База данных интернет магазина “AZON”

В данной базе присутствует схема “AZON”.

В которой находятся:

таблицы:

“assortment” - таблица товара,

“client” - таблица клиентов и информации о них,

“order” - таблица заказов,

”pick\_up\_points” - таблица адресов пунктов выдачи;

представления:

“available\_products” - отображает наличие товара,

“full\_inform\_order” - отображает полную информацию о заказе,

“last\_order” - отображает последний совершенный заказ,

“popular\_product” - отображает товар, в зависимости от его популярности,

“profit\_last\_y\_m\_d” - отображает прибыль за определенный период;

функции:

“add\_new\_client” - вносит сведения о новом клиенте,

“add\_new\_order\_2” - вносит сведения о новом товаре ,

“delete\_order” - удаляет информацию о заказе;

Последовательности:

“pocled” - выдает идентификаторы заказов.

Таблицы

Таблица “assortment”

Содержит в себе список товаров, их стоимость, вес и количество, находящихся на складе или продаваемых ранее.

Атрибуты:

“assortment\_id” sereal NOT NULL - идентификатор продуктов, входит в первичный ключ,

“product” varchar (30) - названия продуктов,

“price” integer - цена продуктов,

“weight” numeric (5,2) - вес продуктов,

“amount” integer - количество товара на складе.

Таблица “client”

Содержит информацию о клиентах, фамилию, имя и отчество, номера телефонов, адреса электронных почт, номера карт клиентов, идентификаторы адреса доставки в пункты выдачи или домашний адрес, куда будет доставлен товар.

Атрибуты:

“client\_id” serial NOT NULL - идентификатор клиентов, входит в первичный ключ,

“surname” varchar (30) - фамилии клиентов,

“name” varchar (30) - имена клиентов,

“patronymic” varchar (30) - отчество клиентов,

“phone\_number” varchar (10) UNIQUE - номера телефонов клиентов,

“email” varchar (50) UNIQUE - адреса электронной почты клиентов,

“bill\_number” varchar (16) UNIQUE - номера карты клиента,

“last\_delivery\_address” integer - идентификационный номер пункта выдачи, при заказе клиентом доставки на дом, это поле будет пустым, данный атрибут является ссылкой на таблицу “pick\_up\_points”, атрибут “points\_id”,

“city” varchar (40) - название города, куда будет произведена доставка, заполняется только в случае доставки на дом,

“street” varchar - название улицы, куда будет произведена доставка, заполняется только в случае доставки на дом,

“house\_number” varchar (20) - номер дома, куда будет произведена доставка, заполняется только в случае доставки на дом,

“apartment\_number” varchar - номер квартиры или дополнительная информация о месте доставки, заполняется только в случае доставки на дом.

Таблица “order”

Содержит дату заказа, идентификаторы клиента, продукта и пункта выдачи, количество продукта, а, так же, информацию, куда выполняется доставка.

Атрибуты:

“order\_id” int NOT NULL - идентификатор заказа,

“order\_date” date NOT NULL - дата создания заказа,

“client\_id” integer - идентификатор клиента,

“assortment\_id” int - идентификатор продукта,

“amount” int NOT NULL - количество заказанного товара,

“points\_id” int - идентификатор пункта выдачи, в случае доставки на дом, это поле будет пустым,

“home\_delivery” bool - флаг доставки на дом, если значение true - доставка на дом, false - доставка в пункт выдачи.

Таблица “pick\_up\_points”

Содержит идентификаторы пунктов выдачи, города, улицы, номер домов и дополнительную информацию.

Атрибуты:

“points\_id” serial NOT NULL - идентификатор пунктов выдачи, входит в первичный ключ,

“city” varchar (20) - название города,

“street” varchar (40) - название улицы,

“house\_number” varchar (10) - номер дома,

“notes” varchar - дополнительная информация о местонахождении пункта выдачи, если такой информации нет, это поле будет пустым.

Представления

Представление “available\_products”

Выводит наименование продуктов, которые доступны для заказа, то есть те, что есть на складе.

CREATE OR REPLACE VIEW "AZON".available\_products

AS SELECT a.product,

a.amount

FROM "AZON".assortment a

WHERE a.amount > 0;

Представление “full\_inform\_order”

Выводит полную информацию о заказе, а именно: идентификатор заказа, фамилию, имя и отчество заказчика, название продукта, количество, цену и вес заказа в зависимости от его количества и адрес доставки, включающий в себя город, улицу, дом и квартиру или дополнительную информацию.

CREATE OR REPLACE VIEW "AZON".full\_inform\_order

AS SELECT o.order\_id,

o.order\_date,

(((c.surname::text || ' '::text) || c.name::text) || ' '::text) || c.patronymic::text AS addressee,

-- Конкатенация строк, будет использоваться здесь и далее

a.product,

o.amount,

CASE

WHEN o.amount = 1 THEN a.price

ELSE a.price \* o.amount

END AS price,

/\* если количество товара больше 1, выводим стоимость умноженную на количество заказанного товара \*/

CASE

WHEN o.amount = 1 THEN a.weight

ELSE round(a.weight \* o.amount::numeric, 2)

END AS weight,

-- если количество товара больше 1, выводим вес умноженный на количество заказанного товара

CASE

WHEN o.home\_delivery = true THEN (((((c.city::text || ', '::text) || c.street::text) || ', '::text) || c.house\_number::text) || ', '::text) || c.apartment\_number::text

ELSE (((((pup.city::text || ', '::text) || pup.street::text) || ', '::text) || pup.house\_number::text) || ', '::text) || COALESCE(pup.notes, ''::character varying)::text

END AS address

/\* Здесь происходит проверка на флаг доставки на дом, если флаг true, то выводится адрес, написанный клиентом, берется из таблицы "client", если false - выводится адрес идентификатора пунктра выдачи, адрес берется из таблицы "assortment"\*/

FROM "AZON"."order" o

JOIN "AZON".client c ON o.client\_id = c.client\_id

JOIN "AZON".assortment a ON o.assortment\_id = a.assortment\_id

LEFT JOIN "AZON".pick\_up\_points pup ON o.points\_id = pup.points\_id;

Представление “last\_order”

Выводит информацию о последнем совершенном заказе. Выводит идентификатор заказа, фамилию, имя и отчество заказчика, название продукта, количество и адрес доставки, включающий в себя город, улицу, дом и квартиру или дополнительную информацию.

CREATE OR REPLACE VIEW "AZON".last\_order

AS SELECT o.order\_id,

o.order\_date,

(((c.surname::text || ' '::text) || c.name::text) || ' '::text) || c.patronymic::text AS addressee,

a.product,

o.amount,

CASE

WHEN o.home\_delivery = true THEN (((((c.city::text || ', '::text) || c.street::text) || ', '::text) || c.house\_number::text) || ', '::text) || c.apartment\_number::text

ELSE (((((pup.city::text || ', '::text) || pup.street::text) || ', '::text) || pup.house\_number::text) || ', '::text) || COALESCE(pup.notes, ''::character varying)::text

END AS address

FROM "AZON"."order" o

JOIN "AZON".client c ON o.client\_id = c.client\_id

JOIN "AZON".assortment a ON o.assortment\_id = a.assortment\_id

LEFT JOIN "AZON".pick\_up\_points pup ON o.points\_id = pup.points\_id

WHERE o.order\_date = (( SELECT max(o\_1.order\_date) AS max

/\* вложенным запросом определяем дату последнего заказа и опираясь на эту дату делаем выборку\*/

FROM "AZON"."order" o\_1));

Представление “popular\_product”

Выводит список товаров и частоту заказов этих товаров.

CREATE OR REPLACE VIEW "AZON".popular\_product

AS SELECT a.product,

sum(o.amount) AS sum

FROM "AZON"."order" o

RIGHT JOIN "AZON".assortment a ON o.assortment\_id = a.assortment\_id

GROUP BY a.product

ORDER BY (sum(o.amount)) DESC;

Представление “rofit\_last\_y\_m\_d”

Выводит прибыль за последний год, за последний месяц и за последний день.

CREATE OR REPLACE VIEW "AZON".profit\_last\_y\_m\_d

AS WITH profit\_month AS (

SELECT sum(a.price) AS profit\_month,

date\_part('month'::text, o.order\_date) AS month\_date,

1 AS id

FROM "AZON"."order" o

JOIN "AZON".assortment a ON a.assortment\_id = o.assortment\_id

WHERE o.order\_date >= (( SELECT date\_trunc('month'::text, max(o2.order\_date)::timestamp with time zone) AS date\_trunc

FROM "AZON"."order" o2))

GROUP BY (date\_part('month'::text, o.order\_date))

), profit\_year AS (

SELECT sum(a.price) AS profit\_year,

date\_part('year'::text, o.order\_date) AS year\_date,

1 AS id

FROM "AZON"."order" o

JOIN "AZON".assortment a ON a.assortment\_id = o.assortment\_id

WHERE o.order\_date >= (( SELECT date\_trunc('year'::text, max(o2.order\_date)::timestamp with time zone) AS date\_trunc

FROM "AZON"."order" o2))

GROUP BY (date\_part('year'::text, o.order\_date))

), profit\_day AS (

SELECT sum(a.price) AS profit\_day,

date\_part('day'::text, o.order\_date) AS day\_date,

1 AS id

FROM "AZON"."order" o

JOIN "AZON".assortment a ON a.assortment\_id = o.assortment\_id

WHERE o.order\_date >= (( SELECT date\_trunc('day'::text, max(o2.order\_date)::timestamp with time zone) AS date\_trunc

FROM "AZON"."order" o2))

GROUP BY (date\_part('day'::text, o.order\_date))

)

SELECT y.profit\_year,

y.year\_date,

m.profit\_month,

m.month\_date,

d.profit\_day,

d.day\_date

FROM profit\_year y

JOIN profit\_month m ON m.id = y.id

JOIN profit\_day d ON d.id = y.id;

Функции

Функция “add\_new\_client”(text, text, text, text, text, text, integer, text, text, text, text)

Функция выполняет регистрацию нового клиента. Данная функция принимает 10 переменных типа text и 1 типа integer. Первые 6 переменных - это фамилия, имя и отчество клиента, номер телефона, адрес электронной почты и номер карты, 7 переменная является идентификатором пункта выдачи, следующие 4 переменных являются названием города, улицы, номером дома и квартиры. Функция проверяет входящие данные на разного рода ошибки такие как: пустое поле, на уникальность данных, неправильное количество знаков или пробелы внутри переменной. Далее данные вносятся в таблицу “client”.

CREATE OR REPLACE FUNCTION "AZON".add\_new\_client(text, text, text, text, text, text, integer, text, text, text, text)

RETURNS void

LANGUAGE plpgsql

AS $function$

declare

p\_surname text := $1;

p\_name text := $2;

p\_patronymic text := $3;

p\_phone\_number text := $4;

p\_email text := $5;

p\_bill\_number text := $6;

p\_last\_delivery\_address int := $7;

p\_city text := $8;

p\_street text := $9;

p\_house\_number text := $10;

p\_apartment\_number text := $11;

begin

if p\_surname is null or p\_surname like ''

then raise exception 'Это поле не может быть пустым.' USING HINT = 'Поле "Фамилия" не может быть пустым.';

/\* Если клиент будет оставляет поле пустым, он не сможет зарегистрироваться, будет выводиться

следующий текст ошибки \*/

elseif

p\_name is null or p\_name like ''

then raise exception 'Это поле не может быть пустым.' USING HINT = 'Поле "Имя" не может быть пустым.';

elseif

p\_phone\_number is null or p\_phone\_number like ''

then raise exception 'Это поле не может быть пустым.' USING HINT = 'Пожалуйста, введите номер телефона.';

elseif

p\_phone\_number = (select phone\_number from client where p\_phone\_number = phone\_number)

then raise exception 'Введенные данные уже используются.' USING HINT = 'Пожалуйста, введите другой номер телефона.';

/\* Если клиент вводит данные, на которые уже был ранее зарегистрирован пользователь, будет выводиться следующий текст ошибки \*/

elseif

p\_email is null or p\_email like ''

then raise exception 'Это поле не может быть пустым.' USING HINT = 'Поле "e-mail" не может быть пустым.';

elseif

p\_email = (select email from client where p\_email = email)

then raise exception 'Введенные данные уже используются.' USING HINT = 'Пожалуйста, введите другой адрес электронной почты.';

elseif

p\_bill\_number is null or p\_bill\_number like ''

then raise exception 'Это поле не может быть пустым.' USING HINT = 'Пожалуйста, введите Ваш платежный номер.';

elseif

p\_bill\_number = (select bill\_number from client where p\_bill\_number = bill\_number)

then raise exception 'Введенные данные уже используются.' USING HINT = 'Пожалуйста, введите другой платежный номер.';

elseif

char\_length(p\_phone\_number) <> 10

then raise exception 'Неверное количестко знаков.' USING HINT = 'Номер телефона должен содержать 10 символов.';

-- Если клиент вводит более или менее 10 знаков, будет выводиться следующий текст ошибки

elseif

char\_length(p\_bill\_number) <> 16

then raise exception 'Неверное количестко знаков.' USING HINT = 'Платежный номер должен содержать 16 символов.';

elseif

p\_phone\_number ~\* ' '

then raise exception 'Недопустимый знак.' USING HINT = 'Номер телефона не может содержать пробелы.';

-- Если клиент вводит пробел вместо символа, будет выводиться следующий текст ошибки

elseif

p\_bill\_number ~\* ' '

then raise exception 'Недопустимый знак.' USING HINT = 'Платежный номер не может содержать пробелы.';

elseif

p\_email ~\* ' '

then raise exception 'Недопустимый знак.' USING HINT = 'Адрес электронной почты не может содержать пробелы.';

end if;

insert into client (surname, "name", patronymic, phone\_number, email, bill\_number, last\_delivery\_address, city, street, house\_number, apartment\_number)

values (p\_surname, p\_name, p\_patronymic, p\_phone\_number, p\_email, p\_bill\_number, p\_last\_delivery\_address, p\_city, p\_street, p\_house\_number, p\_apartment\_number);

end;

$function$

;

Функция “add\_new\_order\_2(date, integer, integer [])

Функция создает новый заказ. Данная функция принимает переменные типа date, integer и двумерный массив типа integer. Первая переменная является датой заказа, вторая переменная идентификационным номером клиента, в первый элемент массива записывается идентификатор товара, во второй элемент - количество этого товара. Далее мы присваиваем из последовательности идентификатор заказа, берем идентификационный номер пункта выдачи из таблицы “client” и относительно его наличия выставляем флаг доставки на дом. Далее в цикле мы присваиваем значения переменным идентификатора товара и его количества, взятые из цикла. Так как затем мы будем вычитать количество заказанного товара из таблицы “assortment”, мы проверяем имеется ли достаточное количество товара на складе, если товра недостаточно, заказ совершить не удастся. Если товара достаточное количество, в цикле мы вносим данные в таблицу “order” и вычитаем заказанное количество товара из таблицы “assortment”.

CREATE OR REPLACE FUNCTION "AZON".add\_new\_order\_2(p\_date date, p\_client integer, array\_a integer[])

RETURNS void

LANGUAGE plpgsql

AS $function$

declare

p\_id\_order int;

p\_assort int;

p\_amount int;

p\_points int;

p\_home\_deliv bool;

begin

select nextval('pocled') into p\_id\_order;

/\* из последовательности “pocled” берется значение и записывается в переменную “p\_id\_order”, которая в дальнейшем будет записываться в идентификатор заказа \*/

select last\_delivery\_address into p\_points from client where p\_client = client\_id;

-- из таблицы “client” берется идентификатор номера пункта выдачи

case

when p\_points is null then p\_home\_deliv := true;

else p\_home\_deliv := false;

end case;

/\* при наличии идентификатора номера пункта выдачи, ставится флаг доставки на дом - true, в

противоположном случае - false \*/

for i in 1..array\_length(array\_a, 2) loop

p\_assort := array\_to\_string(array\_a[1:1][i:i], ',')::int;

p\_amount := array\_to\_string(array\_a[2:2][i:i], ',')::int;

/\* Из массива последовательно вынимаются значения, из первого элемента берется значение идентификатора товара, из второго количество этого товара и записываются в переменные “p\_assort” и “p\_amount” соответственно. Также при вынимании из массива значений, приводим эти значения в тип данных integer. \*/

if ((select amount from assortment where p\_assort = assortment\_id) - p\_amount) < 0

then raise exception 'Товар отсутствует.' USING HINT = 'Нет товара на складе.';

end if;

/\* Если в таблице “assortment” недостаточное количество товара для заказа, функция выдаст ошибку и выведет следующий текст \*/

insert into "order" (order\_id, order\_date, client\_id, assortment\_id, amount, points\_id, home\_delivery)

values (p\_id\_order, p\_date, p\_client, p\_assort, p\_amount, p\_points, p\_home\_deliv);

-- Вводим заначения в таблицу “order”

update assortment

set amount = amount - p\_amount

where assortment\_id = p\_assort;

-- Вычитаем из таблицы “assortment” количество заказанного товара

end loop;

end;

$function$

;

Функция “delete\_order”(p\_id\_order integer, p\_assort int)

Функция удаляет заказ с определенным товаром. Функция принимает 2 значения integer: первое - идентификатор заказа, второе - идентификатор товара. Берем значение количество данного товара из таблицы “order”. Затем функция удаляет из таблицы “order” заказ с указанным идентификатором заказа и товара. После удаления функция добавляет количетсво товара в таблицу “assortment”.

CREATE OR REPLACE FUNCTION "AZON".delete\_order(p\_id\_order integer, p\_assort int)

RETURNS void

LANGUAGE plpgsql

AS $function$

declare

p\_amount int;

begin

select amount into p\_amount from "order" where p\_id\_order = order\_id and p\_assort = assortment\_id;

delete from "order"

where order\_id = p\_id\_order and p\_assort = assortment\_id;

update assortment

set amount = amount + p\_amount

where assortment\_id = p\_assort;

end;

$function$

;