МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«КУРГАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Кафедра «Программное обеспечение автоматизированных систем»

**Проект БД для магазина по продаже вычислительной техники**

**ТЕКСТ ПРОГРАММЫ**

**РФ** **41287462 260091 12**

**Листов 18**

Курган 2025

**АННОТАЦИЯ**

Документ содержит исходный код реализованных на языке PL/pgSQL, функций:

* «products.products\_get\_by\_category»;
* «storages.inventory\_get\_product\_count»;
* «clients.basket\_info\_set»;
* «analitics.get\_product\_discount»;
* «analitics.get\_category\_discount»;
* «clients.client\_get\_discount»;
* «analitics.get\_sales\_analytics».

Оглавление

[ИСХОДНЫЙ КОД ФУНКЦИИ «products.products\_get\_by\_category » 4](#_Toc198495969)

[ИСХОДНЫЙ КОД ФУНКЦИИ «storages.inventory\_get\_product\_count» 6](#_Toc198495970)

[ИСХОДНЫЙ КОД ФУНКЦИИ «clients.basket\_info\_set» 7](#_Toc198495971)

[ИСХОДНЫЙ КОД ФУНКЦИИ «analitics.get\_product\_discount» 10](#_Toc198495972)

[ИСХОДНЫЙ КОД ФУНКЦИИ «analitics.get\_category\_discount» 13](#_Toc198495973)

[ИСХОДНЫЙ КОД ФУНКЦИИ «clients.client\_get\_discount» 15](#_Toc198495974)

[ИСХОДНЫЙ КОД ФУНКЦИИ «analitics.get\_sales\_analytics» 16](#_Toc198495975)

# ИСХОДНЫЙ КОД ФУНКЦИИ «products.products\_get\_by\_category »

CREATE FUNCTION products.products\_get\_by\_category(p\_id\_category integer)

RETURNS TABLE(id integer, c\_category\_name text, c\_name text, c\_description text, c\_price numeric, c\_manufacturer\_name text, c\_model\_name text, c\_count integer)

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

BEGIN

IF NOT EXISTS(SELECT 1

FROM categories.t\_category c

WHERE c.id = p\_id\_category

FOR UPDATE) THEN

RAISE EXCEPTION 'Category with id % does not exist.', p\_id\_category;

END IF;

RETURN QUERY

SELECT p.id,

c.c\_name AS c\_category\_name,

p.c\_name,

p.c\_description,

p.c\_price,

mfr.c\_name AS c\_manufacturer\_name,

m.c\_name AS c\_model\_name,

storages.inventory\_get\_product\_count(p.id) AS c\_count

FROM products.t\_product p

JOIN

products.t\_model m ON p.id\_model = m.id

JOIN

products.t\_manufacturer mfr ON m.id\_manufacturer = mfr.id

JOIN

products.t\_product\_property pp ON pp.id\_product = p.id

JOIN

categories.t\_category\_property cp ON pp.id\_category\_property = cp.id

JOIN

categories.t\_category c ON cp.id\_category = c.id

WHERE c.id IN (SELECT cc.id

FROM categories.t\_category cc

WHERE cc.id\_parent = p\_id\_category)

GROUP BY p.id, c.c\_name, mfr.c\_name, m.c\_name;

IF NOT found THEN

RAISE EXCEPTION 'There are no products in this category with id %', p\_id\_category;

END IF;

END;

$$;

# ИСХОДНЫЙ КОД ФУНКЦИИ «storages.inventory\_get\_product\_count»

CREATE FUNCTION storages.inventory\_get\_product\_count(p\_id integer) RETURNS integer

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

DECLARE

v\_count integer;

BEGIN

IF NOT EXISTS(SELECT 1 FROM products.t\_product WHERE id = p\_id FOR UPDATE) THEN

RAISE EXCEPTION 'Product with id % does not exist.', p\_id;

END IF;

SELECT COUNT(p.id)

INTO v\_count

FROM products.t\_product\_instance pi

JOIN

products.t\_product p ON p.id = pi.id\_product

JOIN

storages.t\_inventory ti ON pi.id = ti.id\_product\_instance

WHERE ti.c\_event\_type NOT IN ('Sold', 'Scrapped')

AND p.id = p\_id

GROUP BY p.id;

RETURN v\_count;

END;

$$;

# ИСХОДНЫЙ КОД ФУНКЦИИ «clients.basket\_info\_set»

CREATE FUNCTION clients.basket\_info\_set(p\_id\_client integer, p\_id\_product integer, p\_count integer) RETURNS integer

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

DECLARE

v\_discount numeric ;

v\_product\_discount numeric ;

v\_id integer;

v\_product\_cost numeric;

BEGIN

IF NOT EXISTS(SELECT 1

FROM clients.t\_client

WHERE id = p\_id\_client

FOR UPDATE) THEN

RAISE EXCEPTION 'Client with id % does not exist.', p\_id\_client;

END IF;

IF NOT EXISTS(SELECT 1

FROM products.t\_product

WHERE id = p\_id\_product

FOR UPDATE) THEN

RAISE EXCEPTION 'Product with id % does not exist.', p\_id\_product;

END IF;

SELECT c\_price

INTO v\_product\_cost

FROM products.t\_product

WHERE id = p\_id\_product

FOR UPDATE;

v\_product\_discount := analitics.get\_product\_discount(

ARRAY ['Product', 'Category', 'Model'],

p\_id\_product

);

v\_discount := COALESCE(v\_product\_discount, clients.client\_get\_discount(p\_id\_client));

IF EXISTS(SELECT 1

FROM clients.t\_basket\_info

WHERE id\_client = p\_id\_client

AND id\_product = p\_id\_product

FOR UPDATE) THEN

UPDATE clients.t\_basket\_info

SET c\_count = c\_count + p\_count,

c\_batch\_cost = c\_batch\_cost + p\_count \* v\_product\_cost,

c\_discount = v\_discount

WHERE id\_client = p\_id\_client

AND id\_product = p\_id\_product

RETURNING id INTO v\_id;

ELSE

INSERT INTO clients.t\_basket\_info (id\_client, id\_product, c\_count, c\_batch\_cost, c\_discount)

VALUES (p\_id\_client, p\_id\_product, p\_count, v\_product\_cost \* p\_count, v\_discount)

RETURNING id INTO v\_id;

END IF;

RETURN v\_id;

EXCEPTION

WHEN foreign\_key\_violation THEN

RAISE EXCEPTION 'Client or product with specified id does not exist.';

WHEN check\_violation THEN

RAISE EXCEPTION 'Invalid basket info data.';

WHEN OTHERS THEN

RAISE EXCEPTION 'Error adding basket info: %', sqlerrm;

END;

$$;

# ИСХОДНЫЙ КОД ФУНКЦИИ «analitics.get\_product\_discount»

CREATE FUNCTION analitics.get\_product\_discount(p\_path text[], p\_id\_object integer) RETURNS numeric

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

DECLARE

v\_discount numeric := 0;

v\_id\_object integer;

v\_id\_category integer;

v\_type text;

BEGIN

IF array\_length(p\_path, 1) NOT BETWEEN 1 AND 3 THEN

RAISE EXCEPTION 'p\_path length must be between (1, 3)';

END IF;

IF p\_path[0] != 'Product'::text THEN

RAISE EXCEPTION 'First element of array must be "Product"';

END IF;

IF NOT EXISTS(SELECT 1 FROM products.t\_product WHERE id = p\_id\_object FOR UPDATE) THEN

RAISE EXCEPTION 'Product with id % does not exist.', p\_id\_object;

END IF;

FOREACH v\_type IN ARRAY p\_path

LOOP

IF v\_type = 'Product' THEN

v\_id\_object = p\_id\_object;

ELSIF v\_type = 'Category' THEN

SELECT cp.id\_category

INTO v\_id\_object

FROM products.t\_product as p JOIN

products.t\_product\_property as pp ON p.id = pp.id\_product JOIN

categories.t\_category\_property cp ON pp.id\_category\_property = cp.id

WHERE p.id = p\_id\_object

FOR UPDATE;

v\_id\_category = v\_id\_object;

ELSIF v\_type = 'Model' THEN

SELECT id\_model

INTO v\_id\_object

FROM products.t\_product

WHERE id = p\_id\_object

FOR UPDATE;

END IF;

SELECT c\_discount

INTO v\_discount

FROM analitics.t\_discount

WHERE c\_type = v\_type AND

c\_id\_object = v\_id\_object AND

c\_is\_active

FOR UPDATE;

IF v\_discount IS NOT NULL THEN

RETURN v\_discount;

END IF;

END LOOP;

RETURN analitics.get\_category\_discount(v\_id\_category);

END;

$$;

# ИСХОДНЫЙ КОД ФУНКЦИИ «analitics.get\_category\_discount»

CREATE FUNCTION analitics.get\_category\_discount(p\_id integer) RETURNS numeric

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

DECLARE

v\_discount numeric;

v\_cur integer;

v\_path integer[];

BEGIN

IF NOT EXISTS(SELECT 1

FROM categories.t\_category c

WHERE c.id = p\_id

FOR UPDATE) THEN

RAISE EXCEPTION 'Category with id % does not exist.', p\_id;

END IF;

SELECT string\_to\_array(c\_path, ':')::integer[]

INTO v\_path

FROM categories.t\_category

WHERE id = p\_id

FOR UPDATE;

FOREACH v\_cur IN ARRAY (

SELECT array\_agg(val ORDER BY idx DESC)

FROM unnest(v\_path) WITH ORDINALITY AS t(val, idx)

)

LOOP

SELECT c\_discount

INTO v\_discount

FROM analitics.t\_discount

WHERE c\_type = 'Category' AND

c\_id\_object = v\_cur AND

c\_is\_active;

IF v\_discount IS NOT NULL THEN

RETURN v\_discount;

END IF;

END LOOP;

RETURN NULL;

END;

$$;

# ИСХОДНЫЙ КОД ФУНКЦИИ «clients.client\_get\_discount»

CREATE FUNCTION clients.client\_get\_discount(p\_id integer, p\_max\_discount numeric DEFAULT 0.5, p\_inflection\_point numeric DEFAULT 500000, p\_slope double precision DEFAULT 0.00005) RETURNS numeric

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

DECLARE

v\_discount numeric;

BEGIN

IF NOT EXISTS(SELECT 1 FROM clients.t\_client WHERE id = p\_id FOR UPDATE) THEN

RAISE EXCEPTION 'Client with id % does not exist.', p\_id;

END IF;

SELECT p\_max\_discount / (1 + exp(-p\_slope \* (c\_money\_spent - p\_inflection\_point)))

INTO v\_discount

FROM clients.t\_client

WHERE id = p\_id

FOR UPDATE;

RETURN v\_discount;

END;

$$;

# ИСХОДНЫЙ КОД ФУНКЦИИ «analitics.get\_sales\_analytics»

CREATE FUNCTION analitics.get\_sales\_analytics(p\_start\_date timestamp with time zone, p\_end\_date timestamp with time zone, p\_group\_by\_type text DEFAULT 'Category'::text)

RETURNS TABLE(group\_name text, total\_sum\_sales numeric, total\_items\_sold bigint, average\_price numeric, median\_price numeric, price\_stddev numeric, item\_sold\_variance numeric)

SECURITY DEFINER

LANGUAGE plpgsql

AS

$$

DECLARE

query TEXT;

BEGIN

IF p\_group\_by\_type NOT IN ('Category', 'Manufacturer', 'Model', 'Product', 'ALL') THEN

RAISE EXCEPTION 'Invalid group\_by\_type. Must be Category, Manufacturer, Model, Product, or ALL.';

END IF;

query := format(

'WITH sales\_data AS (

SELECT DISTINCT

%s AS group\_name,

p.c\_price AS product\_price,

p.id

FROM storages.t\_inventory i

JOIN products.t\_product\_instance pi ON i.id\_product\_instance = pi.id

JOIN products.t\_product p ON p.id = pi.id\_product

%s

WHERE i.c\_event\_type = ''Sold''

AND i.c\_upd\_at BETWEEN %L AND %L

),

ranked\_sales AS (

SELECT

group\_name,

product\_price,

ROW\_NUMBER() OVER (PARTITION BY group\_name ORDER BY product\_price) AS row\_num,

COUNT(\*) OVER (PARTITION BY group\_name) AS total\_items

FROM sales\_data

),

median\_data AS (

SELECT

group\_name,

AVG(product\_price) AS median\_price

FROM ranked\_sales

WHERE row\_num BETWEEN (total\_items + 1) / 2 AND (total\_items + 2) / 2

GROUP BY group\_name

)

SELECT

sd.group\_name,

SUM(sd.product\_price) AS total\_sum\_sales,

COUNT(\*) AS total\_items\_sold,

AVG(sd.product\_price) AS average\_price,

md.median\_price AS median\_price,

STDDEV(sd.product\_price) AS price\_stddev,

VAR\_POP(sd.product\_price) AS item\_sold\_variance

FROM sales\_data sd

JOIN median\_data md ON sd.group\_name = md.group\_name

GROUP BY sd.group\_name, md.median\_price

ORDER BY total\_sum\_sales DESC;',

CASE

WHEN p\_group\_by\_type = 'Category' THEN 'c.c\_name'

WHEN p\_group\_by\_type = 'Manufacturer' THEN 'mfr.c\_name'

WHEN p\_group\_by\_type = 'Model' THEN 'm.c\_name'

WHEN p\_group\_by\_type = 'Product' THEN 'p.c\_name'

ELSE '''ALL''' -- For ALL, group by a constant, so there's only one group

END,

CASE

WHEN p\_group\_by\_type = 'Category' THEN 'JOIN products.t\_product\_property pp ON pp.id\_product = p.id

JOIN

categories.t\_category\_property cp ON pp.id\_category\_property = cp.id

JOIN

categories.t\_category c ON cp.id\_category = c.id'

WHEN p\_group\_by\_type = 'Manufacturer' THEN 'JOIN products.t\_model m ON p.id\_model = m.id JOIN products.t\_manufacturer mfr ON m.id\_manufacturer = mfr.id'

WHEN p\_group\_by\_type = 'Model' THEN 'JOIN products.t\_model m ON p.id\_model = m.id'

WHEN p\_group\_by\_type = 'Product' THEN '' --No join needed

ELSE '' --No join needed

END,

p\_start\_date,

p\_end\_date

);

RETURN QUERY EXECUTE query;

END;

$$;