Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.03 (ПРИКЛАДНАЯ ИНФОРМАТИКА)**

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К КУРСОВОЙ РАБОТЕ

по дисциплине «Базы Данных»

НА ТЕМУ:
«Заказ продуктов»

Студент ИУ6-43Б (Группа) _____ (Подпись, дата) В.К. Залыгин (И.О. Фамилия)

Руководитель _____ (Подпись, дата) М.А. Скворцова (И.О. Фамилия)

2024 г.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В последние годы активно развиваются сети продуктовых магазинов, в основе которых лежит модель даркстороов (“магазины без покупателей”) с мгновенной доставкой продуктов. Такая модель становится удобна конечному потребителю, так как позволяет в течение короткого промежутка времени получить необходимые продукты и не тратить время на самостоятельный их закуп. Так как бизнес-модель основана на сверхбыстрой доставке (не более 30 минут с момента оформления заказа), предприятия ее реализующие у предъявляют высокие требования к уровню организованности и эффективности оконечных точек сети доставки – даркстороов. Создаваемая база данных предназначена для обеспечения эффективной работы сети даркстороов с мгновенной доставкой.

Актуальность данной системы заключается в возможности эффективной организации операционной деятельности оконечных точек сетей с вышеописанной бизнес-моделью. Система позволяет контролировать ассортимент продуктов, состояние заказов, доставок, поставок со склада, смены персонала оконечной точки, а также проводить анализ операционной деятельности с целью подсчета бизнес-метрик и повышения эффективности предприятия. Данная система позволяет повысить качество и скорость обслуживания пользователей, что является важнейшими показателями для бизнеса.

Целью работы является создание системы, которая обеспечит эффективную операционную деятельностью оконечных точек сети даркстороов.

Для достижения поставленной цели необходимо решить следующие задачи:

- Проведение анализа предметной области заказа продуктов, выделение основных сущностей и процессов, определение требований к базе данных;

- Проектирование базы данных, разработка инфологической и даталогической модели базы данных;
- Написание скрипта для создания базы данных со всеми инструментами обеспечения консистентности хранящейся информации;
- SQL-запросы для получения статистической информации.

Для реализации используется реляционная база данных PostgerSQL 16, графический клиент DBeaver 23.2.3, язык Python 3.10, среда разработки Visual Studio Code.

1 ПРОЕКТИРОВАНИЕ

1.1 Анализ предметной области

В сфере заказа продуктов существуют различные бизнес-процессы, поддерживающие возможность выбора продуктов и их доставки до клиента. Ниже рассмотрены основные аспекты данной предметной области.

Предметной областью данного проекта является управление операционной деятельностью предприятия, позволяющего клиенту осуществлять заказы продуктов. База данных предоставляет возможность организации заказов, сборки и доставки списка продуктов на основе актуального ассортимента, учета продуктов и их своевременного списания по достижению срока хранения, рабочего цикла на предприятиях и распределения людей на роли.

Функционал базы данных включает в себя:

- Запись и учет данных о поставках партий продуктов в оконечные точки, включая количество поставляемых продуктов и сроки хранения партии;
- Запись и учет данных о клиенте, включая имя, фамилию, контактную информацию, сохранённые адреса;
- Хранение и формирование, обработка информации о заказах пользователя, состояниях заказа, доставки и сборки заказанных продуктов;
- Запись и учет информации о сотрудниках и выставляемых сменах;
- Запись и учет данных об ассортименте даркстеров на основе прихода (поставок) и ухода (заказов);
- Списание партий продуктов с истекшим сроком хранения;
- Создание заданий для сборщиков и доставщиков на основе состояний о текущих заказах.

Данная база данных является актуальной и эффективной, поскольку позволяет управлять информацией о заказах, поставках и сборках,

обеспечивая консистентность данных и быстрый доступ к необходимой информации для управления процессом осуществления заказа продуктов.

1.2 Выделение сущностей

Для проектирования базы данных необходимо выделить элементы предметной области. В рамках анализа можно выделить следующие сущности:

1) Клиент (Client):

Данная сущность содержит информацию о клиентах предприятия. Атрибуты: уникальный номер, персональные данные клиентов, такие как имя, фамилия, сохранённые адреса доставки, контактный телефон, почта.

2) Заказ (Order):

Данная сущность хранит информацию о заказе продуктов. Атрибуты: уникальный номер, состояние заказа, клиент, сделавших заказ, сборки продуктов, адрес доставки, время создания, время начала доставки, время закрытия заказа, дарстор, из которого осуществляется доставка.

3) Продукт (Product):

Хранит информацию о продукте, продающемся предприятием. Атрибуты: уникальный номер, название, описание, изображение продукта, цена за упаковку.

4) Сборка

Хранит информацию о сборке продуктов для заказов. Атрибуты: продукт, заказ, количество продуктов, время создания начала сборки, время окончания сборки, статус.

5) Доставка

Хранит информацию о доставках заказов. Атрибуты: уникальный номер, доставляемый заказ, доставляющий курьер, время создания, статус доставки.

6) Сотрудник

Хранит информацию о сотруднике, работающем в сети. Атрибуты: уникальный номер, имя, фамилия, должность.

7) Смена

Хранит информацию о смене сотрудников дарксторов. Атрибуты: уникальный номер, сотрудник, время начала и окончания смены, магазин, в котором проходит смена.

8) Поставка

Хранит информацию о поставках партий продуктов в даркстоpы. Атрибуты: уникальный номер, магазин, продукт, время доставки, окончание срока годности партии, статус поставки, количество единиц продукта.

9) Ассортимент

Хранит информацию об остатках продуктов в определенном даркстоpе. Атрибуты: продукт, магазин, остаток (количество).

10) Даркстор

Хранит информацию об оконечной точке сети. Атрибуты: уникальный номер, адрес, техническое имя, владелец.

11) Расположение продукта

Хранит информацию о расположении продукта в рамках одного даркстора для обеспечения быстрого поиска продуктов и сборки заказа. Атрибуты: продукт, магазин, описание расположения (буквенно-численный код, обозначающий местоположение продукта)

Главной сущностью в системе является Заказ.

1.3 Проектирование инфологической модели базы данных.

Исходя из сущностей и их свойств, определенных в пункте 1.2, возможно построить инфологическую модель базы данных. Это необходимо для дальнейшего корректного проектирования базы данных. На рисунке 1 представлена инфологическая модель БД.

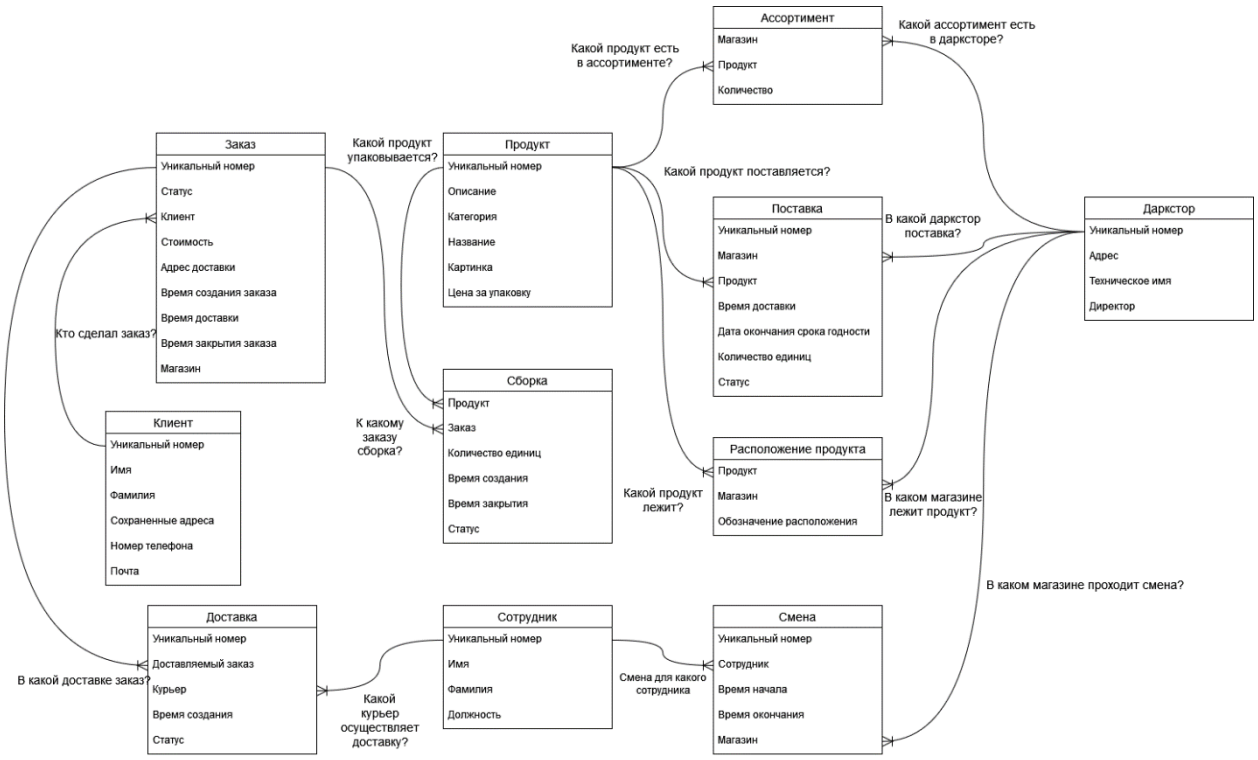


Рисунок 1 – Инфологическая модель БД

При разработке инфологической модели были выделены основные процессы, через которые происходит взаимодействие сущностей.

1.4 Проектирование даталогической модели базы данных.

На основании инфологической модели можно построить даталогическую модель, необходимую для написания корректного скрипта создания базы данных и входящих в нее таблиц.

Из пунктов 1.3 и 1.4 следуют следующие основные функции, которые должна реализовывать разрабатываемая база данных:

1) Управление и хранение данных о клиентах;
2) Управление и хранение данных о продуктах, ассортименте и сопутствующих сущностях;
3) Управление и хранение данных о сотрудниках;
4) Управление и хранение необходимой дополнительной информации;
5) Поддержание консистентного состояния данных на основе триггеров.

На основании инфологической модели, и выделенных из нее функций, была построена даталогическая модель, представленная на рисунке 2.

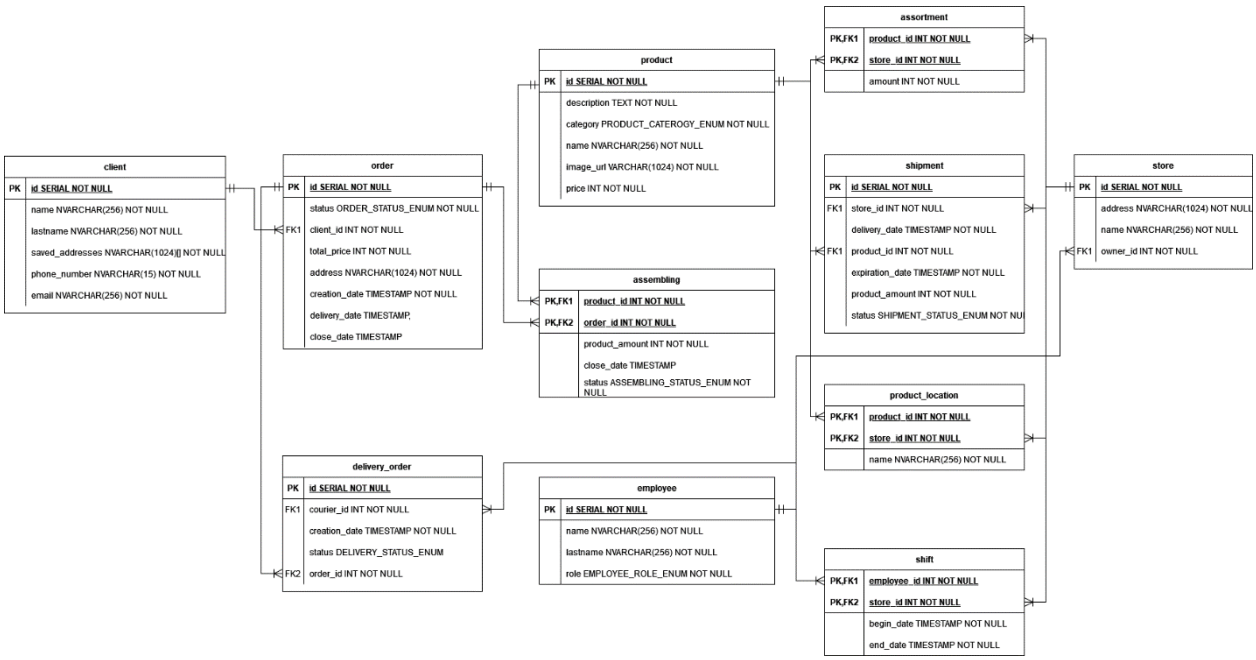


Рисунок 2 – Даталогическая модель БД

Для того, чтобы пояснить назначение полей в каждой из таблиц, а связи между ними были более понятными, ниже приведено описание каждой из таблиц и ее полей.

Таблица 1 – Описание таблицы “clients”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор	Целочисленный последовательный
name	Фамилия	Строковый
lastname	Имя	Строковый
saved_addresses	Сохранённые адреса	Массив строковый
phone_number	Телефон	Строковый
email	Почта	Строковый

Таблица 2 – Описание таблицы “order”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор каждого заказа	Целочисленный последовательный
status	Статус	Перечисление статусов
client_id	Идентификатор клиента	Целочисленный
total_price	Общая стоимость заказа	Целочисленный
address	Адрес доставки для заказа	Строковый
creation_date	Дата и время создания заказа	Дата и время
delivery_date	Дата и время готовности заказа для доставки	Дата и время
close_date	Дата и время закрытия заказа	Дата и время
store_id	Идентификатор магазина, где был размещен заказ	Целочисленный

Таблица 3 – Описание таблицы “assembling”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор каждой доставки	Целочисленный последовательный
order_id	Идентификатор связанного заказа	Целочисленный
courier_id	Идентификатор курьера, выполняющего доставку	Целочисленный
status	Статус доставки	Перечисление статусов

Таблица 4 – Описание таблицы “delivery”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор каждого сотрудника	Целочисленный последовательный
name	Имя сотрудника	Строковый
lastname	Фамилия сотрудника	Строковый
role	Роль сотрудника	Перечисление статусов

Таблица 5 – Описание таблицы “employee”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор каждого сотрудника	Целочисленный последовательный
name	Имя сотрудника	Строковый
lastname	Фамилия сотрудника	Строковый
role	Роль сотрудника	Перечисление статусов

Таблица 6 – Описание таблицы “shift”

Название поля	Предназначение поля	Тип данных
employee_id	Идентификатор сотрудника, работающего сменой	Целочисленный
store_id	Идентификатор магазина, где работает сотрудник	Целочисленный
begin_date	Дата и время начала смены	Дата и время
end_date	Дата и время конца смены	Дата и время

Таблица 7 – Описание таблицы “product_location”

Название поля	Предназначение поля	Тип данных
product_id	Идентификатор продукта	Целочисленный
store_id	Идентификатор магазина	Целочисленный
description	Описание расположения продукта	Строковый

Таблица 8 – Описание таблицы “product”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор продукта	Целочисленный последовательный
name	Название продукта	Строковый
description	Описание продукта	Строковый
category	Категория продукта	Перечисление статусов
image_url	URL-адрес изображения продукта	Строковый
price	Цена продукта	Целочисленный

Таблица 9 – Описание таблицы “assortment”

Название поля	Предназначение поля	Тип данных
store_id	Идентификатор магазина	Целочисленный
product_id	Идентификатор продукта	Целочисленный
amount	Количество продукта в ассортименте определенного магазина	Целочисленный

Таблица 10 – Описание таблицы “shipment”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор поставки	Целочисленный последовательный
store_id	Идентификатор магазина, куда поставляется товар	Целочисленный
product_id	Идентификатор продукта в поставке	Целочисленный
delivery_date	Дата и время доставки поставки	Дата и время
expiration_date	Дата и время истечения срока годности партии продуктов	Дата и время
product_amount	Количество продуктов в поставке	Целочисленный
status	Статус поставки	Перечисление статусов

Таблица 11 – Описание таблицы “store”

Название поля	Предназначение поля	Тип данных
id	Уникальный идентификатор магазина	Целочисленный последовательный
address	Адрес магазина	Строковый
name	Название магазина	Строковый
owner_id	Идентификатор владельца магазина	Целочисленный

2 РЕАЛИЗАЦИЯ

2.1 Написание скрипта создания базы данных

При задании схемы базы данных первоочередно создаются типы данных, используемых в таблицах. В листинге 1 представлен скрипт, создающий необходимые типы-перечисления для таблиц.

Листинг 1 – Создание перечислений

```
DROP TYPE IF EXISTS ORDER_STATUS_ENUM;
CREATE TYPE ORDER_STATUS_ENUM AS ENUM (
     'assembling', 'assembled', 'delivering', 'closed',
'cancelled'
);

DROP TYPE IF EXISTS DELIVERY_STATUS_ENUM;
CREATE TYPE DELIVERY_STATUS_ENUM AS ENUM (
     'scheduled', 'on_the_way', 'closed'
);

DROP TYPE IF EXISTS EMPLOYEE_ROLE_ENUM;
CREATE TYPE EMPLOYEE_ROLE_ENUM AS ENUM (
     'courier', 'assembler', 'manager'
);

DROP TYPE IF EXISTS PRODUCT_CATEGORY_ENUM;
CREATE TYPE PRODUCT_CATEGORY_ENUM AS ENUM (
     'alcohol', 'bakery', 'bread', 'cheese',
'fish_and_seafood', 'fruit', 'juice', 'lemonade', 'meat',
'milk_and_egg', 'sausage', 'snack', 'water', 'yogurt'
);

DROP TYPE IF EXISTS SHIPMENT_STATUS_ENUM;
CREATE TYPE SHIPMENT_STATUS_ENUM AS ENUM (
     'on_the_way', 'delivered', 'accepted', 'run_out',
'expired'
);
```

Типы с окончанием STATUS_ENUM перечисляют множество значений соответствующей сущности. Тип PRODUCT_CATEGORY_ENUM перечисляет категории продуктов.

Листинг 2 содержит операторы создания таблиц.

Листинг 2 – Скрипт создания таблиц

```
CREATE TABLE IF NOT EXISTS client (
     id SERIAL PRIMARY KEY,
     "name" VARCHAR(256) NOT NULL,
     lastname VARCHAR(256) NOT NULL,
     saved_addresses VARCHAR(1024)[] NOT NULL,
     phone_number VARCHAR(256) NOT NULL,
     email VARCHAR(256) NOT NULL
);

CREATE TABLE IF NOT EXISTS "order" (
     id SERIAL,
     "status" ORDER_STATUS_ENUM NOT NULL DEFAULT 'assembling',
     client_id INT NOT NULL,
     total_price INT NOT NULL CHECK (total_price >= 0) DEFAULT
100,
     "address" VARCHAR(1024) NOT NULL,
     creation_date TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
     delivery_date TIMESTAMP,
     close_date TIMESTAMP,
     store_id INT NOT NULL,

     PRIMARY KEY (id),

     CHECK (delivery_date = null or delivery_date >=
creation_date),
     CHECK (close_date = null or delivery_date != null and
close_date >= delivery_date)
);

CREATE TABLE IF NOT EXISTS delivery (
     id SERIAL,
     order_id INT NOT NULL,
     courier_id INT NOT NULL,
     "status" DELIVERY_STATUS_ENUM,

     PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS employee (
     id SERIAL,
     "name" VARCHAR(256) NOT NULL,
     lastname VARCHAR(256) NOT NULL,
     "role" EMPLOYEE_ROLE_ENUM NOT NULL,

     PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS product (
     id SERIAL,
```

```
     "name" VARCHAR(256) NOT NULL,
     "description" TEXT NOT NULL,
     category PRODUCT_CATEGORY_ENUM NOT NULL,
     image_url VARCHAR(1024) NOT NULL,
     price INT NOT NULL,

     PRIMARY KEY (id),

     CHECK (price >= 0)
);

CREATE TABLE IF NOT EXISTS shipment (
     id SERIAL,
     store_id INT NOT NULL,
     product_id INT NOT NULL,
     delivery_date TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
     expiration_date TIMESTAMP NOT NULL,
     product_amount INT NOT NULL,
     "status" SHIPMENT_STATUS_ENUM NOT NULL DEFAULT
'delivered',

     PRIMARY KEY (id),

     CHECK (product_amount >= 0)
);

CREATE TABLE IF NOT EXISTS assortment (
     store_id INT NOT NULL,
     product_id INT NOT NULL,
     amount INT NOT NULL,

     PRIMARY KEY (store_id, product_id),

     CHECK (amount >= 0)
);

CREATE TABLE IF NOT EXISTS assembling (
     product_id INT NOT NULL,
     order_id INT NOT NULL,
     product_amount INT NOT NULL CHECK (product_amount > 0),
     creation_date TIMESTAMP NOT NULL DEFAULT
CURRENT_TIMESTAMP,
     close_date TIMESTAMP,

     PRIMARY KEY (product_id, order_id),

     CHECK (close_date = null or close_date > creation_date)
);

CREATE TABLE IF NOT EXISTS store (
     id SERIAL,
```

```
     "address" VARCHAR(1024) NOT NULL,
     "name" VARCHAR(256) NOT NULL,
     "owner_id" INT NOT NULL,

     PRIMARY KEY (id)
);

CREATE TABLE IF NOT EXISTS product_location (
     product_id INT NOT NULL,
     store_id INT NOT NULL,
     "description" VARCHAR(256) NOT NULL,

     PRIMARY KEY (product_id, store_id)
);

CREATE TABLE IF NOT EXISTS shift (
     employee_id INT NOT NULL,
     store_id INT NOT NULL,
     begin_date TIMESTAMP NOT NULL,
     end_date TIMESTAMP NOT NULL,

     PRIMARY KEY (employee_id, store_id),

     CHECK (end_date > begin_date)
);
```

После создания таблиц следует определения внешних ключей. Листинг 3 содержит скрипт задания внешних ключей в соответствии с даталогической моделью.

Листинг 3 – Создание внешних ключей

```
ALTER TABLE "order" ADD CONSTRAINT fk_order_client FOREIGN KEY
(client_id) REFERENCES client;
ALTER TABLE "order" ADD CONSTRAINT fk_order_store FOREIGN KEY
(store_id) REFERENCES store;

ALTER TABLE delivery ADD CONSTRAINT fk_delivery_courier FOREIGN
KEY (courier_id) REFERENCES employee;
ALTER TABLE delivery ADD CONSTRAINT fk_delivery_order FOREIGN
KEY (order_id) REFERENCES "order";

ALTER TABLE shipment ADD CONSTRAINT fk_shipment_store FOREIGN
KEY (store_id) REFERENCES store;
ALTER TABLE shipment ADD CONSTRAINT fk_shipment_product FOREIGN
KEY (product_id) REFERENCES product;

ALTER TABLE store ADD CONSTRAINT fk_store_owner FOREIGN KEY
("owner") REFERENCES employee;
```

```
ALTER TABLE assembling ADD CONSTRAINT fk_assembling_product
FOREIGN KEY (product_id) REFERENCES product;
ALTER TABLE assembling ADD CONSTRAINT fk_assembling_order
FOREIGN KEY (order_id) REFERENCES "order";

ALTER TABLE product_location ADD CONSTRAINT
fk_product_location_product FOREIGN KEY (product_id) REFERENCES
product;
ALTER TABLE product_location ADD CONSTRAINT
fk_product_location_store FOREIGN KEY (store_id) REFERENCES
store;

ALTER TABLE shift ADD CONSTRAINT fk_shift_employee FOREIGN KEY
(employee_id) REFERENCES employee;
ALTER TABLE shift ADD CONSTRAINT fk_shift_store FOREIGN KEY
(store_id) REFERENCES store;

ALTER TABLE assortment ADD CONSTRAINT fk_assortment_store
FOREIGN KEY (store_id) REFERENCES store;
ALTER TABLE assortment ADD CONSTRAINT fk_assortment_product
FOREIGN KEY (product_id) REFERENCES product;
```

Для обеспечения согласованности данных необходимы триггеры. Листинг 4 содержит скрипт для создания триггеров.

Листинг 4 – Создание триггеров

```
CREATE OR REPLACE FUNCTION
set_order_assembled_on_assembly_ready_func()
RETURNS TRIGGER AS
$$
BEGIN
     IF
          (SELECT count(*)
           FROM assembling
           WHERE assembling.order_id = NEW.order_id AND
assembling.close_date = null)
          = 0
     THEN
          UPDATE "order"
          SET "status" = 'assembled'
          WHERE "order".id = NEW.order_id;
     END IF;
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER set_order_assembled_on_assembly_ready
     AFTER UPDATE
     ON assembling
     FOR EACH ROW
```

```
     EXECUTE PROCEDURE
set_order_assembled_on_assembly_ready_func();

CREATE OR REPLACE FUNCTION
update_assortment_on_shipment_accepted_func()
RETURNS TRIGGER AS
$$
BEGIN
     IF
          OLD."status" != NEW."status" AND NEW."status" =
'accepted'
     THEN
          IF (
               SELECT count(*) FROM assortment WHERE
assortment.store_id = NEW.store_id AND assortment.product_id =
NEW.product_id
          ) = 0 THEN
               INSERT INTO assortment (store_id, product_id,
amount)
               VALUES (NEW.store_id, NEW.product_id, 0);
          END IF;
          UPDATE assortment
          SET amount = amount + NEW.product_amount
          WHERE assortment.store_id = NEW.store_id AND
assortment.product_id = NEW."product_id";
     END IF;
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER
update_assortment_on_shipment_accepted
     AFTER UPDATE
     ON shipment
     FOR EACH ROW
     EXECUTE PROCEDURE
update_assortment_on_shipment_accepted_func();

CREATE OR REPLACE FUNCTION
decrease_product_amount_on_new_assembling_func()
RETURNS TRIGGER AS
$$
DECLARE
     store_id_var INT := (
          SELECT "order".store_id FROM "order" WHERE "order".id
= NEW.order_id
     );
BEGIN
     UPDATE assortment
     SET amount = amount - NEW.product_amount
     WHERE assortment.store_id = store_id_var AND
assortment.product_id = NEW.product_id;
```

```
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER
decrease_product_amount_on_new_assembling
     BEFORE INSERT
     ON assembling
     FOR EACH ROW
     EXECUTE PROCEDURE
decrease_product_amount_on_new_assembling_func();

CREATE OR REPLACE FUNCTION
set_shipment_run_out_on_zero_amount_func()
RETURNS TRIGGER AS
$$
BEGIN
     IF
          NEW.product_amount = 0 AND NEW.status = 'accepted'
     THEN
          UPDATE shipment
          SET "status" = 'run_out'
          WHERE shipment.id = NEW.id;
     END IF;
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER set_shipment_run_out_on_zero_amount
     AFTER UPDATE
     ON shipment
     FOR EACH ROW
     EXECUTE PROCEDURE
set_shipment_run_out_on_zero_amount_func();

CREATE OR REPLACE FUNCTION
set_order_timestamp_on_status_change_func()
RETURNS TRIGGER AS
$$
BEGIN
     IF
          NEW."status" = 'delivering' AND OLD."status" !=
'delivering'
     THEN
          UPDATE "order"
          SET delivery_date = CURRENT_TIMESTAMP
          WHERE "order".id = NEW.id;
     END IF;
     IF
          (NEW."status" = 'closed' AND OLD."status" !=
'closed') OR
```

```
          (NEW."status" = 'cancelled' AND OLD."status" !=
'cancelled')
     THEN
          UPDATE "order"
          SET close_date = CURRENT_TIMESTAMP
          WHERE "order".id = NEW.id;
     END IF;
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER set_order_timestamp_on_status_change
     AFTER UPDATE
     ON "order"
     FOR EACH ROW
     EXECUTE PROCEDURE
set_order_timestamp_on_status_change_func();

CREATE OR REPLACE FUNCTION
decrease_shipment_product_amount_on_assortment_decrease_func()
RETURNS TRIGGER AS
$$
DECLARE
     remaining INT := NEW.amount;
     shipment_r RECORD;
BEGIN
     IF OLD.amount > NEW.amount THEN
          FOR shipment_r IN
               (SELECT *
               FROM shipment
               WHERE shipment.store_id = NEW.store_id AND
shipment.product_id = NEW.product_id
               ORDER BY expiration_date ASC)
          LOOP
               IF remaining > 0 THEN
                    IF remaining > shipment_r.product_amount
THEN
                         UPDATE shipment
                         SET product_amount = 0
                         WHERE shipment.id = shipment_r.id;
                         remaining := remaining -
shipment_r.product_amount;
                    ELSE
                         UPDATE shipment
                         SET product_amount = product_amount -
remaining
                         WHERE shipment.id = shipment_r.id;
                         remaining := 0;
                    END IF;
               END IF;
          END LOOP;
     END IF;
```

```
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER
decrease_shipment_product_amount_on_assortment_decrease
     AFTER UPDATE
     ON assortment
     FOR EACH ROW
     EXECUTE PROCEDURE
decrease_shipment_product_amount_on_assortment_decrease_func();

return_products_to_assortment_on_order_cancel_func()
RETURNS TRIGGER AS
$$
DECLARE
     assembling_r RECORD;
BEGIN
     IF NEW."status" = 'cancelled' AND OLD."status" !=
'cancelled' THEN
          FOR assembling_r IN
               (SELECT *
               FROM assembling
               WHERE assembling.order_id = NEW.id)
          LOOP
               UPDATE assortment
               SET amount = amount +
assembling_r.product_amount
               WHERE assortment.store_id = NEW.store_id AND
assortment.product_id = assembling_r.product_id;
          END LOOP;
     END IF;
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER
return_products_to_assortment_on_order_cancel
     AFTER UPDATE
     ON "order"
     FOR EACH ROW
     EXECUTE PROCEDURE
return_products_to_assortment_on_order_cancel_func();

CREATE OR REPLACE FUNCTION upd_price_on_assembling_func()
RETURNS TRIGGER AS
$$
DECLARE
     old_price INT;
     new_price INT;
BEGIN
```

```
     old_price := (SELECT total_price FROM "order" WHERE
"order".id = NEW.order_id);
     UPDATE "order"
     SET total_price = total_price + (SELECT product.price FROM
product WHERE product.id = NEW.product_id)
     WHERE "order".id = NEW.order_id;
     new_price := (SELECT total_price FROM "order" WHERE
"order".id = NEW.order_id);
     IF new_price >= 1000 AND old_price < 1000 THEN
          UPDATE "order"
          SET total_price = total_price - 100
          WHERE "order".id = NEW.order_id;
     END IF;
     RETURN NEW;
END;
$$
LANGUAGE 'plpgsql';
CREATE OR REPLACE TRIGGER upd_price_on_assembling
     AFTER INSERT
     ON assembling
     FOR EACH ROW
     EXECUTE PROCEDURE upd_price_on_assembling_func();
```

Пояснение семантики каждого триггера представлено в списке ниже.

- upd_price_on_assembling обновляет итоговую цену заказа на основе добавляемых продуктов;

- return_products_to_assortment_on_order_cancel возвращает продукты в ассортимент из отменных заказов;

- decrease_shipment_product_amount_on_assortment_decrease при уменьшении количества товара в ассортименте уменьшает остаток в соответствующих поставках. Необходим для корректного учета поставок;

- set_order_timestamp_on_status_change_func обновлении статуса заказа проставляет соответствующую временную метку;

- set_shipment_run_out_on_zero_amount обновляет статус поставки при нулевом остатке;

- decrease_product_amount_on_new_assembling уменьшает остаток продукта в магазине при его добавлении в заказ;

- update_assortment_on_shipment_accepted увеличивает остаток в ассортименте при окончании разгрузки поставки;

- set_order_assembled_on_assembly_ready если заказ собран, то меняет ему статус.

Наконец необходимы запланированные функции, которые также обеспечивают корректную работу базы данных. Поскольку используемая база данных не имеет функциональности, позволяющей задать расписание вызовов функций, используя только инструментарий сервера базы данных, предполагается некоторый внешний источник, который будет осуществлять их вызов по расписанию. Такие функции имеет смысл заранее определить в рамках базы данных для их последующей оптимизации движком базы данных. Листинг 5 представляет скрипт для создания функций.

Листинг 5 – Создание функций

```
CREATE OR REPLACE FUNCTION check_and_mark_expired_shipments()
RETURNS void
LANGUAGE 'plpgsql' AS
$$
DECLARE
shipment_r RECORD;
BEGIN
FOR shipment_r IN
(SELECT *
FROM shipment
WHERE shipment.expiration_date < CURRENT_TIMESTAMP AND
shipment.status = 'accepted')
LOOP
UPDATE shipment
SET status = 'expired'
WHERE shipment.id = shipment_r.id;

UPDATE assortment
SET amount = amount - shipment_r.product_amount
WHERE assortment.store_id = shipment_r.store_id AND
assortment.product_id = shipment_r.product_id;
END LOOP;
END;
$$;

CREATE OR REPLACE FUNCTION
check_ready_to_delivery_orders_and_assign_couriers()
RETURNS void
LANGUAGE 'plpgsql' AS
$$
DECLARE
order_r RECORD;
courier_id_r INT;
```

```
BEGIN
FOR order_r IN
(SELECT *
FROM "order"
WHERE "order"."status" = 'assembled')
LOOP
courier_id_r := (
SELECT employee_shift.employee_id
FROM employee_shift
WHERE employee_shift."role" = 'courier' AND
employee_shift.store_id = order_r.store_id
LIMIT 1
);
UPDATE "order"
SET "status" = 'delivering'
WHERE "order".id = order_r.id;

INSERT INTO delivery (order_id, courier_id, "status")
VALUES (order_r.id, courier_id_r, 'scheduled');
END LOOP;
END;
$$;
```

Семантика функций описывается в списке ниже.

- check_ready_to_delivery_orders_and_assign_couriers назначает курьеров, находящихся на смене, для собранных заказов;

- check_and_mark_expired_shipments помечает просроченные поставки и убирает просроченные продукты из ассортимента.

Наконец необходимо определить представления и роли для повышения удобства работы с базой данных. Представления и роли создаются в листинге 6.

Листинг 6 – Создание ролей и представлений

```
CREATE ROLE dba;
GRANT ALL PRIVILEGES ON DATABASE postgres TO dba;

CREATE ROLE manager;
GRANT ALL PRIVILEGES ON ALL TABLES IN SCHEMA public TO manager;
GRANT ALL PRIVILEGES ON ALL FUNCTIONS IN SCHEMA public TO
manager;

CREATE ROLE "user";
GRANT SELECT, UPDATE ON TABLE "order", client TO "user";
GRANT SELECT ON TABLE product, assortment, store, client_order,
client_assortment TO "user";
```

```
CREATE ROLE courier;
GRANT UPDATE ON TABLE delivery TO courier;
GRANT SELECT ON TABLE courier_delivery TO courier;

CREATE ROLE assembler;
GRANT SELECT, UPDATE ON TABLE assembling, shipment TO
assembler;
GRANT SELECT ON TABLE assembler_assembling TO assembler;

CREATE OR REPLACE VIEW client_order AS (
     WITH order_product AS (
          SELECT assembling.order_id AS order_id,
product."name" AS product_name, assembling.product_amount AS
product_amount
          FROM assembling
          JOIN product ON product.id = assembling.product_id
     )
     SELECT "order".id, "order"."status", "order".client_id,
"order".total_price, "order".address, "order".creation_date,
"order".close_date, array_agg(order_product.product_name) as
products, array_agg(order_product.product_amount) as amounts
     FROM "order"
     JOIN order_product ON order_product.order_id = "order".id
     GROUP BY "order".id
);

CREATE OR REPLACE VIEW client_assortment AS (
     SELECT store.address, product.name, assortment.amount
     FROM assortment
     JOIN product ON product.id = assortment.product_id
     JOIN store ON store.id = assortment.store_id
);

CREATE OR REPLACE VIEW courier_delivery AS (
     SELECT "order".id, "order"."address", employee.id as
courier_id, employee.name, employee.lastname
     FROM delivery, "order", employee
     WHERE delivery.order_id = "order".id AND
delivery.courier_id = employee.id AND delivery."status" !=
'closed'
);

CREATE OR REPLACE VIEW assembler_assembling AS (
     SELECT "order".store_id, assembling.order_id,
assembling.product_amount, product_location."description"
     FROM "order", assembling, product, product_location
     WHERE
          "order".id = assembling.order_id AND
          assembling.product_id = product.id AND
          product_location.product_id = product.id AND
          product_location.store_id = "order".store_id
);
```

```
CREATE OR REPLACE VIEW employee_shift AS (
     SELECT employee.id AS employee_id, employee."role",
shift.begin_date, shift.end_date, shift.store_id
     FROM employee, shift
     WHERE shift.employee_id = employee.id AND shift.end_date >
CURRENT_TIMESTAMP
);
```

Для повышения безопасности создаются роли и выдаются соответствующие разрешения.

- dba – администратор базы данных;

- manager – выполняет административную работу в магазине;

- user – клиент предприятия;

- assembler – сборщик заказов;

- courier – доставщик заказов.

Для повышения удобства пользования базой данных созданы представления.

- client_order агрегирует данные заказов в удобный вид с указанием списка продуктов;

- client_assortment агрегирует данные по ассортименту в разных магазинах в удобный вид;

- courier_delivery собирает для курьеров всю необходимую инфомрацию о доставках;

- assembler_assembling собирает всю необходимую информацию для сборщиков;

- employee_shift отображает сотрудников, находящихся на смене в данный момент.

2.2 Заполнение базы данных

Для заполнения базы данных был написан скрипт на языке Python с использованием стандартной библиотеки языка, а также сторонних библиотек psycogp2, faker. Полученные данные согласованы, соответствуют предметной области и предполагаемым типам данных. Начало скрипта показано в листинге 6. Полный листинг находится в Приложении А.

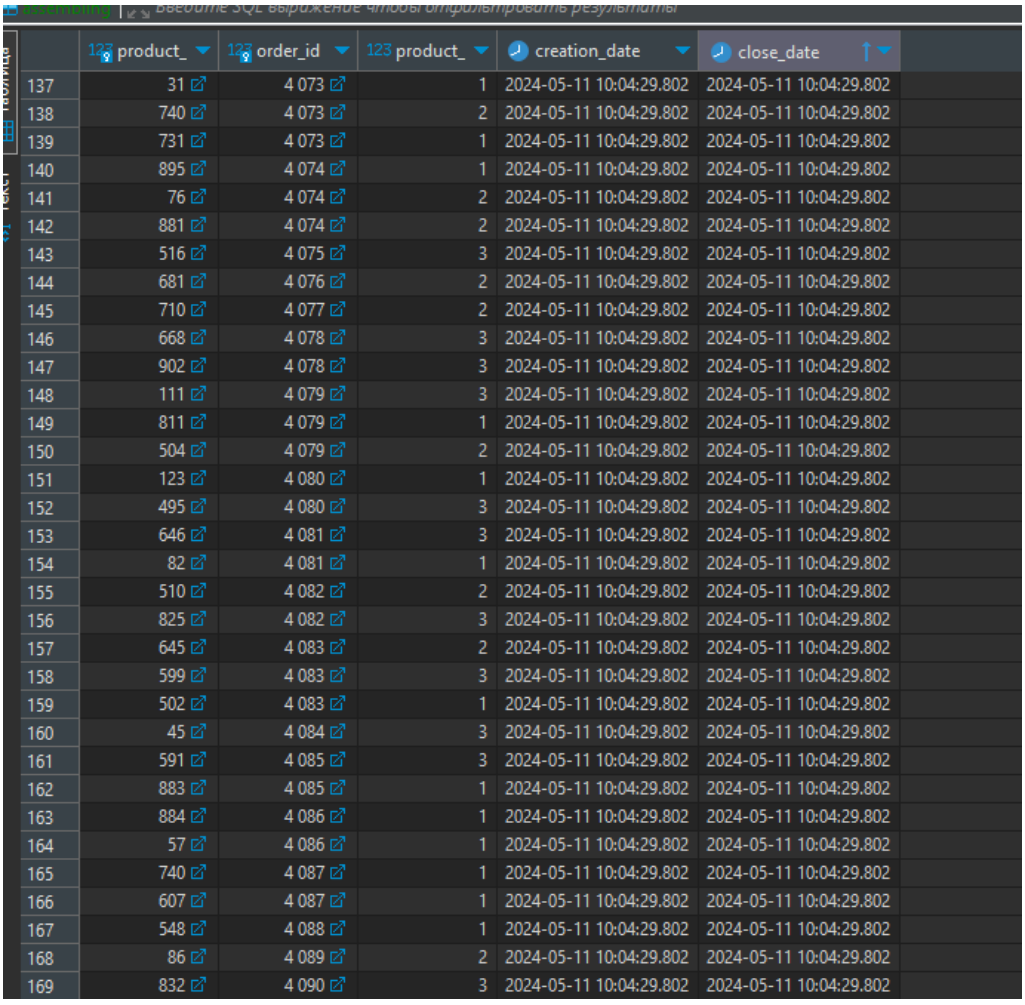
Листинг 6 – первые 90 строчек скрипта заполнения базы данных

```
#!/bin/pypy

from faker import Faker

try:
    import psycopg2 as ps
    from psycopg2.extras import execute_batch
except:
    import psycopg2cffi as ps
    from psycopg2cffi.extras import execute_batch
from itertools import groupby
import random
import json
from datetime import datetime, timezone, timedelta
import sys

CLIENT_AMOUNT = 150
PRODUCT_AMOUNT = 931
STORE_AMOUNT = 1000
ORDER_AMOUNT = 1_000_000
OWNER_IDS = range(STORE_AMOUNT * 0 + 1, STORE_AMOUNT * 1 + 1)
MANAGER_IDS = range(STORE_AMOUNT * 1 + 1, STORE_AMOUNT * 2 + 1)
COURIER_IDS = range(STORE_AMOUNT * 2 + 1, STORE_AMOUNT * 3 + 1)
ASSEMBER_IDS = range(STORE_AMOUNT * 3 + 1, STORE_AMOUNT * 4 +
1)

random.seed(40)
Faker.seed(42)
fake = Faker(locale="ru_RU")
conn = ps.connect(
    dbname="postgres",
    user="postgres",
    password="12345678",
    host="localhost",
    port="5433",
)
cur = conn.cursor()


def ignore_on_fail(f):
```

```
    def g(*args):
        try:
            return f(*args)
        except Exception as e:
            print(f"continue on err: {e}", file=sys.stderr)

    return g


@ignore_on_fail
def clients():
    data = []
    for _ in range(CLIENT_AMOUNT):
        name = fake.first_name()
        lastname = fake.last_name()
        saved_addresses = [fake.address() for _ in
range(fake.random_int(0, 5))]
        phone_number = fake.phone_number()
        email = fake.email()
        data.append((name, lastname, saved_addresses,
phone_number, email))
    execute_batch(
        cur,
        """
            INSERT INTO client (\"name\", lastname,
saved_addresses, phone_number, email)
            VALUES (%s, %s, %s, %s, %s)
        """,
        data,
    )
    print(f"clients records {len(data)}")
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/insert/clients.json", "w"
    ) as f:
        json.dump(data, f)
    return data


@ignore_on_fail
def products():
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/data/product.json", "r"
    ) as f:
        data = json.load(f)
    data = [(p["name"], p["name"], p["caterogy"], p["img"],
p["price"]) for p in data]
    print("insert into db")
    execute_batch(
        cur,
        """
```

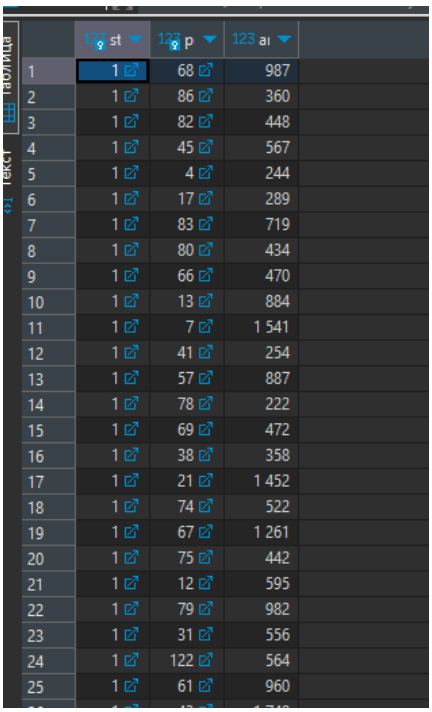
```
            INSERT INTO product ("name", "description",
category, image_url, price)
            VALUES (%s, %s, %s::PRODUCT_CATEGORY_ENUM, %s, %s)
        """,
        data,
    )
    print(f"product records {len(data)}")
    return data
```

Для проверки корректности заполнения таблиц необходимо выполнить SELECT-запросы к каждой таблице. Результаты обращений представлены на рисунках 3-13.



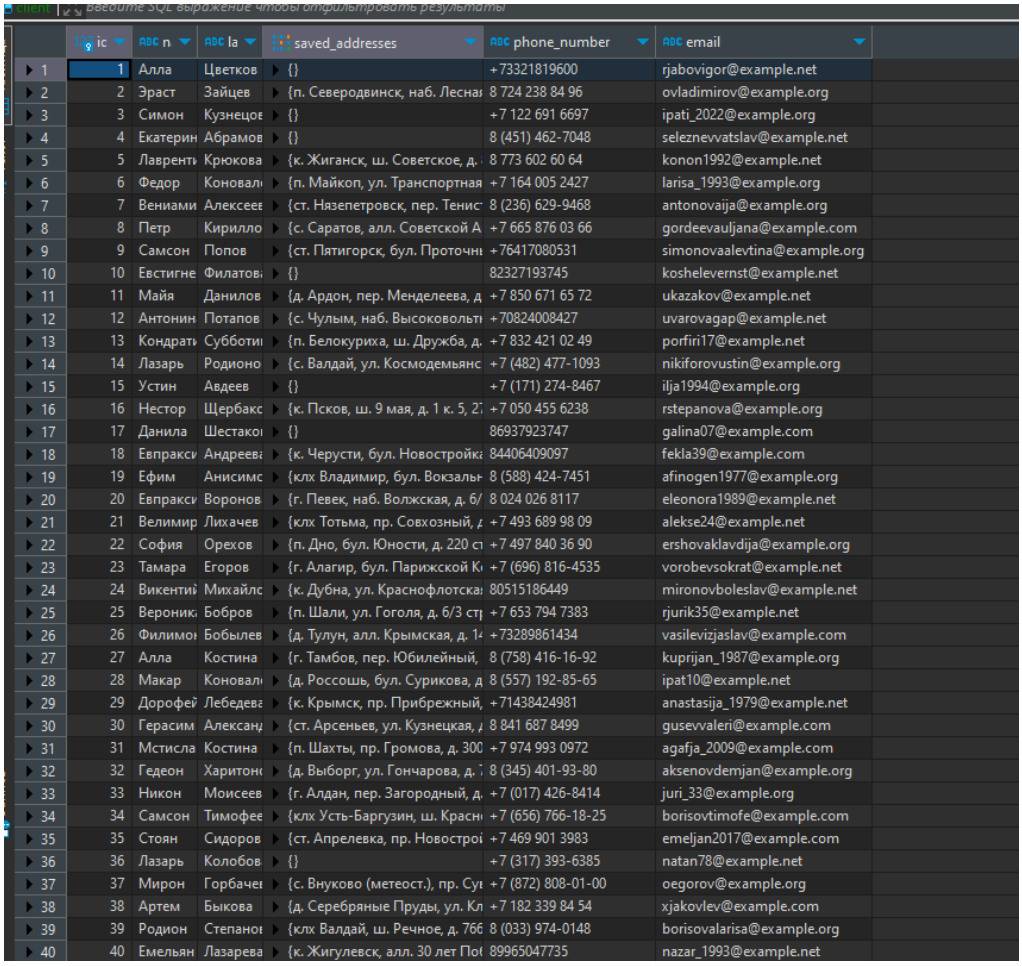
	product_	order_id	product_	creation_date	close_date
137	31	4 073	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
138	740	4 073	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
139	731	4 073	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
140	895	4 074	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
141	76	4 074	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
142	881	4 074	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
143	516	4 075	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
144	681	4 076	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
145	710	4 077	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
146	668	4 078	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
147	902	4 078	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
148	111	4 079	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
149	811	4 079	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
150	504	4 079	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
151	123	4 080	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
152	495	4 080	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
153	646	4 081	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
154	82	4 081	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
155	510	4 082	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
156	825	4 082	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
157	645	4 083	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
158	599	4 083	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
159	502	4 083	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
160	45	4 084	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
161	591	4 085	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
162	883	4 085	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
163	884	4 086	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
164	57	4 086	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
165	740	4 087	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
166	607	4 087	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
167	548	4 088	1	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
168	86	4 089	2	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802
169	832	4 090	3	2024-05-11 10:04:29.802	2024-05-11 10:04:29.802

Рисунок 3 – Результат SELECT-запроса к таблице “assembling”



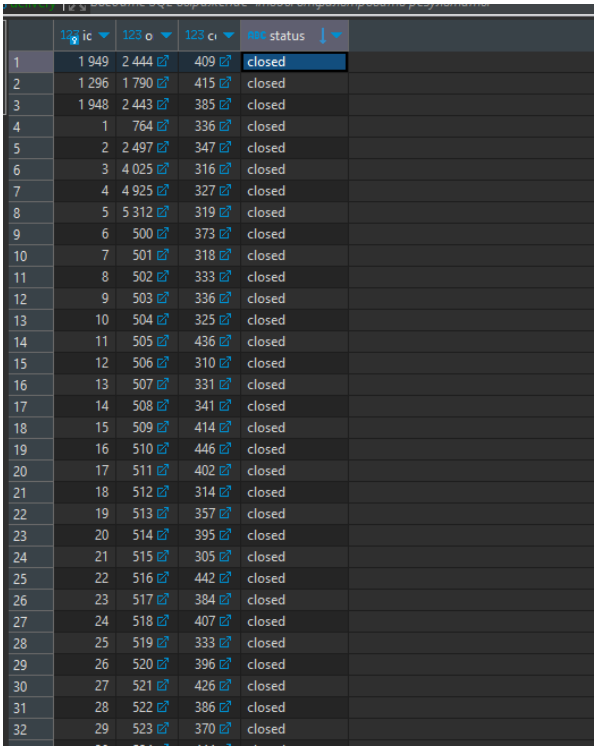
	st	p	a
1	1	68	987
2	1	86	360
3	1	82	448
4	1	45	567
5	1	4	244
6	1	17	289
7	1	83	719
8	1	80	434
9	1	66	470
10	1	13	884
11	1	7	1 541
12	1	41	254
13	1	57	887
14	1	78	222
15	1	69	472
16	1	38	358
17	1	21	1 452
18	1	74	522
19	1	67	1 261
20	1	75	442
21	1	12	595
22	1	79	982
23	1	31	556
24	1	122	564
25	1	61	960

Рисунок 4 – Результат SELECT-запроса к таблице “assortment”



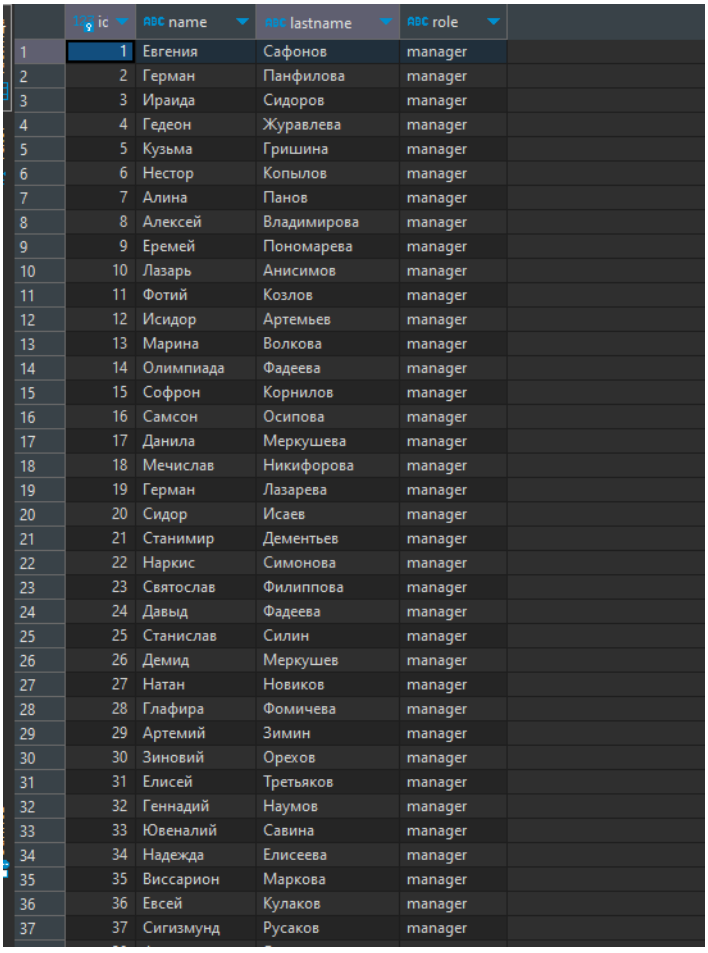
	ic	n	la	saved_addresses	phone_number	email
1	1	Алла	Цветков	{}	+73321819600	rjabovigor@example.net
2	2	Эраст	Зайцев	{п. Северодвинск, наб. Лесна	8 724 238 84 96	ovladimirov@example.org
3	3	Симон	Кузнецов	{}	+7 122 691 6697	ipati_2022@example.org
4	4	Екатерин	Абрамов	{}	8 (451) 462-7048	seleznevvatslav@example.net
5	5	Лаврент	Крюкова	{к. Жиганск, ш. Советское, д.	8 773 602 60 64	konon1992@example.net
6	6	Федор	Коновал	{п. Майкоп, ул. Транспортная	+7 164 005 2427	larisa_1993@example.org
7	7	Вениами	Алексеев	{ст. Нязепетровск, пер. Тенис	8 (236) 629-9468	antonovaija@example.org
8	8	Петр	Кирилло	{с. Саратов, алл. Советской А	+7 665 876 03 66	gordeevauljana@example.com
9	9	Самсон	Попов	{ст. Пятигорск, бул. Проточны	+76417080531	simonovaalevtina@example.org
10	10	Евстигне	Филатов	{}	82327193745	koshelevernst@example.net
11	11	Майя	Данилов	{д. Ардон, пер. Менделеева, д	+7 850 671 65 72	ukazakov@example.net
12	12	Антонин	Потапов	{с. Чулым, наб. Высоковольт	+70824008427	uvarovagap@example.net
13	13	Кондрат	Субботи	{п. Белокуриха, ш. Дружба, д.	+7 832 421 02 49	porfiri17@example.net
14	14	Лазарь	Родионо	{с. Валдай, ул. Космодемьянс	+7 (482) 477-1093	nikiforovustin@example.org
15	15	Устин	Авдеев	{}	+7 (171) 274-8467	ilja1994@example.org
16	16	Нестор	Щербако	{к. Псков, ш. 9 мая, д. 1 к. 5, 2	+7 050 455 6238	rstepanova@example.org
17	17	Данила	Шестако	{}	86937923747	galina07@example.com
18	18	Евпракси	Андреева	{к. Черусти, бул. Новостройка	84406409097	fekla39@example.com
19	19	Ефим	Анисимс	{клх Владимир, бул. Вокзальн	8 (588) 424-7451	afinogen1977@example.org
20	20	Евпракси	Воронов	{г. Певек, наб. Волжская, д. 6/	8 024 026 8117	eleonora1989@example.net
21	21	Велимир	Лихачев	{клх Тотьма, пр. Совхозный, д	+7 493 689 98 09	alekse24@example.net
22	22	София	Орехов	{п. Дно, бул. Юности, д. 220 ст	+7 497 840 36 90	ershovaklavdija@example.org
23	23	Тамара	Егоров	{г. Алагир, бул. Парижской К	+7 (696) 816-4535	vorobevsokrat@example.net
24	24	Викентий	Михайло	{к. Дубна, ул. Краснофлотска	80515186449	mironovboleslav@example.net
25	25	Вероник	Бобров	{п. Шали, ул. Гоголя, д. 6/3 ст	+7 653 794 7383	rjurik35@example.net
26	26	Филимон	Бобылев	{д. Тулун, алл. Крымская, д. 1	+73289861434	vasilevizjaslav@example.com
27	27	Алла	Костина	{г. Тамбов, пер. Юбилейный,	8 (758) 416-16-92	kuprijan_1987@example.org
28	28	Макар	Коновал	{д. Россошь, бул. Сурикова, д	8 (557) 192-85-65	ipat10@example.net
29	29	Дорофей	Лебедева	{к. Крымск, пр. Прибрежный,	+71438424981	anastasija_1979@example.net
30	30	Герасим	Александ	{ст. Арсеньев, ул. Кузнецкая, д	8 841 687 8499	gusevvaleri@example.com
31	31	Мстисла	Костина	{п. Шахты, пр. Громова, д. 300	+7 974 993 0972	agafja_2009@example.com
32	32	Гедеон	Харитон	{д. Выборг, ул. Гончарова, д. 7	8 (345) 401-93-80	aksenovdemjan@example.org
33	33	Никон	Моисеев	{г. Алдан, пер. Загородный, д.	+7 (017) 426-8414	juri_33@example.org
34	34	Самсон	Тимофее	{клх Усть-Баргузин, ш. Красн	+7 (656) 766-18-25	borisovtimofe@example.com
35	35	Стоян	Сидоров	{ст. Апрелевка, пр. Новостроі	+7 469 901 3983	emeljan2017@example.com
36	36	Лазарь	Колобов	{}	+7 (317) 393-6385	natan78@example.net
37	37	Мирон	Горбачев	{с. Внуково (метеост.), пр. Су	+7 (872) 808-01-00	oegorov@example.org
38	38	Артем	Быкова	{д. Серебряные Пруды, ул. Кл	+7 182 339 84 54	xjakovlev@example.com
39	39	Родион	Степано	{клх Валдай, ш. Речное, д. 76	8 (033) 974-0148	borisovalarisa@example.org
40	40	Емельян	Лазарева	{к. Жигулевск, алл. 30 лет По	89965047735	nazar_1993@example.net

Рисунок 5 – Результат SELECT-запроса к таблице “client”



	ic	o	c	status
1	1 949	2 444	409	closed
2	1 296	1 790	415	closed
3	1 948	2 443	385	closed
4	1	764	336	closed
5	2	2 497	347	closed
6	3	4 025	316	closed
7	4	4 925	327	closed
8	5	5 312	319	closed
9	6	500	373	closed
10	7	501	318	closed
11	8	502	333	closed
12	9	503	336	closed
13	10	504	325	closed
14	11	505	436	closed
15	12	506	310	closed
16	13	507	331	closed
17	14	508	341	closed
18	15	509	414	closed
19	16	510	446	closed
20	17	511	402	closed
21	18	512	314	closed
22	19	513	357	closed
23	20	514	395	closed
24	21	515	305	closed
25	22	516	442	closed
26	23	517	384	closed
27	24	518	407	closed
28	25	519	333	closed
29	26	520	396	closed
30	27	521	426	closed
31	28	522	386	closed
32	29	523	370	closed

Рисунок 6 – Результат SELECT-запроса к таблице “delivery”



	ic	name	lastname	role
1	1	Евгения	Сафонов	manager
2	2	Герман	Панфилова	manager
3	3	Ираида	Сидоров	manager
4	4	Гедеон	Журавлева	manager
5	5	Кузьма	Гришина	manager
6	6	Нестор	Копылов	manager
7	7	Алина	Панов	manager
8	8	Алексей	Владимирова	manager
9	9	Еремей	Пономарева	manager
10	10	Лазарь	Анисимов	manager
11	11	Фотий	Козлов	manager
12	12	Исидор	Артемьев	manager
13	13	Марина	Волкова	manager
14	14	Олимпиада	Фадеева	manager
15	15	Софрон	Корнилов	manager
16	16	Самсон	Осипова	manager
17	17	Данила	Меркушева	manager
18	18	Мечислав	Никифорова	manager
19	19	Герман	Лазарева	manager
20	20	Сидор	Исаев	manager
21	21	Станимир	Дементьев	manager
22	22	Наркис	Симонова	manager
23	23	Святослав	Филиппова	manager
24	24	Давыд	Фадеева	manager
25	25	Станислав	Силин	manager
26	26	Демид	Меркушев	manager
27	27	Натан	Новиков	manager
28	28	Глафира	Фомичева	manager
29	29	Артемий	Зимин	manager
30	30	Зиновий	Орехов	manager
31	31	Елисей	Третьяков	manager
32	32	Геннадий	Наумов	manager
33	33	Ювеналий	Савина	manager
34	34	Надежда	Елисеева	manager
35	35	Виссарион	Маркова	manager
36	36	Евсей	Кулаков	manager
37	37	Сигизмунд	Русаков	manager

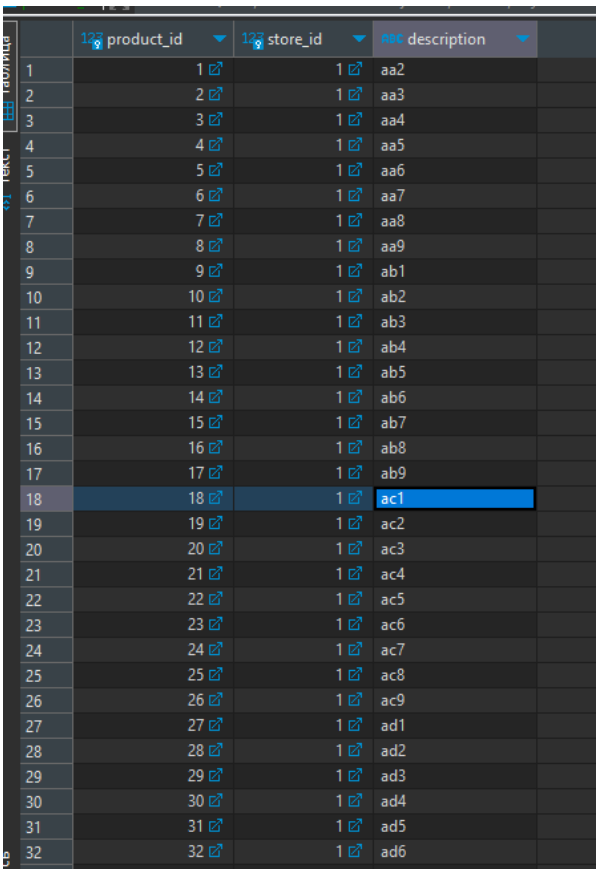
Рисунок 7 – Результат SELECT-запроса к таблице “employee”

	id	status	client_id	total_price	address	creation_date	delivery_date	close_date	store_id
1	8 434	assembling	103	431	к. Щелково, ш. Га	2024-05-11 10:04:29.802	[NULL]	[NULL]	45
2	764	delivering	46	189	д. Нефедова, бул.	2024-05-11 10:04:29.802	2024-05-11 10:07:48.661	[NULL]	36
3	2 497	delivering	66	297	п. Салават, алл. Ц	2024-05-11 10:04:29.802	2024-05-11 10:07:48.661	[NULL]	47
4	4 025	delivering	27	299	п. Яшкуль, пр. Тр	2024-05-11 10:04:29.802	2024-05-11 10:07:48.661	[NULL]	16
5	4 925	delivering	83	510	с. Ряжск, ш. Заозе	2024-05-11 10:04:29.802	2024-05-11 10:07:48.661	[NULL]	27
6	5 312	delivering	132	219	п. Грозный, ш. Ле	2024-05-11 10:04:29.802	2024-05-11 10:07:48.661	[NULL]	19
7	6 991	assembling	53	153	с. Усть-Баргузин,	2024-05-11 10:04:29.802	[NULL]	[NULL]	30
8	6 992	assembling	128	189	к. Тихвин, ш. Нов	2024-05-11 10:04:29.802	[NULL]	[NULL]	79
9	6 993	assembling	75	378	п. Владивосток, п	2024-05-11 10:04:29.802	[NULL]	[NULL]	6
10	6 994	assembling	114	606	с. Талдом, пер. М	2024-05-11 10:04:29.802	[NULL]	[NULL]	48
11	6 995	assembling	78	315	ст. Чита, алл. Лес	2024-05-11 10:04:29.802	[NULL]	[NULL]	55
12	6 996	assembling	42	531	д. Владикавказ, бу	2024-05-11 10:04:29.802	[NULL]	[NULL]	145
13	6 997	assembling	117	818	к. Сухиничи, алл.	2024-05-11 10:04:29.802	[NULL]	[NULL]	8
14	6 998	assembling	47	627	ст. Юровск, пер. Г	2024-05-11 10:04:29.802	[NULL]	[NULL]	17
15	6 999	assembling	13	808	к. Кыштым, алл. У	2024-05-11 10:04:29.802	[NULL]	[NULL]	117
16	7 000	assembling	90	457	с. Мончегорск, пе	2024-05-11 10:04:29.802	[NULL]	[NULL]	86
17	7 001	assembling	125	599	д. Челюскин, наб.	2024-05-11 10:04:29.802	[NULL]	[NULL]	85
18	7 002	assembling	132	164	д. Меренга, ш. Ст	2024-05-11 10:04:29.802	[NULL]	[NULL]	84
19	7 003	assembling	49	155	с. Усть-Калманка,	2024-05-11 10:04:29.802	[NULL]	[NULL]	80
20	7 004	assembling	109	858	д. Мостовской, ш	2024-05-11 10:04:29.802	[NULL]	[NULL]	20
21	7 005	assembling	111	896	д. Наро-Фоминск,	2024-05-11 10:04:29.802	[NULL]	[NULL]	8
22	7 006	assembling	106	545	с. Ярославль, ул. І	2024-05-11 10:04:29.802	[NULL]	[NULL]	129
23	7 007	assembling	68	289	д. Касимов, пр. Ге	2024-05-11 10:04:29.802	[NULL]	[NULL]	122
24	7 008	assembling	49	409	п. Сочи, наб. Свої	2024-05-11 10:04:29.802	[NULL]	[NULL]	143
25	7 009	assembling	53	516	клх Коломна, пр.	2024-05-11 10:04:29.802	[NULL]	[NULL]	12
26	7 010	assembling	38	319	с. Урай, пр. Совет	2024-05-11 10:04:29.802	[NULL]	[NULL]	6
27	7 011	assembling	61	613	г. Касимов, наб. К	2024-05-11 10:04:29.802	[NULL]	[NULL]	30
28	7 012	assembling	71	249	д. Вытегра, пер. Н	2024-05-11 10:04:29.802	[NULL]	[NULL]	100
29	7 013	assembling	40	507	ст. Энгельс, бул. Г	2024-05-11 10:04:29.802	[NULL]	[NULL]	63
30	7 014	assembling	3	199	ст. Елатьма, наб. А	2024-05-11 10:04:29.802	[NULL]	[NULL]	84
31	7 015	assembling	6	603	ст. Шелехов, пер.	2024-05-11 10:04:29.802	[NULL]	[NULL]	63
32	7 016	assembling	57	545	ст. Холмск, бул. Л	2024-05-11 10:04:29.802	[NULL]	[NULL]	67

Рисунок 8 – Результат SELECT-запроса к таблице “order”

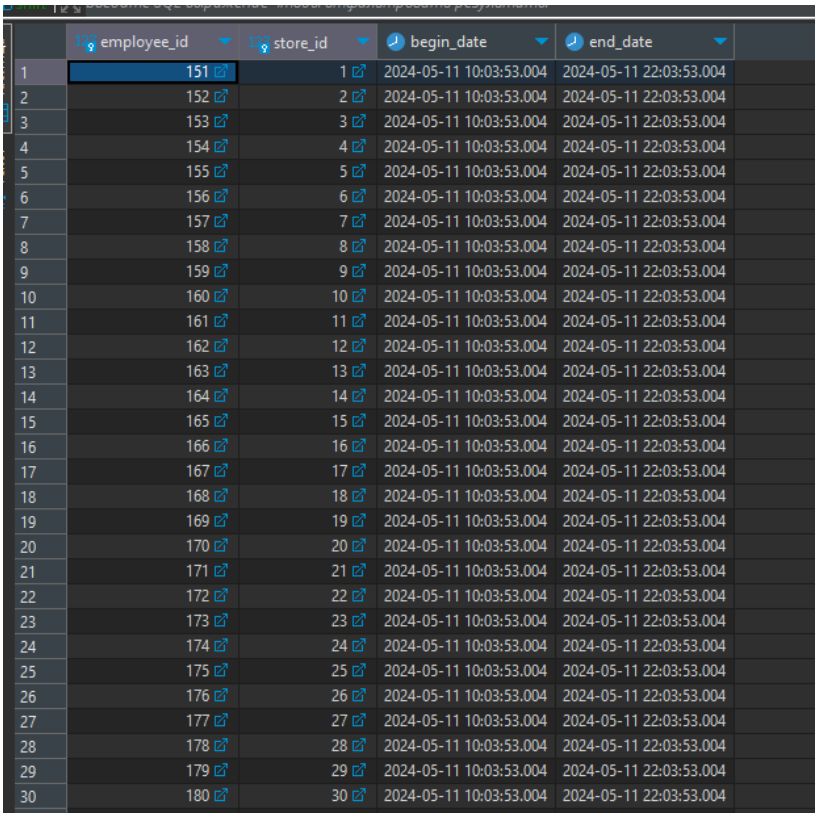
	ic	name	description	category	image_url	p
1	1	Adrenaline Rush, 0,449 л	Adrenaline Rush, 0	lemonade	https://cm.samokat.ru	159
2	2	Coca-Cola, 1 л	Coca-Cola, 1 л	lemonade	https://cm.samokat.ru	275
3	3	Dr. Pepper Original, 0,33 л	Dr. Pepper Original	lemonade	https://cm.samokat.ru	119
4	4	Вода Абрау-Дюрсо 4 воды, виноград, с газом, 0,75 л	Вода Абрау-Дюрс	lemonade	https://cm.samokat.ru	208
5	5	Из Армении: Напиток Yan Sparkling, с соком, манго, с газо	Из Армении: Напи	lemonade	https://cm.samokat.ru	116
6	6	Из Грузии: Лимонад Zandukeli, крем-сода, с газом, 0,5 л	Из Грузии: Лимон	lemonade	https://cm.samokat.ru	161
7	7	Из Грузии: Лимонад Zandukeli, саперави, с газом, 0,5 л	Из Грузии: Лимон	lemonade	https://cm.samokat.ru	159
8	8	Из Грузии: Лимонад Zandukeli, тархун, с газом, 0,5 л	Из Грузии: Лимон	lemonade	https://cm.samokat.ru	129
9	9	Кола Evervess Zero Sugar, с газом, 1 л	Кола Evervess Zerc	lemonade	https://cm.samokat.ru	119
10	10	Кола Evervess Zero Sugar, с газом, 2 л	Кола Evervess Zerc	lemonade	https://cm.samokat.ru	199
11	11	Кола Evervess, с газом, 1 л	Кола Evervess, с га	lemonade	https://cm.samokat.ru	119
12	12	Кола Evervess, с газом, 1,5 л	Кола Evervess, с га	lemonade	https://cm.samokat.ru	174
13	13	Кола Sever, с газом, 2 л	Кола Sever, с газо	lemonade	https://cm.samokat.ru	136
14	14	Кола Добрый, с газом, 1 л	Кола Добрый, с га	lemonade	https://cm.samokat.ru	136
15	15	Кола Любимая, с газом, 1 л	Кола Любимая, с	lemonade	https://cm.samokat.ru	115
16	16	Кола Любимая, с газом, 1,5 л	Кола Любимая, с	lemonade	https://cm.samokat.ru	149
17	17	Кола Самокат, без сахара, с газом, 2 л	Кола Самокат, бе	lemonade	https://cm.samokat.ru	162
18	18	Кола Самокат, с газом, 2 л	Кола Самокат, с г	lemonade	https://cm.samokat.ru	162
19	19	Лимонад Star Bar Craft, грейпфрут и малина, с газом, 0,33 л	Лимонад Star Bar (	lemonade	https://cm.samokat.ru	119
20	20	Лимонад Starbar Craft, японская груша и белая хризантема	Лимонад Starbar C	lemonade	https://cm.samokat.ru	119
21	21	Лимонад Боржоми, с соком, аджарский мандарин, 0,33 л	Лимонад Боржом	lemonade	https://cm.samokat.ru	141
22	22	Лимонад Боржоми, с соком, тархун, 0,33 л	Лимонад Боржом	lemonade	https://cm.samokat.ru	141
23	23	Лимонад Самокат, арбуз, с газом, 0,33 л	Лимонад Самокат	lemonade	https://cm.samokat.ru	188
24	24	Лимонад Самокат, тархун, с газом, 2 л	Лимонад Самокат	lemonade	https://cm.samokat.ru	162
25	25	Лимонад Старые добрые традиции, с газом, 0,5 л	Лимонад Старые ,	lemonade	https://cm.samokat.ru	61
26	26	Лёд Самокат 1 кг	Лёд Самокат 1 кг	lemonade	https://cm.samokat.ru	118
27	27	Напиток CoolCola Zero, с газом, без сахара, 1,5 л	Напиток CoolCola	lemonade	https://cm.samokat.ru	139
28	28	Напиток CoolCola, с газом, 1,5 л	Напиток CoolCola	lemonade	https://cm.samokat.ru	135
29	29	Напиток Sever, апельсин, с газом, 2 л	Напиток Sever, ап	lemonade	https://cm.samokat.ru	136
30	30	Напиток Самокат Lemon-Lime, с газом, 2 л	Напиток Самокат	lemonade	https://cm.samokat.ru	162
31	31	Напиток Самокат Orange, с газом, 2 л	Напиток Самокат	lemonade	https://cm.samokat.ru	162
32	32	Напиток Самокат, яблочный шорли, с газом, 0,5 л	Напиток Самокат,	lemonade	https://cm.samokat.ru	167
33	674	Red Bull, без сахара, 0,25 л	Red Bull, без сахар	alcohol	https://cm.samokat.ru	189

Рисунок 9 – Результат SELECT-запроса к таблице “product”



	product_id	store_id	description
1	1	1	aa2
2	2	1	aa3
3	3	1	aa4
4	4	1	aa5
5	5	1	aa6
6	6	1	aa7
7	7	1	aa8
8	8	1	aa9
9	9	1	ab1
10	10	1	ab2
11	11	1	ab3
12	12	1	ab4
13	13	1	ab5
14	14	1	ab6
15	15	1	ab7
16	16	1	ab8
17	17	1	ab9
18	18	1	ac1
19	19	1	ac2
20	20	1	ac3
21	21	1	ac4
22	22	1	ac5
23	23	1	ac6
24	24	1	ac7
25	25	1	ac8
26	26	1	ac9
27	27	1	ad1
28	28	1	ad2
29	29	1	ad3
30	30	1	ad4
31	31	1	ad5
32	32	1	ad6

Рисунок 10 – Результат SELECT-запроса к таблице “product_location”



	employee_id	store_id	begin_date	end_date
1	151	1	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
2	152	2	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
3	153	3	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
4	154	4	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
5	155	5	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
6	156	6	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
7	157	7	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
8	158	8	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
9	159	9	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
10	160	10	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
11	161	11	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
12	162	12	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
13	163	13	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
14	164	14	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
15	165	15	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
16	166	16	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
17	167	17	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
18	168	18	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
19	169	19	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
20	170	20	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
21	171	21	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
22	172	22	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
23	173	23	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
24	174	24	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
25	175	25	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
26	176	26	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
27	177	27	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
28	178	28	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
29	179	29	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004
30	180	30	2024-05-11 10:03:53.004	2024-05-11 22:03:53.004

Рисунок 11 – Результат SELECT-запроса к таблице “shift”

	id	store_id	product_id	delivery_date	expiration_date	product_amount	status
1	52 981	133	525	2024-05-11 10:03:53.200	2024-05-25 10:03:53.903	0	run_out
2	52 966	133	525	2024-05-11 10:03:53.200	2024-05-28 10:03:53.903	2	accepted
3	2 160	6	510	2024-05-11 10:03:53.200	2024-05-21 10:03:53.525	0	run_out
4	2 197	6	510	2024-05-11 10:03:53.200	2024-05-30 10:03:53.525	3	accepted
5	2 162	6	516	2024-05-11 10:03:53.200	2024-05-24 10:03:53.525	0	run_out
6	2 199	6	516	2024-05-11 10:03:53.200	2024-05-24 10:03:53.525	1	accepted
7	56 796	142	720	2024-05-11 10:03:53.200	2025-11-26 10:03:53.915	3	accepted
8	49 304	124	902	2024-05-11 10:03:53.200	2024-05-15 10:03:53.890	0	run_out
9	49 344	124	902	2024-05-11 10:03:53.200	2024-05-15 10:03:53.890	0	run_out
10	49 348	124	902	2024-05-11 10:03:53.200	2024-05-15 10:03:53.890	3	accepted
11	49 521	124	830	2024-05-11 10:03:53.200	2024-05-23 10:03:53.890	1	accepted
12	49 409	124	603	2024-05-11 10:03:53.200	2024-05-31 10:03:53.890	2	accepted
13	39 422	99	562	2024-05-11 10:03:53.200	2024-05-24 10:03:53.849	2	accepted
14	42 425	107	40	2024-05-11 10:03:53.200	2024-07-02 10:03:53.859	3	accepted
15	42 417	107	66	2024-05-11 10:03:53.200	2024-07-01 10:03:53.859	1	accepted
16	33 013	83	593	2024-05-11 10:03:53.200	2024-05-20 10:03:53.817	0	run_out
17	33 030	83	593	2024-05-11 10:03:53.200	2024-05-23 10:03:53.817	1	accepted
18	21 606	55	45	2024-05-11 10:03:53.200	2024-06-18 10:03:53.777	2	accepted
19	28 800	72	685	2024-05-11 10:03:53.200	2025-11-07 10:03:53.803	1	accepted
20	42 650	107	618	2024-05-11 10:03:53.200	2024-05-30 10:03:53.860	1	accepted
21	42 593	107	498	2024-05-11 10:03:53.200	2024-05-21 10:03:53.860	2	accepted
22	21 982	55	668	2024-05-11 10:03:53.200	2025-11-30 10:03:53.778	0	run_out
23	21 986	55	668	2024-05-11 10:03:53.200	2026-03-11 10:03:53.778	3	accepted
24	17 292	44	134	2024-05-11 10:03:53.200	2024-06-19 10:03:53.753	1	accepted
25	17 351	44	503	2024-05-11 10:03:53.200	2024-05-22 10:03:53.753	0	run_out
26	17 387	44	503	2024-05-11 10:03:53.200	2024-05-23 10:03:53.753	0	run_out
27	17 396	44	503	2024-05-11 10:03:53.200	2024-05-30 10:03:53.753	2	accepted
28	17 009	43	603	2024-05-11 10:03:53.200	2024-05-31 10:03:53.752	2	accepted
29	44 951	113	503	2024-05-11 10:03:53.200	2024-05-22 10:03:53.868	0	run_out
30	44 987	113	503	2024-05-11 10:03:53.200	2024-05-23 10:03:53.868	0	run_out
31	44 996	113	503	2024-05-11 10:03:53.200	2024-05-30 10:03:53.868	2	accepted
32	43 381	109	525	2024-05-11 10:03:53.200	2024-05-25 10:03:53.863	0	run_out
33	43 366	109	525	2024-05-11 10:03:53.200	2024-05-28 10:03:53.863	3	accepted
34	26 191	66	497	2024-05-11 10:03:53.200	2024-05-27 10:03:53.794	3	accepted
35	35 792	90	513	2024-05-11 10:03:53.200	2024-05-22 10:03:53.827	0	run_out
36	35 764	90	513	2024-05-11 10:03:53.200	2024-05-27 10:03:53.827	0	run_out
37	35 768	90	513	2024-05-11 10:03:53.200	2024-05-28 10:03:53.827	0	run_out
38	35 756	90	513	2024-05-11 10:03:53.200	2024-05-30 10:03:53.827	1	accepted
39	48 583	122	502	2024-05-11 10:03:53.200	2024-05-23 10:03:53.886	0	run_out

Рисунок 12 – Результат SELECT-запроса к таблице “shipment”

	id	address	name	owner_id
1	1	ст. Назрань, пер. Ключевой, д. 7, 863341	FVV2-FAKE	1
2	2	д. Нурлат, ш. Комарова, д. 1/5, 983665	GZA94-FAKE	2
3	3	п. Майкоп, пр. Макарова, д. 59, 095481	DZV84857-FAKE	3
4	4	ст. Кимры, бул. Интернациональный, д. 77 стр. 3, 865675	SMS2955-FAKE	4
5	5	п. Муравленко, ш. Трактовое, д. 8 стр. 39, 742333	UKGR6-FAKE	5
6	6	с. Звенигород, ул. Декабристов, д. 3/1, 859626	PZ2-FAKE	6
7	7	клх Шамары, бул. Менделеева, д. 576, 501111	YLL41852-FAKE	7
8	8	клх Рязань, пер. Кедровый, д. 951 к. 16, 888718	AJCZ24233-FAKE	8
9	9	г. Губкин, алл. Алтайская, д. 184 стр. 518, 826034	SU59838-FAKE	9
10	10	клх Волгоград, ш. Красное, д. 40 стр. 44, 983954	YP72955-FAKE	10
11	11	с. Великий Устюг, бул. Шмидта, д. 48 к. 643, 625660	HIHR25-FAKE	11
12	12	к. Верхний Уфалей, наб. Абрикосовая, д. 53 к. 416, 112849	WSK9-FAKE	12
13	13	д. Волгоград, алл. Карла Маркса, д. 3/6, 792408	YKVR33-FAKE	13
14	14	клх Красная Поляна, пер. Правды, д. 3/8 стр. 8, 468341	UPF6222-FAKE	14
15	15	г. Лотошино, пр. Комсомольский, д. 9/3 стр. 8/9, 609306	KRWH64918-FAKE	15
16	16	г. Ноябрьск, пер. К.Маркса, д. 767, 515381	NAV855-FAKE	16
17	17	к. Тольятти, пр. Советов, д. 850, 451052	FMF127-FAKE	17
18	18	г. Сузун, алл. Сплавная, д. 77, 456284	VB8336-FAKE	18
19	19	ст. Камышин, алл. Балтийская, д. 726, 023741	WWX9836-FAKE	19
20	20	п. Выборг, ш. Колхозное, д. 62, 305690	WBP5582-FAKE	20
21	21	п. Лазаревское, ш. Механическое, д. 46, 189239	IZCR42-FAKE	21
22	22	с. Анапа, алл. Социалистическая, д. 966 стр. 4, 567086	HRXR5-FAKE	22
23	23	п. Моршанск, алл. Дорожная, д. 405, 749162	GQA6652-FAKE	23
24	24	к. Новый Оскол, ш. Попова, д. 44 стр. 2/2, 910933	TUEW2296-FAKE	24
25	25	г. Губаха, пр. Калужский, д. 6, 161768	MRSI3-FAKE	25
26	26	д. Биробиджан, алл. Розы Люксембург, д. 9/9 стр. 927, 866943	KN3-FAKE	26
27	27	д. Туапсе, наб. Проточная, д. 5 к. 123, 402424	HA49-FAKE	27
28	28	к. Великий Устюг, пер. Путевой, д. 56 стр. 1, 534240	COUR27594-FAKE	28
29	29	п. Светлогорск (Калин.), бул. Блюхера, д. 3, 172722	KL83515-FAKE	29
30	30	г. Дно, пр. Высоковольтный, д. 7/2, 091660	IP1-FAKE	30
31	31	п. Ямбург, ул. Зеленая, д. 285 стр. 82, 245723	GI59211-FAKE	31
32	32	д. Пушкин, ул. Вольная, д. 736, 543617	GS4-FAKE	32
33	33	д. Нальчик, наб. Промышленная, д. 392, 859262	RFQX562-FAKE	33
34	34	ст. Агата, ш. Братское, д. 3, 771196	ILXY326-FAKE	34
35	35	с. Ржев, бул. Красных Партизан, д. 3 к. 76, 887883	HKX7-FAKE	35
36	36	п. Новый Оскол, алл. Северная, д. 9, 210552	NOD3974-FAKE	36
37	37	клх Дно, бул. Заводской, д. 6/1 к. 891, 025213	IW7-FAKE	37

Рисунок 13 – Результат SELECT-запроса к таблице “store”

2.3 Сложные запросы

Для представления данных в удобном для анализа виде написан ряд SQL-запросов. В листингах 7-13 представлены соответствующие запросы

Листинг 7 – Заказы с списком продуктов и количеством продуктов

```
WITH order_product AS (
     SELECT assembling.order_id AS order_id, product."name" AS
product_name, assembling.product_amount AS product_amount
     FROM assembling
     JOIN product ON product.id = assembling.product_id
)
SELECT "order".id, "order"."status", "order".client_id,
"order".total_price, "order".address, "order".creation_date,
"order".close_date, array_agg(order_product.product_name) as
products, array_agg(order_product.product_amount) as amounts
FROM "order"
JOIN order_product ON order_product.order_id = "order".id
GROUP BY "order".id;
```

Листинг 8 – Списки сотрудников, находящихся на смене в магазинах

```
WITH working_employee AS(
     SELECT employee.id AS employee_id, employee."name",
employee.lastname, employee."role", shift.store_id
     FROM employee, shift
     WHERE shift.employee_id = employee.id AND shift.end_date >
CURRENT_TIMESTAMP
)
SELECT store_id,
     array_agg(("name", lastname)) filter(WHERE
working_employee."role" = 'manager'),
     array_agg(("name", lastname)) filter(WHERE
working_employee."role" = 'courier'),
     array_agg(("name", lastname)) filter(WHERE
working_employee."role" = 'assembler')
FROM working_employee
group by store_id
```

Листинг 9 – Магазины с самыми маленькими остатками по продукту и средний остаток в магазине, используется представление client_assortment.

```
select *, avg(ca.amount) over (partition by ca.address) as
avg_amount
from client_assortment ca
order by amount asc;
```

Листинг 10 – Самые продаваемые продукты

```
with sales as (select p."name", count(*) as sales
from product p, assembling a
where p.id = a.product_id
group by p."name")
select *
from sales
order by sales.sales desc
```

Листинг 11 – Среднее время сборки по магазинам

```
select s."name", avg(a.close_date - a.creation_date)
from store s, assembling a, "order" o
where a.order_id = o.id and s.id = o.store_id and a.close_date
is not null
group by s."name"
```

Листинг 12 – Процент списаний партий по магазинам

```
select s2."name", (count(*) filter(where s.status = 'expired'))
/ (count(*) filter(where s.status != 'expired')) as perc
from shipment s, store s2
where s2.id = s.store_id
group by s2."name"
```

Листинг 13 – Средняя цена товаров по категориям, которые были доставлены в магазин с определенным названием (в листинге - ‘FVV2-FAKE’)

```
select p.category , avg(p.price) as avg_price
from product p
join shipment s on s.product_id = p.id
join store s2 on s2.id = s.store_id
where s2.name = 'FVV2-FAKE'
group by p.category
```

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы выполнено следующее:

- Проведен анализ предметной области, выделены основные сущности и процессы;
- Спроектирована база данных, разработаны инфологическая и даталогическая модель базы данных;
- Написан скрипт создания таблиц базы данных и триггеров, обеспечивающих согласованность данных, представлений, функций, ролей, индексов, внешних ключей;
- Разработаны и написаны сложные SQL-запросы;

В результате курсовой работы, была реализована система, позволяющая выполнять заказы продуктов.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1) Скворцова М.А. Методические указания по выполнению курсовой работы по курсу «Базы данных»
2) Иванова Г.С., Ничушкина Т.Н. Оформление текстовых документов. Методические указания по оформлению расчетно-пояснительных записок дипломных и квалификационных работ. -М.:Издательство МГТУ им. Н.Э. Баумана, 2004. -10с.
3) [Документация библиотеки psycopg] URL: https://www.psycopg.org/
4) [Документация библиотеки faker] URL: https://faker.readthedocs.io/en/master/

ПРИЛОЖЕНИЕ А.

Листинг 1 – Программа для генерации данных

```
#!/bin/pypy

from faker import Faker

try:
    import psycopg2 as ps
    from psycopg2.extras import execute_batch
except:
    import psycopg2cffi as ps
    from psycopg2cffi.extras import execute_batch
from itertools import groupby
import random
import json
from datetime import datetime, timezone, timedelta
import sys

CLIENT_AMOUNT = 150
PRODUCT_AMOUNT = 931
STORE_AMOUNT = 150
ORDER_AMOUNT = 10_000
OWNER_IDS = range(STORE_AMOUNT * 0 + 1, STORE_AMOUNT * 1 + 1)
MANAGER_IDS = range(STORE_AMOUNT * 1 + 1, STORE_AMOUNT * 2 + 1)
COURIER_IDS = range(STORE_AMOUNT * 2 + 1, STORE_AMOUNT * 3 + 1)
ASSEMBER_IDS = range(STORE_AMOUNT * 3 + 1, STORE_AMOUNT * 4 +
1)

random.seed(40)
Faker.seed(42)
fake = Faker(locale="ru_RU")
conn = ps.connect(
    dbname="postgres",
    user="postgres",
    password="12345678",
    host="localhost",
    port="5434",
)
cur = conn.cursor()


def ignore_on_fail(f):
    def g(*args):
        try:
            return f(*args)
        except Exception as e:
            print(f"continue on err: {e}", file=sys.stderr)

    return g
```

```
@ignore_on_fail
def clients():
    data = []
    for _ in range(CLIENT_AMOUNT):
        name = fake.first_name()
        lastname = fake.last_name()
        saved_addresses = [fake.address() for _ in
range(fake.random_int(0, 5))]
        phone_number = fake.phone_number()
        email = fake.email()
        data.append((name, lastname, saved_addresses,
phone_number, email))
    execute_batch(
        cur,
        """
            INSERT INTO client (\"name\", lastname,
saved_addresses, phone_number, email)
            VALUES (%s, %s, %s, %s, %s)
        """,
        data,
    )
    print(f"clients records {len(data)}")
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/clients.json", "w"
    ) as f:
        json.dump(data, f)
    return data


@ignore_on_fail
def products():
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/data/product.json", "r"
    ) as f:
        data = json.load(f)
    data = [(p["name"], p["name"], p["caterogy"], p["img"],
p["price"]) for p in data]
    print("insert into db")
    execute_batch(
        cur,
        """
            INSERT INTO product ("name", "description",
category, image_url, price)
            VALUES (%s, %s, %s::PRODUCT_CATEGORY_ENUM, %s, %s)
        """,
        data,
    )
    print(f"product records {len(data)}")
    return data
```

```
@ignore_on_fail
def stores():
    data = []
    for i in range(1, STORE_AMOUNT + 1):
        address = fake.address()
        owner_id = i
        name = fake.nic_handle()
        data.append((address, name, owner_id))
    print("insert into db")
    execute_batch(
        cur,
        """
            INSERT INTO store ("address", "name", "owner_id")
            VALUES (%s, %s, %s)
        """,
        data,
    )
    print(f"store records {len(data)}")
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/stores.json", "w"
    ) as f:
        json.dump(data, f)
    return data


@ignore_on_fail
def employees():
    data = []
    for _ in OWNER_IDS:  # owner
        name = fake.first_name()
        lastname = fake.last_name()
        role = "manager"
        data.append((name, lastname, role))
    for _ in OWNER_IDS:  # managers
        name = fake.first_name()
        lastname = fake.last_name()
        role = "manager"
        data.append((name, lastname, role))
    for _ in COURIER_IDS:  # couriers
        name = fake.first_name()
        lastname = fake.last_name()
        role = "courier"
        data.append((name, lastname, role))
    for _ in ASSEMBER_IDS:  # assemblers
        name = fake.first_name()
        lastname = fake.last_name()
        role = "assembler"
        data.append((name, lastname, role))
    print("insert into db")
    execute_batch(
```

```
        cur,
        """
            INSERT INTO employee ("name", lastname, "role")
            VALUES (%s, %s, %s::EMPLOYEE_ROLE_ENUM)
        """,
        data,
    )
    print(f"employee records {len(data)}")
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/employees.json", "w"
    ) as f:
        json.dump(data, f)
    return data


@ignore_on_fail
def product_locations():
    data = []
    descs = [
        "".join(chr(x) + chr(y) + chr(z))
        for x in range(ord("a"), ord("z") + 1)
        for y in range(ord("a"), ord("z") + 1)
        for z in range(ord("1"), ord("9") + 1)
    ]
    for store_id in range(1, STORE_AMOUNT + 1):
        for product_id in range(1, PRODUCT_AMOUNT + 1):
            desc = descs[product_id]
            data.append((product_id, store_id, desc))
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/product_locations.json",
        "w",
    ) as f:
        json.dump(data, f)
    print("insert into db")
    execute_batch(
        cur,
        """
            INSERT INTO product_location (product_id, store_id,
"description")
            VALUES (%s, %s, %s)
        """,
        data,
    )
    print(f"product_location records {len(data)}")
    return data


@ignore_on_fail
def shifts():
    store_id = lambda x: (x - 1) % STORE_AMOUNT + 1
```

```
    data = []
    start = datetime.now(timezone(timedelta(hours=3)))
    end = start + timedelta(hours=12)
    for employee_id in MANAGER_IDS:
        data.append((employee_id, store_id(employee_id), start,
end))
    for employee_id in COURIER_IDS:
        data.append((employee_id, store_id(employee_id), start,
end))
    for employee_id in ASSEMBER_IDS:
        data.append((employee_id, store_id(employee_id), start,
end))
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/shifts.json", "w"
    ) as f:
        json.dump(data, f, default=str)
    print("insert into db")
    execute_batch(
        cur,
        """
            INSERT INTO shift (employee_id, store_id,
begin_date, end_date)
            VALUES (%s, %s, %s, %s)
        """,
        data,
    )
    print(f"shift records {len(data)}")
    return data


@ignore_on_fail
def shipments():
    data = []
    exp_data = []
    exp_shipments = [
        (random.randint(1, 931), -1),  # expired
        (random.randint(1, 931), -1),  # expired
    ]
    shipments = [
        *[
            [random.randint(1, 90), random.randint(30, 60)] for
_ in range(50)
        ],  # lemonade
        *[
            [random.randint(91, 167), random.randint(20, 40)]
for _ in range(50)
        ],  # cheese
        *[[random.randint(890, 931), random.randint(1, 4)] for
_ in range(50)],  # bread
        *[
```

```
            [random.randint(493, 536), random.randint(10, 20)]
for _ in range(50)
        ],  # meat
        *[
            [random.randint(537, 663), random.randint(5, 20)]
for _ in range(50)
        ],  # yogurt
        *[
            [random.randint(810, 889), random.randint(10, 30)]
for _ in range(50)
        ],  # fruit
        *[
            [random.randint(810, 889), random.randint(10, 30)]
for _ in range(50)
        ],  # fruit
        *[
            [random.randint(664, 743), random.randint(360,
720)] for _ in range(50)
        ],  # alcohol
    ]
    for store_id in range(1, STORE_AMOUNT + 1):
        for product_id, exp in shipments:
            product_amount = random.randint(200, 1000)
            expiration_date =
datetime.now(timezone(timedelta(hours=-3))) + timedelta(
                days=exp
            )
            data.append([store_id, product_id, expiration_date,
product_amount])
        for product_id, exp in exp_shipments:
            product_amount = random.randint(10, 100)
            expiration_date =
datetime.now(timezone(timedelta(hours=-3))) + timedelta(
                days=exp
            )
            exp_data.append((store_id, product_id,
expiration_date, product_amount))
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/shipments.json", "w"
    ) as f:
        json.dump(data, f, default=str)
    print("insert into db")
    execute_batch(
        cur,
        """
            INSERT INTO shipment (store_id, product_id,
expiration_date, product_amount)
            VALUES (%s, %s, %s, %s)
        """,
        data,
    )
```

```
    execute_batch(
        cur,
        """
            INSERT INTO shipment (store_id, product_id,
expiration_date, product_amount)
            VALUES (%s, %s, %s, %s)
        """,
        exp_data,
    )
    cur.execute(
        """
            UPDATE shipment
            SET "status" = 'accepted'
            WHERE shipment.id < %s
        """,
        (str(int(len(data) * 0.95)),),
    )
    cur.execute(
        """
            SELECT check_and_mark_expired_shipments()
        """
    )
    print(f"shipment records {len(data)+len(exp_data)}")
    return list(map(lambda x: [x[0] + 1, *x[1]],
enumerate(data)))


@ignore_on_fail
def orders(shipments):
    assems = []
    orders = []

    def dec(x, i, y):
        x[i] -= y
        return y

    for order_id in range(1, ORDER_AMOUNT + 1):
        client_id = random.randint(1, CLIENT_AMOUNT)
        address = fake.address()
        shs = []
        while len(shs) < 2:
            store_id = random.randint(1, STORE_AMOUNT + 1)
            shs = list(filter(lambda x: x[1] == store_id and
x[4] > 5, shipments))
        assemblings = [
            (p, order_id, sum(x[2] for x in s))
            for p, s in groupby(
                (
                    (shipment[2], order_id, dec(shipment, 4,
random.randint(1, 3)))
                    for shipment in random.sample(
```

```
                        shs, k=random.randint(1, min(len(shs),
3))
                    )
                ),
                key=lambda x: x[0],
            )
        ]
        assems += assemblings
        orders.append((client_id, address, store_id))
    assems = list(set({(x[0], x[1]): x for x in
assems}.values()))
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/orders.json", "w"
    ) as f:
        json.dump(orders, f, default=str)
    with open(
        "/home/vzalygin/repos/bmstu-ics6/cource-
papers/db/~insert/assemblings.json", "w"
    ) as f:
        json.dump(assems, f, default=str)
    print("insert into db")
    execute_batch(
        cur,
        """
            INSERT INTO "order" (client_id, "address",
store_id)
            VALUES (%s, %s, %s)
        """,
        orders,
    )
    execute_batch(
        cur,
        """
            INSERT INTO assembling (product_id, order_id,
product_amount)
            VALUES (%s, %s, %s)
        """,
        assems,
    )
    print("insert into db")
    cur.execute(
        """
            UPDATE assembling
            SET close_date = CURRENT_TIMESTAMP
            WHERE assembling.order_id in (SELECT
assembling.order_id FROM assembling LIMIT %s)
        """,
        (str(int(len(assems) * 0.70)),),
    )
    print("insert into db")
    cur.execute(
```

```
        """
            UPDATE "order"
            SET "status" = 'cancelled'
            WHERE "order".id < %s
        """,
        (str(int(ORDER_AMOUNT * 0.05)),),
    )

    print(f"order records {ORDER_AMOUNT}")
    print(f"assembling records {len(assems)}")


@ignore_on_fail
def deliveries():
    cur.execute(
        """
            SELECT
check_ready_to_delivery_orders_and_assign_couriers();
        """
    )
    cur.execute(
        """
           SELECT count(*) FROM delivery
        """
    )
    (delivery_amount,) = cur.fetchone()
    cur.execute(
        """
            UPDATE delivery
            SET "status" = 'on_the_way'
            WHERE delivery.id < %s
        """,
        (str(int(delivery_amount * 0.70)),),
    )
    cur.execute(
        """
            UPDATE delivery
            SET "status" = 'closed'
            WHERE delivery.id < %s
        """,
        (str(int(delivery_amount * 0.50)),),
    )
    print(f"delivery records {delivery_amount}")


# static
clients()
conn.commit()
products()
conn.commit()
stores()
employees()
```

```
conn.commit()
product_locations()
conn.commit()
# dynamic
shifts()
s = shipments()
conn.commit()
orders(s)
conn.commit()
deliveries()
conn.commit()
print("db filled successfully")
```