

1. Написать представление, которое будет выводить список пользователей и их средний чек для выполненных заказов(login пользователя, средний чек)

CREATE VIEW view\_avg\_check AS

SELECT

u.login AS login,

AVG(o.price) AS avg\_price

FROM users AS u

LEFT JOIN orders AS o ON o.user\_id = u.id

GROUP BY

u.login;

2. Написать пользовательскую функцию, которая будет списывать сумму с вашего счета, если это возможно (входные параметры - login пользователя, сумма, возвращаемое значение - остаток на счете)

CREATE FUNCTION pay\_sum (login varchar, value money) RETURNS money AS $ $ declare account\_value money;

BEGIN

SELECT

account.value into account\_value

FROM

users AS user

JOIN accounts AS account ON account.user\_id = user.id

WHERE

user.login = $ 1;

IF account\_value < $ 2 then raise exception;

ELSE

UPDATE

account

SET

accout.value = account\_value - $ 2

FROM

accounts

JOIN users AS user ON account.user\_id = user.id

WHERE

user.login = $ 1;

RETURN account\_value - $ 2;

END IF;

END;

$ $ language plpgsql;  
  
3.**Написать хранимую процедуру, которая будет рассчитывать сумму заказа пользователя и скидку на заказ, исходя из стоимости доставки (если таковая имеется) и стоимости товаров в корзине.**

CREATE FUNCTION update\_order\_info (order\_id bigserial) AS $ $

declare

order\_price money,

delivery\_price money;

BEGIN

SELECT

SUM(product\_basket.price) INTO order\_price,

delivery\_record.price INTO delivery\_price

FROM

orders order

LEFT JOIN basket AS product\_basket ON product\_basket.order\_id = order.id AND

LEFT JOIN delivery\_orders AS delivery\_order ON delivery\_order.order\_id = order.id AND

LEFT JOIN delivery AS delivery\_record ON delivery\_order.delivery\_id = delivery\_record.id

WHERE

order.id = $ 1;

UPDATE

order

SET

order.price = order\_price + delivery\_price

FROM

orders

WHERE

order.id = $ 1

END;

$ $ language plpgsql;

**4. Написать триггер, который будет пересчитывать сумму заказа пользователя и скидку на заказ, при изменении корзины товаров в заказе.**

CREATE FUNCRION update\_trigger() RETURNS trigger AS $ $

BEGIN

EXECUTE PROCEDURE update\_order\_info(NEW.id)

RETURN NEW;

END;

$ $ language plpgsql;

CREATE trigger trigger\_basket\_update

AFTER INSERT OR UPDATE

OF basket ON orders FOR EACH ROW

EXECUTE PROCEDURE update\_trigger();

**5.Написать триггер, который будет устанавливать статус заказа "Оплачен",**

**если заказ полностью оплачен, и списывать с остатков на складе то количество товара, которое было в заказе.**

CREATE FUNCTION update\_orders() RETURNS trigger AS $$

DECLARE is\_paid boolean;

BEGIN

SELECT

(order.price = SUM(payment\_order.value)) INTO is\_paid

FROM

orders order

RIGHT JOIN payments\_orders payment\_order ON payment\_order.order\_id = order.id

WHERE

order.id = NEW.id

GROUP BY

order.id;

IF is\_paid then

UPDATE

order

SET

order.status = 'P'

FROM

orders

WHERE

order.id = NEW.id;

UPDATE

product

SET

product.quantity = product.quantity - paid\_product\_basket.quantity

FROM

products,

(

SELECT

product\_basket.\*

FROM

basket product\_basket

WHERE

product\_basket.order\_id = NEW.id

AND product\_basket.product\_id = product.id

) AS paid\_product\_basket;

END IF;

RETURN NEW;

END;

$$ language plpgsql;

CREATE TRIGGER trigger\_order\_update

AFTER INSERT OR UPDATE of payments\_orders on orders

FOR EACH ROW

EXECUTE PROCEDURE update\_orders();

**6.Написать запрос, используя оконную функцию, который для заданного пользователя (login) будет выводить прогресс**

**по сумме его выполненных заказов (каждая строка должна содержать 3 столбца:**

**login пользователя, сумму заказа и сумму заказов начиная с первого до текущего (прогрессирующая сумма)).**

SELECT

user.login AS login,

order.price AS price,

SUM(order.price) OVER(

partition BY order.id

) AS sum

FROM

orders AS order

LEFT JOIN users AS user ON order.user\_id = user.id

ORDER BY

order.created\_at asc;