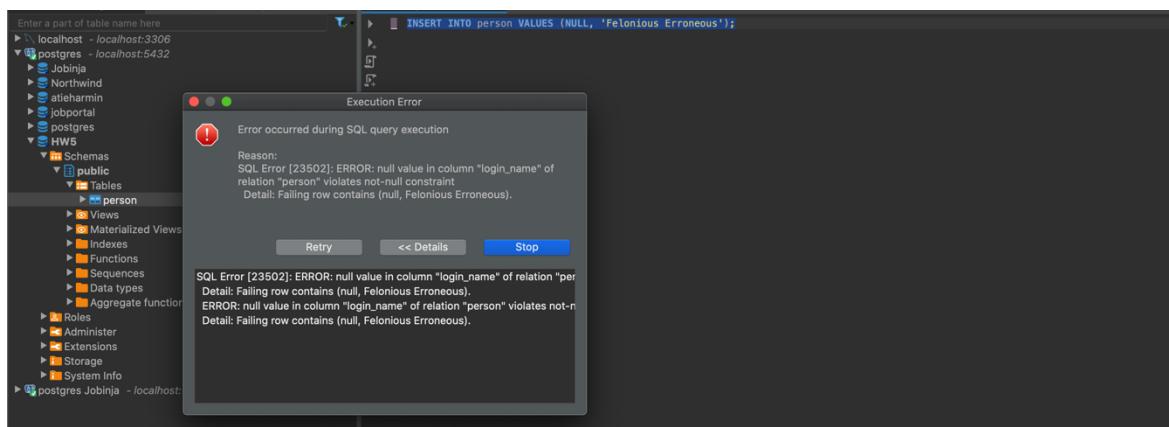
### بخش اول

۱ - در این دستور یک جدول به نام person ساخته شده که دو ویژگی login name و display name دارد. Login display name person و login name نمیتواند null و یا بیش از ۹ حرف باشد.

```

CREATE TABLE person (
    login_name varchar(9) not null primary key,
    display_name text
);

```



۲ - برخی دیگر از محدودیت ها را میتوان با alert نشان داد. با این دستور روی یک جدول یک محدودیت جدید را اضافه کرده و با کلمه check شرط جدید را تعریف میکنیم. در ادامه ۲ نمونه از این دستورات آمده است که اولیه روی جدول person محدودیت آنکه طول login name حتی بیشتر از ۹ باشد را اضافه کرده و دومی شرط آنکه در space، login name باشیم را چک میکند.

```

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

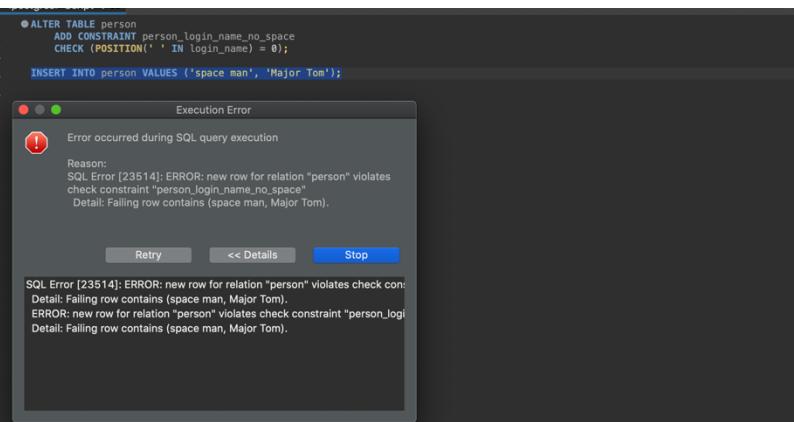
INSERT INTO person VALUES ('', 'Felonious Erroneous');

```

Error occurred during SQL script execution

Reason:  
SQL Error [23514]: ERROR: new row for relation "person" violates check constraint "person\_login\_name\_non\_null"  
Detail: Failing row contains (, Felonious Erroneous).

SQL Error [23514]: ERROR: new row for relation "person" violates check constraint "person\_login\_name\_non\_null"  
Detail: Failing row contains (, Felonious Erroneous).  
ERROR: new row for relation "person" violates check constraint "person\_login\_name\_non\_null"  
Detail: Failing row contains (, Felonious Erroneous).

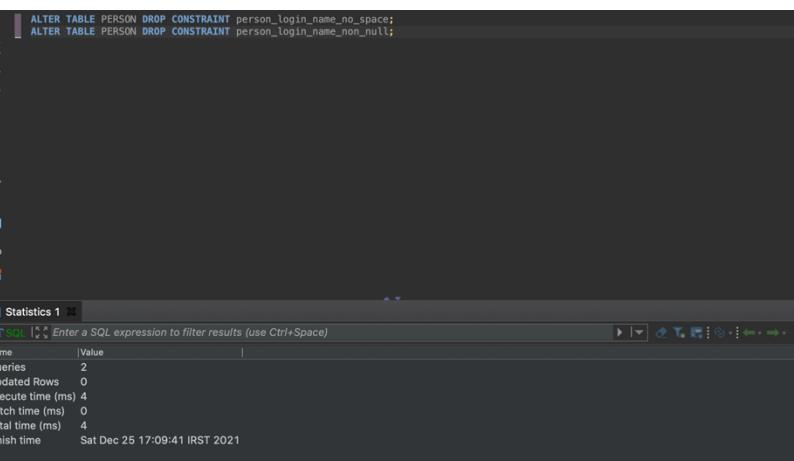


```

ALTER TABLE person
ADD CONSTRAINT person_login_name_no_space
CHECK (POSITION(' ' IN login_name) = 0);
INSERT INTO person VALUES ('space man', 'Major Tom');

```

۳ - با دستور `drop constraint` میتوان محدودیت ها را از روی یک جدول خاص حذف کرد.

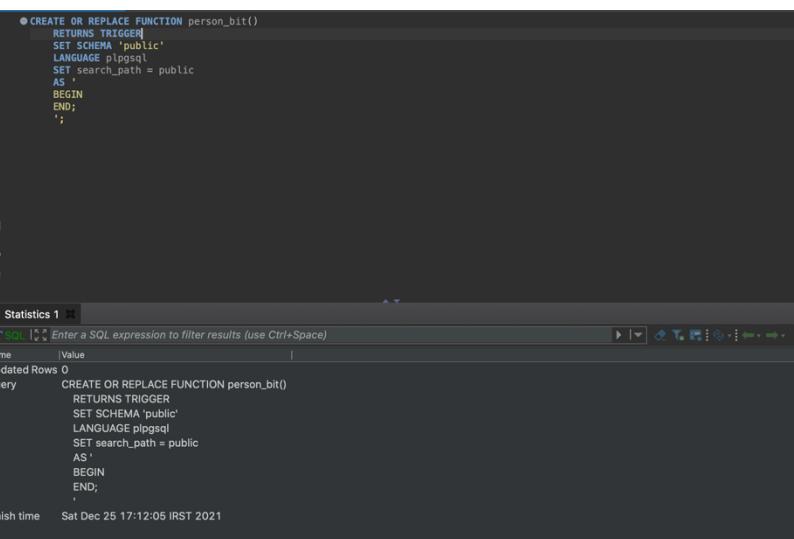


```

ALTER TABLE PERSON DROP CONSTRAINT person_login_name_no_space;
ALTER TABLE PERSON DROP CONSTRAINT person_login_name_no_null;

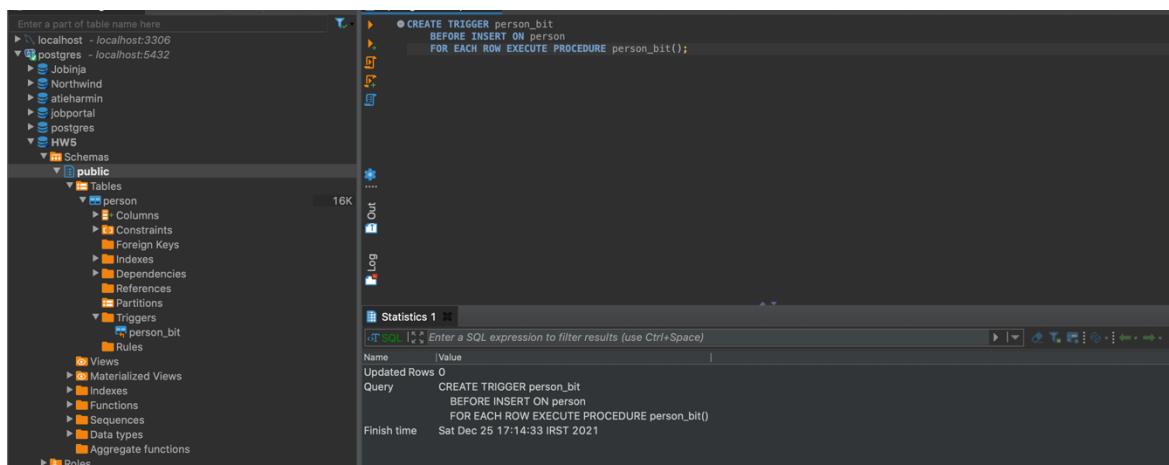
```

۴ - در زیر یک تابع خالی به نام `person_bit` تعریف شده است. طبق حروف bit این تابع قرار است یک تابع باشد که باشد و قبل از اضافه شدن یک `person`, توسط یک `before insert trigger` انجام شود که نحوه استفاده آن نیز در عکس دوم آمده است.



```

CREATE OR REPLACE FUNCTION person_bit()
RETURNS TRIGGER
SET SCHEMA 'public'
LANGUAGE plpgsql
SET search_path = public
AS
BEGIN
END;
'
```



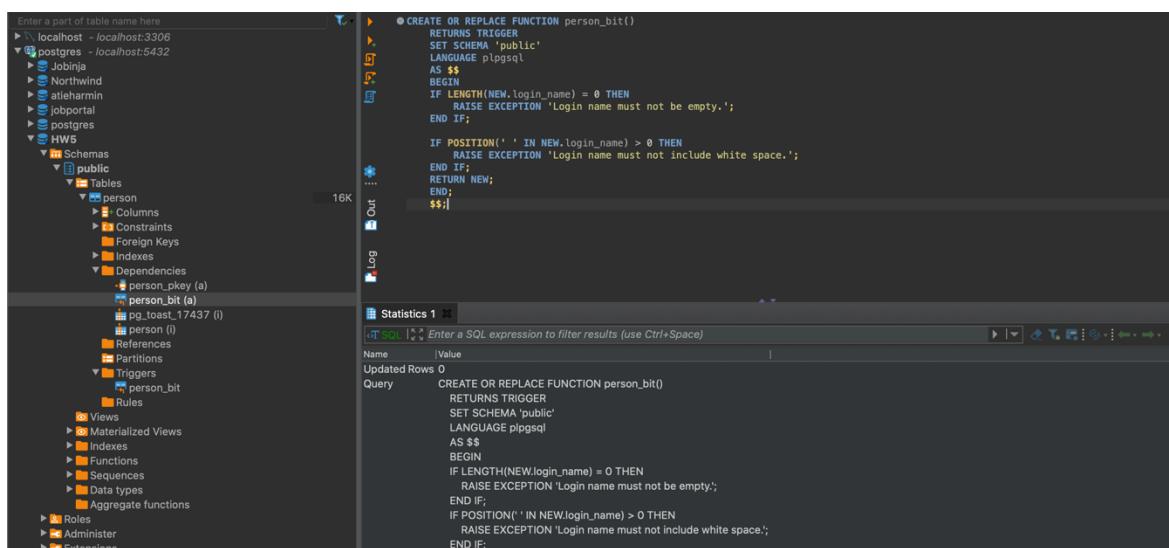
```

CREATE TRIGGER person_bit
BEFORE INSERT ON person
FOR EACH ROW EXECUTE PROCEDURE person_bit();

```

The screenshot shows the pgAdmin interface. On the left, the database structure is visible, including the 'public' schema which contains a 'person' table. On the right, a query editor window displays the SQL code for creating a trigger named 'person\_bit' that triggers before an insert operation on the 'person' table, executing the 'person\_bit()' procedure.

۵- در مثال زیر محدودیت هایی که بالاتر تعریف کرده بودیم و حذف کردیم را در تابع (person\_bit) بیاده میکنیم تا پیش از اضافه شدن یک person، این شرط ها چک شوند. لازم به ذکر است که NEW نشان دهنده ردیفی از دیتا است که قرار است insert شود. در عکس دوم مانند قبل سعی بر اضافه کردن یک person بدون login name خواهیم داشت که با ارور مواجه میشویم.



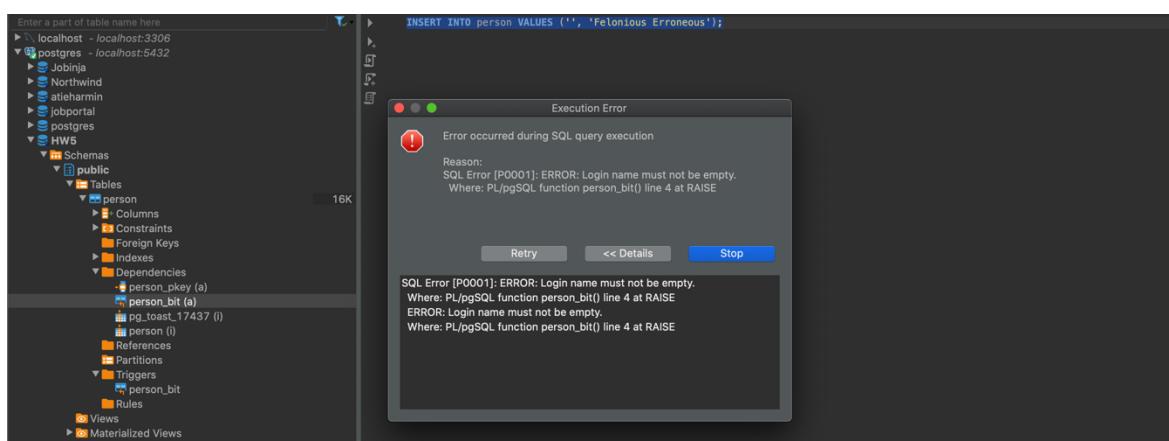
```

CREATE OR REPLACE FUNCTION person_bit()
RETURNS TRIGGER
SET SCHEMA 'public'
LANGUAGE plpgsql
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;
RETURN NEW;
END;
$$;

```

This screenshot shows the function definition for 'person\_bit()'. The function is defined to return a trigger that sets the schema to 'public', uses the PL/pgSQL language, and contains logic to check if the length of the new login name is zero and if it contains any whitespace. It then returns the new row.



An 'Execution Error' dialog box is displayed, stating that an error occurred during SQL query execution. The error message is: "SQL Error [P0001]: ERROR: Login name must not be empty. Where: PL/pgSQL function person\_bit() line 4 at RAISE".

The screenshot also shows the pgAdmin interface with the 'public' schema and 'person' table selected on the left, and the error dialog box in the foreground.

6 - در این مثال میخواهیم تا پیش از insert و update، اطلاعات در یک جدول دیگر به نام person\_audit ذخیره شوند. همچنین میخواهیم اطلاعات ای که قرار از پاک شود، پیش از پاک شدن در این جدول ذخیره شود. برای این کار ابتدا این جدول را پیاده سازی کرده، سپس () را با دستور insert into person\_bit() را با طوری تغییر میدهیم که اطلاعات را در person\_audit ذخیره کند. همچنینتابع جدیدی به نام person\_bdt() پیاده سازی میکنیم تا پیش از پاک کردن نیز اطلاعات را بتوانیم در person\_audit ذخیره کنیم.

```

CREATE TABLE person_audit (
    login_name varchar(9) NOT NULL,
    display_name TEXT,
    operation CHAR(1),
    effective_at timestamp NOT NULL DEFAULT now(),
    user_id name NOT NULL DEFAULT session_user
);

CREATE OR REPLACE FUNCTION person_bit()
RETURNS TRIGGER
SET SCHEMA 'public'
LANGUAGE plpgsql
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;

    -- New code to record audits
    INSERT INTO person_audit (login_name, display_name, operation)
    VALUES (NEW.login_name, NEW.display_name, TG_OP);

    RETURN NEW;
END;
$$;

DROP TRIGGER person_bit ON person;

CREATE TRIGGER person_bit
BEFORE INSERT OR UPDATE ON person
FOR EACH ROW EXECUTE PROCEDURE person_bit();

```

```

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();

```

حال با اضافه و حذف کردن چند رکورد، کارایی تابع های بالا را میبینیم.

```

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

SELECT * FROM person;

SELECT * FROM person_audit;

```

	login_name	display_name
1	dfunny	Doug Funny
2	pmayo	Patti Mayonnaise

	login_name	display_name	operation	effective_at	userid
1	dfunny	Doug Funny	INSERT	2021-12-25 17:28:09	postgres
2	pmayo	Patti Mayonnaise	INSERT	2021-12-25 17:28:09	postgres

حال میبینیم با تغییر display\_name بکی از این رکورد ها جدول person\_audit هم تغییر میکند.

	login_name	display_name	operation	effective_at	userid
1	dfunny	Doug Funny	INSERT	2021-12-25 17:28:09	postgres
2	pmayo	Patti Mayonnaise	INSERT	2021-12-25 17:28:09	postgres
3	dfunny	Doug Yancey Funny	UPDATE	2021-12-25 17:30:52	postgres

The screenshot shows the pgAdmin interface with the following details:

- Schemas:** public
- Tables:** person, person\_audit
- person\_audit Structure:**
  - Columns: login\_name, display\_name, operation, abstract, effective\_at, userid
  - Foreign Keys: None
  - Indexes: None
  - Dependencies: None
  - References: None
  - Partitions: None
  - Triggers: None
  - Rules: None
- Data in person\_audit:**

login_name	display_name	operation	abstract	effective_at	userid
dfunny	Doug Funny	INSERT		2021-12-25 17:28:09	postgres
pmayo	Patti Mayonnaise	INSERT		2021-12-25 17:28:09	postgres
dfunny	Doug Yancey Funny	UPDATE		2021-12-25 17:30:52	postgres
pmayo	Patti Mayonnaise	DELETE		2021-12-25 17:34:39	postgres

7- در این مثال میخواهیم یک ستون به نام `abstract` که نوع آن `text` است و هیچ محدودیتی برای این `text` نخواهیم داشت. سپس یک ستون دیگر به نام `ts_abstract` که `TSVECTOR` از ستون `abstract` است و جایگاه هر کلمه را در `text` نشان میدهد. در جدول `person_audit` نیازی به نگه داری این بردار برای سرچ راحت نخواهیم داشت بنابر این تنها ستون `abstract` را اضافه میکنیم.

The screenshot shows the pgAdmin interface with the following details:

- Schemas:** public
- Tables:** person, person\_audit
- person\_audit Structure:**
  - Columns: login\_name, display\_name, operation, abstract, effective\_at, userid
  - Foreign Keys: None
  - Indexes: ts\_abstract
  - Dependencies: None
  - References: None
  - Partitions: None
  - Triggers: None
  - Rules: None
- Data in person\_audit:**

login_name	display_name	operation	abstract	effective_at	userid
dfunny	Doug Funny	INSERT		2021-12-25 17:28:09	postgres
pmayo	Patti Mayonnaise	INSERT		2021-12-25 17:28:09	postgres
dfunny	Doug Yancey Funny	UPDATE		2021-12-25 17:30:52	postgres
pmayo	Patti Mayonnaise	DELETE		2021-12-25 17:34:39	postgres

The screenshot shows the pgAdmin interface with the following details:

- Schemas:** public
- Tables:** person, person\_audit
- person\_audit Structure:**
  - Columns: login\_name, display\_name, operation, abstract, effective\_at, userid
  - Foreign Keys: None
  - Indexes: ts\_abstract
  - Dependencies: None
  - References: None
  - Partitions: None
  - Triggers: None
  - Rules: None
- Data in person\_audit:**

login_name	ts_abstract
dfunny	'11:11'12:13 'boy':16 'crowd':24 'depict':3 'do

8 - چون نمیخواهیم که افراد به ستون ts\_abstract دسترسی داشته باشند، یک view از person می‌سازیم که تنها ستون های abstract و login name,display name را داشته باشد. حال با اضافه کردن یک رکورد جدید به این view، ستون خودکار پر می‌شود.

```

CREATE VIEW abridged_person AS SELECT login_name, display_name, abstract FROM person;
INSERT INTO abridged_person VALUES ('skeeter', 'Mosquito Valentine', 'Skeeter is Doug''s best friend. He is famous in both series f');
SELECT login_name, ts_abstract FROM person WHERE login_name = 'skeeter';
SELECT login_name, display_name, operation, userid FROM person_audit ORDER BY effective_at;

```

	login_name	ts_abstract
1	skeeter	'best':5 'doug':3 'famous':9 'frequent':18 'friend':1

	login_name	display_name	operation	userid
1	dfunny	Doug Funny	INSERT	postgres
2	pmayo	Patti Mayonnaise	INSERT	postgres
3	dfunny	Doug Yancey Funny	UPDATE	postgres
4	pmayo	Patti Mayonnaise	DELETE	postgres
5	dfunny	Doug Yancey Funny	UPDATE	postgres
6	skeeter	Mosquito Valentine	INSERT	postgres

۹- در این مثال یک جدول اضافه میکنیم به نام register transaction تا debit و credit را چک کنیم. از طرفی balance در اپلیکیشن ما مهم است بنابر این برای آنکه هر بار برای هر فرد نخواهیم balance را محاسبه کنیم، یک ستون به عنوان balance برای جدول person در نظر گرفته و پیش از هر بار insert کردن یک update transaction را این balance را میکنیم.

```

-- 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();

```

در اینجا چون balance کمتر از ۰ میشود به اور میخوریم.

Execution Error

Error occurred during SQL query execution

Reason:

SQL Error [P0001]: ERROR: Insufficient funds:  
(dfunny,2018-01-17,'FOR:BGE PAYMENT ACH  
Withdrawal','\$2,780.52')  
Where: PL/pgSQL function transaction\_bit() line 27 at RAISE

SQL Error [P0001]: ERROR: Insufficient funds: (dfunny,2018-01-17,'FOR:BGE PAYMENT ACH V  
Where: PL/pgSQL function transaction\_bit() line 27 at RAISE

	login_name	balance
1	dfunny	\$2,000.00

- ۱۰ در مثال قبل میتوان به راحتی با یک update مقدار balance هر فرد را تغییر داد. برای آنکه از این اتفاق جلوگیری کنیم ابتدا view ای که بالاتر ساخته بودیم را به روز رسانی کرده و ستون balance را به آن اضافه میکنیم. سپس یک trigger جدید با مضمون instead of update do something روی این view تعریف میکنیم. در این صورت امکان تغییر ستون balance روی این view نخواهد بود.

```

CREATE OR REPLACE VIEW abridged_person AS
SELECT login_name, display_name, abstract, balance FROM person;

CREATE FUNCTION abridged_person_iut() RETURNS TRIGGER
LANGUAGE plpgsql
SET search_path TO public
AS $$
BEGIN
    -- Disallow non-transactional changes to balance
    NEW.balance = OLD.balance;
    RETURN NEW;
END;
$$;

CREATE TRIGGER abridged_person_iut
INSTEAD OF UPDATE ON abridged_person
FOR EACH ROW EXECUTE PROCEDURE abridged_person_iut();

UPDATE abridged_person SET balance = '1000000000.00';

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

```

login_name	balance
dfunny	\$2,000.00

- ۱۱ در ادامه یک user با قابلیت های محدود تعریف میکنیم تا هر فردی نتواند تمام اطلاعات را تغییر دهد. مثلاً یک user به نام eve ساخته و اختیارات او را روی view ساخته شده محدود به update و insert و select میکنیم. کرده و اختیارات او را روی جدول transaction محدود به select و insert میکنیم.

```

CREATE USER eve;
GRANT SELECT, INSERT, UPDATE ON abridged_person TO eve;
GRANT SELECT, INSERT ON transaction TO eve;

SET SESSION AUTHORIZATION eve;
SELECT * FROM person;

SELECT * from person_audit;

```

Error occurred during SQL script execution

Reason:  
SQL Error [42501]: ERROR: permission denied for table person

Retry Skip Skip all << Details

SQL Error [42501]: ERROR: permission denied for table person  
ERROR: permission denied for table person

هر چند این user درسترسی نوشتن روی جدول transaction را دارد ولی چون اضافه کردن یک transaction روی جدول person در ستون balance میگذارد باید دسترسی نوشتن به person را هم داشته باشد و گرنه به ارور زیر برمیخورد. برای دور زدن این ارور و تعویض مقدار balance بدون دادن دسترسی نوشتن در جدول person به صورتی که در عکس دوم نشان داده شده است، عمل میکنیم.

```

SET SESSION AUTHORIZATION eve;
INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny', '2018-01-23', 'ACH CREDIT FROM: FINAN...

```

```

RESET SESSION AUTHORIZATION;
ALTER FUNCTION transaction_bit() SECURITY DEFINER;
SET SESSION AUTHORIZATION eve;
INSERT INTO transaction (login_name, post_date, description, credit, debit) VALUES ('dfunny', '2018-01-23', 'ACH CREDIT FROM: FINAN...
SELECT * FROM transaction;

```

	login_name	post_date	description	debit
1	dfunny	2018-01-23	ACH CREDIT FROM: FINANCE AND ACCO ALLOT	\$2,000.00

## بخش دوم

1 - دستور زیر تاریخ آخرین order هر فرد را نشان میدهد.

```

select t.customer_id, t.last_order_date
from (select customer_id, order_date, max(order_date) over (partition by customer_id) as last_order_date
      from orders) t
where t.order_date = t.last_order_date

```

customer_id	last_order_date
ALFKI	1998-04-09
ANATR	1998-03-04
ANTON	1998-01-28
AROUT	1998-04-10
BERGS	1998-03-04
BLAUS	1998-04-29
BLONP	1998-01-12
BOLID	1998-03-24
BONAP	1998-05-06
BOTTM	1998-04-24
BSBEV	1998-04-14

۲ - این دستور به ازای هر `supplier` آن `product` ای از او را نشان میدهد که بیشترین تعداد را در `order` داشته است.

```
Enter a part of table name here
localhost - localhost:3306
postgres - localhost:5432
HW5
Jobinja
Northwind
  Schemas
    public
      Tables
        categories
        customer_customer_demo
        customer_demographics
        customers
        employee_territories
        employees
        order_details
        orders
        products
        region
        shippers
        suppliers
        territories
        us_states
      Views
      Materialized Views
      Indexes
      Functions
      Sequences
      Data types
      Aggregate functions
    Roles
    Administer
    Extensions
    Storage
    System Info
  atieharmi
  jobportal

select t.supplier_id, product_name, t.maximum_product_order from supplier
  from (select supplier_id, units_on_order , product_name ,max(units_on_order) over (partition by supplier_id) as maximum_product_order
        from products ) t
  where t.units_on_order = t.maximum_product_order_from_supplier|
```

	supplier_id	product_name	maximum_product_order_from_supplier
1	1	Aniseed Syrup	70
2	2	Louisiana Hot Spiced Okra	100
3	3	Uncle Bob's Organic Dried Pears	0
4	3	Northwoods Cranberry Sauce	0
5	3	Grandma's Boysenberry Spread	0
6	4	Longlife Tofu	20
7	5	Queso Cabrales	30
8	6	Konbu	0
9	6	Tofu	0
10	6	Genen Shouyu	0
11	7	Outback Lager	10

۳ - دستور زیر به ازای هر `customer` هزینه‌ای که در یک `order` کرده است را نشان میدهد.

```
Enter a part of table name here
localhost - localhost:3306
postgres - localhost:5432
HW5
Jobinja
Northwind
  Schemas
    public
      Tables
        categories
        customer_customer_demo
        customer_demographics
        customers
        employee_territories
        employees
        order_details
        orders
        products
        region
        shippers
        suppliers
        territories
        us_states
      Views
      Materialized Views
      Indexes
      Functions
      Sequences
      Data types
      Aggregate functions
    Roles
    Administer
    Extensions
    Storage
    System Info
  atieharmi
  jobportal
  postgres

select t.customer_id, t.max_order
  from (select c.customer_id , od.unit_price * od.quantity as price ,max(od.unit_price * od.quantity) over (partition by c.customer_id
        from orders o , order_details od , customers c
        where o.order_id = od.order_id and o.customer_id = c.customer_id)
  where t.price = t.max_order|
```

	customer_id	max_order
1	ALFKI	878.0000305176
2	ANATR	347.9999923706
3	ANTON	1,050
4	AROUT	4,050
5	BERGS	3,952.5
6	BLAUS	714
7	BLONP	3,465
8	BOLID	2,475.8000183105
9	BONAP	1,500
10	BONAP	1,500
11	BOTTM	2,957.9999542236