

РЕФЕРАТ

Расчетно-пояснительная записка _ страниц, _ рисунков, _ таблиц, _ источников.

Объектом разработки является база данных для учета продаж деталей строительного и промышленного назначения.

Цель работы – создание автоматизированной системы учета со следующими функциями:

- Управление складскими запасами
- Формирование накладных
- Анализ продаж

Поставленная цель достигается за счет СУБД PostgreSQL версии 16.3, средства проектирования pgAdmin 4 версии 8.9, языка программирования Python версии 3.12.2 и среды разработки Microsoft Visual Studio Code версии 1.88.1. Разрабатываемая база данных содержит 7 таблиц, взаимодействующих друг с другом при помощи связей.

Ключевые слова – база данных, индексы, PostgreSQL, Python, SQL-запросы, таблицы, триггеры, записи, поля.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной работе создается является база данных для учета продаж деталей строительного и промышленного назначения. Она предназначена для управления складскими запасами, формирования накладных и анализа продаж. Интерфейс предоставляет пользователю возможность быстрого и удобного формирования соответствующих запросов в базу данных.

Актуальность данной работы обусловлена проблемами ручного учета на фоне роста промышленного и строительного секторов.

Интерфейс разрабатывался в соответствии со следующими требованиями:

- Простота и понятность
- Остальные

1 Анализ предметной области

Предметной областью разрабатываемого продукта является формирование накладных.

Разрабатываемая база данных "Детали для строительных и производственных компаний" предназначена для автоматизации учета и управления продажами промышленных комплектующих.

База данных выполняет следующие функции:

1. Хранение информации о поставщиках
2. Хранение информации о деталях
3. Хранение информации о сотрудниках
4. Хранение информации о покупателях

Данные требования к разрабатываемой базе данных были учтены в дальнейшем при создании модели, а также доработаны в соответствии с возникающими проблемами при проектировании.

После рассмотрения предметной области было решено создать 8 таблиц: "Деталь", "Тип детали", "Поставщик", "Покупатель", "Сотрудник", "Накладная", "Строка накладной", "Платеж".

Схема связи таблиц представлена на рисунке 1.

2. Разработка бизнес-процессов

3 Проектирование базы данных

3.1 Выделение главной сущности

В разрабатываемой базе данных можно выделить следующую иерархию:

- Центральная сущность: накладная. Выбор обусловлен предметной областью разрабатываемого продукта.
- Второстепенные сущности: деталь, тип детали, поставщик, покупатель, платеж, строка накладной, сотрудник.

3.2 Описание таблиц

Т.к. в базе данных необходимо формирование накладных, были созданы таблицы “Накладная”. Она содержит данные о дате и времени оформления накладной, общей сумме покупки, покупателе из таблицы “Покупатель”, сотруднике, который оформил накладную, из таблицы “Сотрудник” и статусе оплаты (оплачено, не оплачено, частично оплачено).

Для хранения более подробной информации о накладных была создана таблица “Строка накладной”, содержащая информацию о номере накладной, к которой принадлежит строка, из таблицы “Накладная”, детали из таблицы “Деталь”, количестве, цене за единицу, которая совпадает с соответствующей ценой детали из таблицы “Деталь”, и общей сумме покупки, которая вычисляется как произведение предыдущих двух полей.

Таблица “Деталь” содержит информацию о материале, весе, цене, типе детали из таблицы “Тип детали”, поставщике из таблицы “Поставщик”, количестве на складе, минимальном необходимом количестве на складе и наличии в продаже. Наличие в продаже определяется в зависимости от того, превышает ли текущее количество деталей на складе минимальное необходимое количество.

Для хранения более подробной информации о типах деталей была создана таблица “Тип детали”. Она содержит информацию о названии типа и описание.

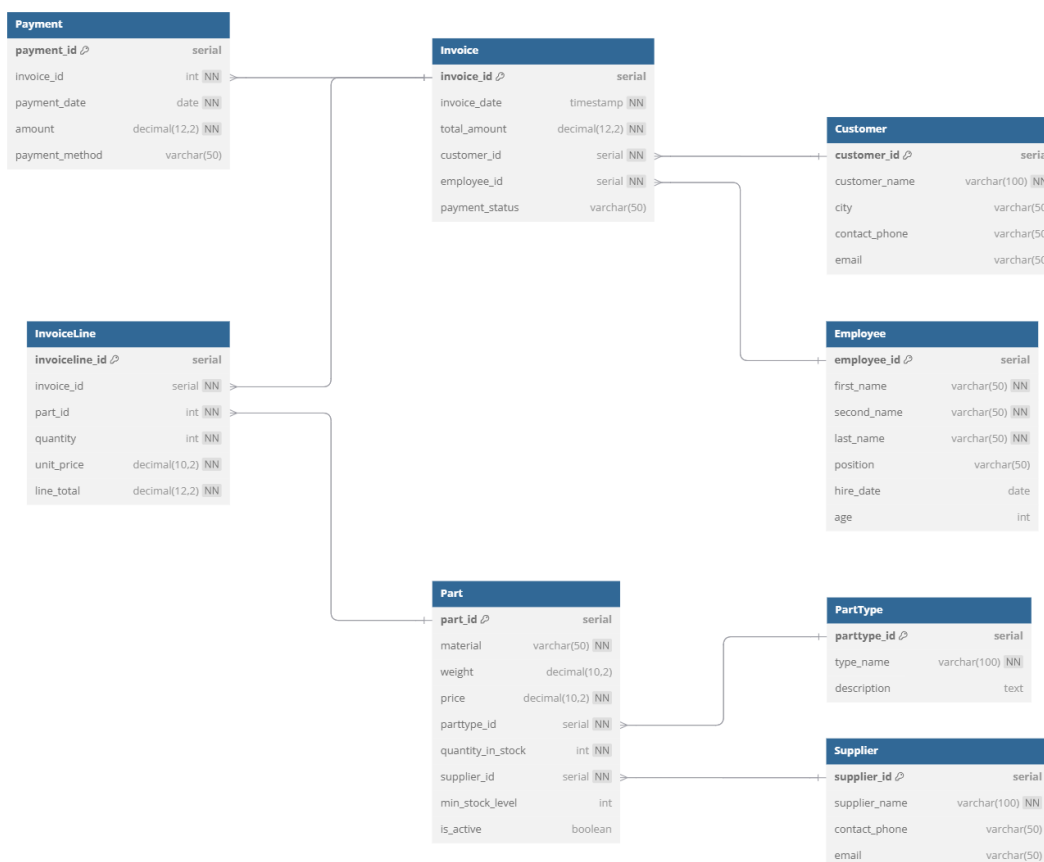
Для хранения информации о поставщиках необходимо наличие таблицы “Поставщик”. Она содержит название компании поставщика, контактный телефон и почту.

Аналогично таблице “Поставщик” была создана таблица “Покупатель”, содержащая информацию о названии компании покупателя, городе, контактном телефоне и почте.

Также была создана таблица “Сотрудник” с информацией об имени, фамилии и отчестве сотрудника, занимаемой должности, дате найма и возрасте.

3.3 Схема базы данных

Модель разработанной базы данных “Продажа деталей” представлена на рисунке 2.



На данном рисунке присутствует множество стрелок, соединяющих таблицы.

Стрелка, соединяющая таблицы “Строка накладной” и “Накладная”, является идентификацией первичного ключа “invoice_id” таблицы “Накладная” и внешнего ключа “invoice_id” таблицы “Строка накладной”.

Стрелка, соединяющая таблицы “Строка накладной” и “Деталь”, является идентификацией первичного ключа “part_id” таблицы “Деталь” и внешнего ключа “part_id” таблицы “Строка накладной”.

Стрелка, соединяющая таблицы “Платеж” и “Накладная”, является идентификацией первичного ключа “invoice_id” таблицы “Накладная” и внешнего ключа “invoice_id” таблицы “Платеж”.

Стрелка, соединяющая таблицы “Накладная” и “Покупатель”, является идентификацией первичного ключа “customer_id” таблицы “Покупатель” и внешнего ключа “customer_id” таблицы “Накладная”.

Стрелка, соединяющая таблицы “Накладная” и “Сотрудник”, является идентификацией первичного ключа “employee_id” таблицы “Сотрудник” и внешнего ключа “employee_id” таблицы “Накладная”.

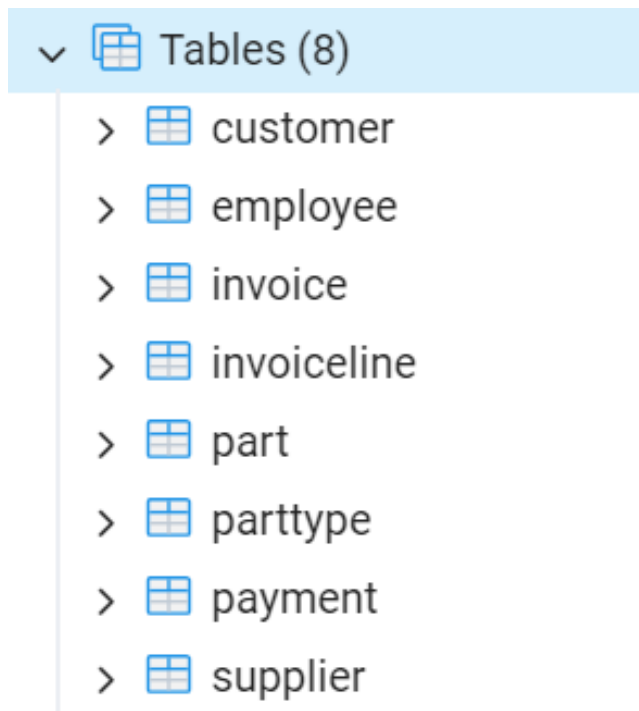
Стрелка, соединяющая таблицы “Деталь” и “Тип детали”, является идентификацией первичного ключа “part type_id” таблицы “Деталь” и внешнего ключа “paratype_id” таблицы “Тип детали”.

Стрелка, соединяющая таблицы “Деталь” и “Поставщик”, является идентификацией первичного ключа “supplier_id” таблицы “Деталь” и внешнего ключа “supplier_id” таблицы “Поставщик”.

4 Заполнение таблиц

4.1 Создание базы данных в pgAdmin4

Таблицы были созданы при помощи sql-запроса, показанном в приложении 1. В результате выполнения запроса были созданы 8 таблиц, показанных на рисунке 3:



4.2 Заполнение таблиц

Созданные ранее таблицы были заполнены при помощи функции “copy” после выполнения sql-запроса по созданию таблиц (см. приложение 1). Файлы .csv были созданы при помощи скрипта на Python, с которым можно ознакомиться в приложении 2.

4.3 Создание триггеров

Для поддержания целостности данных, оптимизации запросов и предотвращения ошибок, были созданы триггеры для:

1. Проверки на занятость номера при добавлении и изменении записи в таблице “Поставщик”
2. Проверки на занятость названия и города при добавлении и изменении записи в таблице “Покупатель”
3. Проверки на занятость названия типа при добавлении и изменении записи в таблице “Тип детали”
4. Проверки на занятость материала и типа детали при добавлении и изменении записи в таблице “Деталь”
5. Проверки совершеннолетия сотрудника при добавлении записи в таблицу “Сотрудник”
6. Проверки наличия необходимого количества деталей на складе перед добавлением записи в таблицу “Строка накладной”
7. Автоматического уменьшения количества деталей на складе при добавлении записи в таблицу “Строка накладной”
8. Предотвращения ошибки, связанной с добавлением даты платежа до даты оформления соответствующей накладной, при добавлении записи в таблицу “Платеж”
9. Предотвращения ошибки, связанной с добавлением суммы платежа, превышающей оставшуюся сумму долга при добавлении записи в таблицу “Платеж”
10. Предотвращения ошибки, связанной с попыткой оплатить уже оплаченную накладную, при добавлении записи в таблицу “Платеж”
11. Поиска минимального незанятого id при добавлении записи в таблицу “Покупатель”
12. Поиска минимального незанятого id при добавлении записи в таблицу “Поставщик”
13. Поиска минимального незанятого id при добавлении записи в таблицу “Платеж”

14. Поиска минимального незанятого id при добавлении записи в таблицу “Сотрудник”
15. Поиска минимального незанятого id при добавлении записи в таблицу “Деталь”
16. Поиска минимального незанятого id при добавлении записи в таблицу “Тип детали”
17. Поиска минимального незанятого id при добавлении записи в таблицу “Накладная”
18. Поиска минимального незанятого id при добавлении записи в таблицу “Строка накладной”
19. Автоматического обновления статуса наличия в продаже при изменении количества деталей на складе в таблице “Деталь”
20. Автоматического обновления статуса оплаты при добавлении записи в таблицу “Платеж”
21. Автоматического обновления общей суммы покупки в накладной при добавлении, удалении и изменении записи в таблице “Строка накладной”

Подробно ознакомиться с каждым из них можно в приложениях 3 – 23.

5 Разработка запросов

ПРИЛОЖЕНИЕ 1

```
CREATE TABLE PartType (
```

```
    parttype_id SERIAL PRIMARY KEY,
```

```
    type_name VARCHAR(100) NOT NULL,
```

```
    description TEXT
```

```
);
```

```
copy parttype FROM 'C:/tables/part_types.csv' WITH (FORMAT csv, HEADER  
true, DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE Supplier (
```

```
    supplier_id SERIAL PRIMARY KEY,
```

```
    supplier_name VARCHAR(100) NOT NULL,
```

```
    contact_phone VARCHAR(50),
```

```
    email VARCHAR(50)
```

```
);
```

```
copy supplier FROM 'C:/tables/suppliers.csv' WITH (FORMAT csv, HEADER  
true, DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE Part (
```

```
    part_id SERIAL PRIMARY KEY,
```

```
material VARCHAR(50) NOT NULL,  
  
weight DECIMAL(10,2),  
  
price DECIMAL(10,2) NOT NULL,  
  
parttype_id SERIAL REFERENCES PartType(parttype_id),  
  
quantity_in_stock INT NOT NULL,  
  
supplier_id SERIAL REFERENCES Supplier(supplier_id),  
  
min_stock_level INT DEFAULT 0,  
  
is_active BOOLEAN DEFAULT TRUE  
  
);  
  
copy part FROM 'C:/tables/parts.csv' WITH (FORMAT csv, HEADER true,  
DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE Customer (  
  
customer_id SERIAL PRIMARY KEY,  
  
customer_name VARCHAR(100) NOT NULL,  
  
city VARCHAR(50),  
  
contact_phone VARCHAR(50),  
  
email VARCHAR(50)  
  
);
```

```
copy customer FROM 'C:/tables/customers.csv' WITH (FORMAT csv, HEADER
true, DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE Employee (
    employee_id SERIAL PRIMARY KEY,
    first_name VARCHAR(50) NOT NULL,
    second_name VARCHAR(50) NOT NULL,
    last_name VARCHAR(50) NOT NULL,
    position VARCHAR(50),
    hire_date DATE,
    age INT CHECK (age BETWEEN 18 AND 65)
);
```

```
copy employee FROM 'C:/tables/employees.csv' WITH (FORMAT csv,
HEADER true, DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE Invoice (
    invoice_id SERIAL PRIMARY KEY,
    invoice_date TIMESTAMP NOT NULL,
    total_amount DECIMAL(12,2) NOT NULL,
    customer_id SERIAL REFERENCES Customer(customer_id),
```

```
employee_id SERIAL REFERENCES Employee(employee_id),  
  
payment_status VARCHAR(50) CHECK (payment_status IN ('Оплачено', 'Не  
оплачено', 'Частично оплачено')) DEFAULT 'Не оплачено'  
  
);  
  
copy invoice FROM 'C:/tables/invoices.csv' WITH (FORMAT csv, HEADER  
true, DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE InvoiceLine (  
  
    invoiceline_id SERIAL PRIMARY KEY,  
  
    invoice_id SERIAL REFERENCES Invoice(invoice_id),  
  
    part_id INT REFERENCES Part(part_id),  
  
    quantity INT NOT NULL CHECK (quantity > 0),  
  
    unit_price DECIMAL(10,2) NOT NULL,  
  
    line_total DECIMAL(12,2) NOT NULL  
  
);  
  
copy invoiceline FROM 'C:/tables/invoice_lines.csv' WITH (FORMAT csv,  
HEADER true, DELIMITER ',', ENCODING 'UTF8');
```

```
CREATE TABLE Payment (  
  
    payment_id SERIAL PRIMARY KEY,  
  
    invoice_id INT REFERENCES Invoice(invoice_id),
```

```
payment_date DATE NOT NULL,  
  
amount DECIMAL(12,2) NOT NULL CHECK (amount > 0),  
  
payment_method VARCHAR(50) CHECK (payment_method IN ('Наличный  
расчет', 'Безналичный расчет')) DEFAULT 'Безналичный расчет'  
  
);  
  
COPY payment FROM 'C:/tables/payments.csv' WITH (FORMAT csv,  
HEADER true, DELIMITER ',', ENCODING 'UTF8');
```

-- Индексы для ускорения

```
CREATE INDEX idx_part_parttype ON Part(parttype_id);  
  
CREATE INDEX idx_part_supplier ON Part(supplier_id);  
  
CREATE INDEX idx_invoice_customer ON Invoice(customer_id);  
  
CREATE INDEX idx_invoice_employee ON Invoice(employee_id);  
  
CREATE INDEX idx_invoice_date ON Invoice(invoice_date);  
  
CREATE INDEX idx_invoiceline_part ON InvoiceLine(part_id);  
  
CREATE INDEX idx_invoiceline_invoice ON InvoiceLine(invoice_id);  
  
CREATE INDEX idx_payment_invoice ON Payment(invoice_id);  
  
CREATE INDEX idx_payment_date ON Payment(payment_date);
```


ПРИЛОЖЕНИЕ 2

```
import csv
```

```
import random
```

```
from datetime import datetime, timedelta
```

```
from faker import Faker
```

```
from natasha import MorphVocab
```

```
m = MorphVocab()
```

```
fake = Faker("ru_RU")
```

```
# Настройки
```

```
NUM_PARTS = 1_000_000 #
```

```
NUM_PART_TYPES = 100 #
```

```
NUM_SUPPLIERS = 114 #
```

```
NUM_CUSTOMERS = 222 #
```

```
NUM_EMPLOYEES = 200 #
```

```
NUM_INVOICES = 300_000
```

```
MAX_INVOICE_LINES = 1_000_000
```

```
# Генерация данных для таблицы Customer (Покупатели)
```

```
def generate_customers():
```

```
    customers = []
```

```
    companies = ['Doosan Heavy Industries', 'Rolls-Royce Holdings', 'General Dynamics',  
'Lockheed Martin', 'Boeing', 'Northrop Grumman', 'Raytheon Technologies', 'L3Harris  
Technologies', 'BAE Systems', 'Safran', 'United Technologies', 'PACCAR', 'Continental  
AG', 'Magna International', 'Dana Incorporated', 'Aisin Seiki', 'NSK Ltd.', 'Schaeffler  
Group', 'GKN Automotive', 'Mahle GmbH', 'Tenneco', 'Yazaki Corporation', 'Sumitomo  
Electric', 'Furukawa Electric', 'Nexans', 'Prysmian Group', 'Legrand', 'Hubbell  
Incorporated', 'nVent Electric', 'Amphenol', 'TE Connectivity', 'Molex', 'Aptiv', 'Lear  
Corporation', 'Adient', 'BASF', 'Dow Chemical', 'DuPont', 'LyondellBasell', 'Linde plc',  
'Air Liquide', 'Praxair', 'Mitsubishi Chemical', 'Sumitomo Chemical', 'Toray Industries',  
'LG Chem', 'Samsung SDI', 'SK Innovation', 'ThyssenKrupp Steel', 'Alcoa', 'Rio Tinto',  
'BHP', 'Glencore', 'Freeport-McMoRan', 'Vale S.A.', 'Anglo American', 'Barrick Gold',  
'Newmont Corporation', 'Fluor Corporation', 'Bechtel', 'Jacobs Engineering', 'AECOM',  
'KBR', 'McDermott International', 'TechnipFMC', 'Saipem', 'Subsea 7', 'Weatherford  
International', 'NOV (National Oilwell Varco)', 'Mapal', 'Guhring', 'Mitsubishi  
Materials', 'Sumitomo Electric Hardmetal', 'Kyocera', 'Ceratizit', 'Plansee Group',  
'Hitachi Metals', 'Daido Steel', 'Nippon Steel & Sumitomo Metal', 'JFE Steel', 'Kobe  
Steel', 'Aichi Steel', 'Sanyo Special Steel', 'Ovako', 'Uddeholm', 'Böhler', 'Carpenter  
Technology', 'Allegheny Technologies', 'Haynes International', 'Special Metals  
Corporation', 'VDM Metals', 'Outokumpu Stainless', 'Steel Dynamics', 'AK Steel', 'U.S.  
Steel', 'Cleveland-Cliffs', 'CSN (Companhia Siderúrgica Nacional)', 'Usiminas',  
'Ternium', 'Techint Group', 'Tenova', 'Danieli', 'SMS Group', 'Primetals Technologies',  
'Andritz', 'Loesche', 'Gebr. Pfeiffer', 'Polysius', 'Claudius Peters', 'Schenck Process',  
'Bühler Group', 'GEA Group', 'ITT Inc.', 'Kaeser Kompressoren', 'Hitachi Construction  
Machinery', 'Kobelco', 'JCB', 'CASE Construction', 'New Holland', 'Terex', 'Manitowoc',  
'Tadano', 'Sennebogen', 'Palfinger', 'Haulotte', 'JLG Industries', 'Genie (Terex)', 'Skyjack',
```

'Manitou Group', 'Doosan Infracore', 'Hyundai Construction Equipment', 'Sany Heavy Industry', 'Zoomlion', 'XCMG', 'Liugong', 'Lonking', 'Shantui', 'SDLG', 'Yutong Heavy Industries', 'Sinotruk', 'FAW', 'Dongfeng Motor', 'Shaanxi Heavy Duty Automobile', 'Beiben Trucks', 'Iveco', 'Volvo Trucks', 'DAF Trucks', 'Renault Trucks', 'Mercedes-Benz Trucks', 'Freightliner', 'Western Star', 'Kenworth', 'Peterbilt', 'Mack Trucks', 'Navistar International', 'Hino Motors', 'Isuzu Motors', 'UD Trucks', 'Fuso', 'Tata Motors', 'Ashok Leyland', 'Eicher Motors', 'Mahindra Truck and Bus', 'Kamaz', 'GAZ Group', 'KAMAZ', 'MAZ', 'KrAZ', 'BelAZ', 'Scania-Vabis', 'Van Hool', 'VDL Groep', 'Daimler Buses', 'Volvo Buses', 'Irizar', 'Solaris Bus & Coach', 'MAN Bus', 'Neoplan', 'Setra', 'Marcopolo', 'Caio Induscar', 'Comil', 'Agrale', 'Tatra', 'Avtotor', 'Sollers', 'UAZ', 'ZIL', 'KAMAZ', 'BAZ', 'ZiL', 'AMO ZIL', 'UralAZ', 'KAvZ', 'PAZ', 'LiAZ', 'NefAZ', 'MAZ', 'BelAZ', 'MoAZ', 'KrAZ', 'LAZ', 'Bogdan', 'Etalon', 'Volkswagen Commercial Vehicles', 'Ford Trucks', 'Iveco Defence Vehicles', 'Oshkosh Corporation', 'Navistar Defense', 'Rheinmetall MAN Military Vehicles', 'BAE Systems Land & Armaments', 'General Dynamics Land Systems', 'Textron Marine & Land Systems', 'Patria', 'Nexter Systems', 'Krauss-Maffei Wegmann', 'Hyundai Rotem', 'Doosan DST', 'Hanwha Defense', 'Norinco', 'Sinomach', 'China North Industries Group']

cities = ['Чханвон', 'Лондон', 'Рестон', 'Бетесда', 'Чикаго', 'Фолс-Черч', 'Уолтем', 'Мельбурн', 'Лондон', 'Париж', 'Фармингтон', 'Белвью', 'Ганновер', 'Орора', 'Моми', 'Карья', 'Токио', 'Херцогенаурах', 'Реддич', 'Штутгарт', 'Лейк-Форест', 'Токио', 'Осака', 'Токио', 'Париж', 'Милан', 'Лимож', 'Шелтон', 'Лондон', 'Уоллингфорд', 'Шаффхаузен', 'Лайл', 'Дублин', 'Саутфилд', 'Плимут', 'Людвигсхафен', 'Мидленд', 'Уилмингтон', 'Хьюстон', 'Гилфорд', 'Париж', 'Данбери', 'Токио', 'Токио', 'Токио', 'Сеул', 'Сеул', 'Сеул', 'Дуйсбург', 'Питтсбург', 'Мельбурн', 'Мельбурн', 'Баар', 'Финикс', 'Рио-де-Жанейро', 'Лондон', 'Торонто', 'Денвер', 'Ирвинг', 'Рестон', 'Даллас', 'Лос-Анджелес', 'Хьюстон', 'Хьюстон', 'Лондон', 'Сан-Донато-Миланезе', 'Лондон', 'Хьюстон', 'Хьюстон', 'Аален', 'Альбштадт', 'Токио', 'Осака', 'Киото', 'Маммер', 'Ройтте', 'Токио', 'Нагоя', 'Токио', 'Токио', 'Кобе', 'Токай', 'Химэдзи',

'Стокгольм', 'Хагфорс', 'Капфенберг', 'Рединг', 'Питтсбург', 'Кокомо', 'Хантингтон',
'Вердоль', 'Эспоо', 'Форт-Уэйн', 'Уэст-Честер', 'Питтсбург', 'Кливленд', 'Сан-Паулу',
'Белу-Оризонти', 'Сан-Николас-де-лос-Гарса', 'Милан', 'Милан', 'Буттрио',
'Дюссельдорф', 'Лондон', 'Грац', 'Дюссельдорф', 'Кайзерслаутерн', 'Эссен',
'Гамбург', 'Дармштадт', 'Уцвиль', 'Дюссельдорф', 'Уайт-Плейнс', 'Кобург', 'Токио',
'Токио', 'Рочестер', 'Расин', 'Турин', 'Норуолк', 'Манитовок', 'Такамацу',
'Штраубинг', 'Зальцбург', 'Лорм', 'Хейгерстаун', 'Редмонд', 'Гвелф', 'Ансени', 'Сеул',
'Сеул', 'Чанша', 'Чанша', 'Сюйчжоу', 'Лючжоу', 'Шанхай', 'Цзинин', 'Линьи',
'Чжэнчжоу', 'Цзинань', 'Чанчунь', 'Ухань', 'Сиань', 'Чунцин', 'Турин', 'Гетеборг',
'Эйндховен', 'Сен-Приест', 'Штутгарт', 'Портленд', 'Портленд', 'Киркланд', 'Дентон',
'Гринсборо', 'Лайл', 'Токио', 'Токио', 'Агео', 'Кавасаки', 'Мумбаи', 'Ченнаи',
'Тургаон', 'Мумбаи', 'Набережные Челны', 'Нижний Новгород', 'Набережные
Челны', 'Минск', 'Кременчуг', 'Жодио', 'Седертелье', 'Конингсхойкт', 'Эйндховен',
'Штутгарт', 'Гетеборг', 'Ормайстеги', 'Болехово', 'Мюнхен', 'Штутгарт', 'Ульм',
'Кашиас-ду-Сул', 'Сан-Паулу', 'Эрешин', 'Кашиас-ду-Сул', 'Копрживнице',
'Калининград', 'Москва', 'Ульяновск', 'Москва', 'Набережные Челны', 'Брянск',
'Москва', 'Москва', 'Миасс', 'Курган', 'Павлово', 'Ликино-Дулево', 'Нефтекамск',
'Минск', 'Жодио', 'Могилев', 'Кременчуг', 'Львов', 'Черкаск', 'Москва', 'Ганновер',
'Стамбул', 'Больцано', 'Ошкош', 'Уоррен', 'Мюнхен', 'Стерлинг-Хайтс', 'Лондон',
'Новый Орлеан', 'Хельсинки', 'Версаль', 'Мюнхен', 'Чханвон', 'Сеул', 'Сеул', 'Пекин',
'Пекин', 'Пекин']

```
mailboxes = ['gmail.com', 'yahoo.com', 'outlook.com', 'hotmail.com', 'aol.com',  
'mail.com', 'protonmail.com', 'zoho.com', 'yandex.ru', 'mail.ru', 'icloud.com', 'gmx.com',  
'tutanota.com', 'fastmail.com', 'hushmail.com', 'qq.com', '163.com', 'rediffmail.com',  
'sina.com', 'naver.com']
```

```
for i in range(0, NUM_CUSTOMERS):
```

```
    customers.append({
```

```
'customer_id': i + 1,  
  
'customer_name': companies[i],  
  
'city': cities[i],  
  
'contact_phone': fake.phone_number(),  
  
'email': f'{companies[i].replace(' ', ').lower()}@{random.choice(mailboxes)}'  
  
}))  
  
return customers
```

Генерация данных для таблицы PartType (Типы деталей)

```
def generate_part_types():
```

```
    part_types = []
```

```
    types = [
```

```
        'Вал', 'Шестерня', 'Подшипник', 'Муфта',
```

```
        'Крышка', 'Болт', 'Гайка', 'Шайба',
```

```
        'Цепь', 'Ролик', 'Клапан', 'Фильтр',
```

```
        'Трубка', 'Штуцер', 'Пластина',
```

```
        'Диск', 'Лента', 'Рычаг', 'Решетка'
```

```
    ]
```

```
    adjectives = [
```

```
        'упорн', 'стопорн', 'переходн', 'соединительн',
```

'защитн', 'крепежн', 'пружинн',
'вращательн', 'клинов', 'уплотнительн',
'армированн', 'обратн',
'регулировочн', 'фрикционн'
]

part_functions = [
 'передачи вращательного момента', 'фиксации компонентов', 'уплотнения
соединений',
 'снижения трения', 'передачи мощности', 'соединения валов', 'защиты
механизмов',
 'крепления деталей', 'создания упора', 'регулировки зазоров', 'отвода жидкостей',
 'очистки рабочих сред', 'демпфирования вибраций', 'изменения передаточного
отношения',
 'торможения системы', 'переключения скоростей', 'охлаждения узлов',
'фильтрации масел',
 'компенсации температурных расширений', 'перераспределения нагрузок'
]

applications = [
 'в редукторах', 'в двигателях внутреннего сгорания', 'в коробках передач',
 'в насосном оборудовании', 'в компрессорах', 'в турбинных установках',
 'в станках ЧПУ', 'в гидравлических системах', 'в пневматических линиях',

'в автомобильных трансмиссиях', 'в авиационных агрегатах', 'в
железнодорожной технике',

 'в горнодобывающем оборудовании', 'в судовых механизмах', 'в
энергетических установках'

]

while len(part_types) < NUM_PART_TYPES:

 type = random.choice(types)

 if m.parse(type)[0].feats['Gender'] == 'Masc':

 ending = 'ый'

 elif m.parse(type)[0].feats['Gender'] == 'Fem':

 ending = 'ая'

 type_name = f'{random.choice(types)} {random.choice(adjectives) + ending}'

 if type_name not in part_types:

 part_types.append({

 'parttype_id': len(part_types) + 1,

 'type_name': type_name,

 'description': f'Используется для {random.choice(part_functions)}
{random.choice(applications)}'

 }))

return part_types

Генерация данных для таблицы Supplier (Поставщики)

```
def generate_suppliers():
```

```
    suppliers = []
```

```
        suppliers_names = ['Bosch', 'Siemens', 'General Electric', 'Caterpillar', 'Honeywell',  
'3M', 'ABB', 'Schneider Electric', 'Mitsubishi Heavy Industries', 'Hitachi', 'Toshiba',  
'Hyundai Heavy Industries', 'Doosan', 'ThyssenKrupp', 'Rolls-Royce', 'Alstom', 'Emerson  
Electric', 'Rockwell Automation', 'Cummins', 'John Deere', 'Volvo Group', 'Daimler  
Truck', 'Parker Hannifin', 'Eaton', 'ZF Friedrichshafen', 'Continental', 'Magna',  
'BorgWarner', 'Valeo', 'Faurecia', 'Mahle', 'Knorr-Bremse', 'Wabtec', 'SKF', 'Timken',  
'NSK', 'NTN', 'JTEKT', 'Schaeffler', 'GKN', 'Sandvik', 'Kennametal', 'Walter AG', 'Iscar',  
'Seco Tools', 'MAPAL', 'Dormer Pramet', 'Gühring', 'Liebherr', 'Komatsu', 'Wärtsilä',  
'MAN Energy Solutions', 'Sulzer', 'Atlas Copco', 'Ingersoll Rand', 'Gardner Denver',  
'Sullair', 'Kaeser', 'Grundfos', 'KSB', 'Wilo', 'Xylem', 'Flowserve', 'ITT Inc', 'Pentair',  
'Alfa Laval', 'GEA', 'SPX Flow', 'SMC Corporation', 'Festo', 'Pall', 'Donaldson', 'Parker  
Hannifin', 'Swagelok', 'Camoszi', 'Norgren', 'Rotork', 'AUMA', 'Siemens Healthineers',  
'Philips', 'GE Healthcare', 'Baker Hughes', 'Halliburton', 'Schlumberger', 'Weatherford',  
'National Oilwell Varco', 'Tenaris', 'Vallourec', 'TMK', 'SSAB', 'Voestalpine',  
'ArcelorMittal', 'Nippon Steel', 'POSCO', 'Tata Steel', 'Outokumpu', 'Aperam', 'Nucor',  
'Gerdau', 'Severstal', 'Metso Outotec', 'FLSmidth', 'Weir Group', 'KHD Humboldt  
Wedag', 'Claas', 'CNH Industrial', 'AGCO', 'Kubota', 'Yanmar', 'Deutz', 'Perkins', 'MTU',  
'Scania', 'MAN Truck & Bus']
```

```
        mailboxes = ['gmail.com', 'yahoo.com', 'outlook.com', 'hotmail.com', 'aol.com',  
'mail.com', 'protonmail.com', 'zoho.com', 'yandex.ru', 'mail.ru', 'icloud.com', 'gmx.com',  
'tutanota.com', 'fastmail.com', 'hushmail.com', 'qq.com', '163.com', 'rediffmail.com',  
'sina.com', 'naver.com']
```

```

for i in range(0, NUM_SUPPLIERS):

    suppliers.append({

        'supplier_id': i + 1,

        'supplier_name': suppliers_names[i],

        'contact_phone': fake.phone_number(),

                                'email':      f'{suppliers_names[i].replace(' ',
''.lower())}@{random.choice(mailboxes)}'

        })

    return suppliers

```

Генерация данных для таблицы Part (Детали)

```

def generate_parts():

    materials = ['Сталь', 'Чугун', 'Алюминий', 'Медь', 'Латунь', 'Бронза', 'Титан',
'Никель', 'Железо', 'Резина', 'Углепластик', 'Керамика', 'Графит']

    parts = []

```

```

while len(parts) < NUM_PARTS:

    part = {

        'part_id': len(parts) + 1,

        'material': random.choice(materials),

```

```
'weight_kg': round(random.uniform(0.01, 50.0), 3),  
  
'price_usd': round(random.uniform(0.1, 500.0), 2),  
  
'parttype_id': random.randint(1, NUM_PART_TYPES),  
  
'quantity_in_stock': random.randint(0, 1000),  
  
'supplier_id': random.randint(1, NUM_SUPPLIERS),  
  
'min_stock_level': random.randint(5, 50)  
  
}
```

```
if part['quantity_in_stock'] < part['min_stock_level']:
```

```
    part['is_active'] = False
```

```
else:
```

```
    part['is_active'] = True
```

```
parts.append(part)
```

```
return parts
```

Генерация данных для таблицы Employee (Сотрудники)

```
def generate_employees():
```

```
positions = ['Менеджер по продажам', 'Специалист по закупкам', 'Логист',  
'Технический консультант', 'Маркетолог', 'Складской работник', 'Сервисный  
инженер']
```

```
employees = []
```

```
for i in range(0, NUM_EMPLOYEES):
```

```
    first_names = ['Александр', 'Дмитрий', 'Максим', 'Сергей', 'Андрей', 'Алексей',  
'Артем', 'Илья', 'Петр', 'Михаил', 'Геннадий', 'Матвей', 'Роман', 'Егор', 'Арсений',  
'Иван', 'Денис', 'Евгений', 'Даниил', 'Павел']
```

```
    last_names = ['Александрович', 'Дмитриевич', 'Максимович', 'Сергеевич',  
'Андреевич', 'Алексеевич', 'Артемович', 'Ильич', 'Петрович', 'Михайлович',  
'Геннадьевич', 'Матвеевич', 'Романович', 'Егорович', 'Арсениевич', 'Иванович',  
'Денисович', 'Евгениевич', 'Даниилович', 'Павелович']
```

```
    age = random.randint(18, 65)
```

```
    employees.append({
```

```
        'employee_id': i + 1,
```

```
        'first_name': random.choice(first_names),
```

```
        'second_name': fake.last_name_male(),
```

```
        'last_name': random.choice(last_names),
```

```
        'position': random.choice(positions),
```

```
        'hire_date': fake.date_between(start_date=f'-{(age-18)*365+10}d', end_date='-  
1d').isoformat(),
```

```
        'age': age
    })

    return employees
```

Генерация данных для таблицы Invoice (Накладные)

```
def generate_invoices():

    invoices = []

    for i in range(1, NUM_INVOICES + 1):

        invoice_date = fake.date_time_between(start_date=datetime.now()-
timedelta(days=365*5), end_date=datetime.now())

        invoices.append({

            'invoice_id': i,

            'invoice_date': invoice_date.isoformat(),

            'total_amount': 0,

            'customer_id': random.randint(1, NUM_CUSTOMERS),

            'employee_id': random.randint(1, NUM_EMPLOYEES),

            'payment_status': 'Не оплачено'

        })

    return invoices
```

Генерация данных для таблицы InvoiceLine (Строки накладных)

```
def generate_invoice_lines(invoices, parts):
```

```
    invoice_lines = []
```

```
    line_id = 1
```

```
    active_parts = [p for p in parts if p['is_active']]
```

```
    for invoice in invoices:
```

```
        num_lines = random.randint(1, 8)
```

```
        invoice_total = 0
```

```
        for _ in range(num_lines):
```

```
            part = random.choice(active_parts)
```

```
            quantity = random.randint(1, 100)
```

```
            unit_price = part['price_usd']
```

```
            line_total = round(quantity * unit_price, 2)
```

```
            invoice_lines.append({
```

```
                'invoiceline_id': line_id,
```

```
                'invoice_id': invoice['invoice_id'],
```

```
                'part_id': part['part_id'],
```

```
        'quantity': quantity,  
        'unit_price': unit_price,  
        'line_total': line_total  
    })
```

```
    invoice_total += line_total
```

```
    line_id += 1
```

```
    invoice['total_amount'] = round(invoice_total, 2)
```

```
    return invoice_lines
```

Генерация данных для таблицы Payments (Платежи)

```
def generate_payments(invoices):
```

```
    payments = []
```

```
    line_id = 1
```

```
    for invoice in invoices:
```

```
        status_chance = random.random()
```

```
        total = invoice['total_amount']
```

```
paid = 0
```

```
if status_chance < 0.7:
```

```
    num_parts = random.randint(1, 5)
```

```
    part_amounts = [round(total / num_parts, 2) for _ in range(num_parts - 1)]
```

```
    part_amounts.append(round(total - sum(part_amounts), 2))
```

```
    for amount in part_amounts:
```

```
        payment_date = fake.date_time_between(
```

```
            start_date=datetime.fromisoformat(invoice['invoice_date']),
```

```
            end_date=datetime.now()
```

```
        )
```

```
        payments.append({
```

```
            'payment_id': line_id,
```

```
            'invoice_id': invoice['invoice_id'],
```

```
            'payment_date': payment_date.isoformat(),
```

```
            'amount': amount,
```

```
            'payment_method': random.choices(
```

```
                ['Наличный расчет', 'Безналичный расчет'], weights=[10, 90]
```

```
            )[0]
```



```

    })

    line_id += 1

    paid += amount

    invoice['payment_status'] = 'Оплачено'

elif status_chance < 0.9:

    amount = round(random.uniform(0.1, total * 0.9), 2)

    payment_date = fake.date_time_between(

        start_date=datetime.fromisoformat(invoice['invoice_date']),

        end_date=datetime.now()

    )

    payments.append({

        'payment_id': line_id,

        'invoice_id': invoice['invoice_id'],

        'payment_date': payment_date.isoformat(),

        'amount': amount,

        'payment_method': random.choice(['Наличный расчет', 'Безналичный
расчет'])

    })

```

```
    line_id += 1

    invoice['payment_status'] = 'Частично оплачено'

else:

    invoice['payment_status'] = 'Не оплачено'

return payments
```

Функция для сохранения данных в CSV

```
def save_to_csv(data, filename, fieldnames):

    with open(filename, 'w', newline="", encoding='utf-8') as csvfile:

        writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

        writer.writeheader()

        writer.writerows(data)

    print(f'Saved {len(data)} records to {filename}')
```

Основной процесс генерации данных

```
def main():

    print('Generating customers...')

    customers = generate_customers()
```

```
    save_to_csv(customers, 'tables/customers.csv', ['customer_id', 'customer_name', 'city',  
'contact_phone', 'email'])
```

```
print('Generating part types...')
```

```
part_types = generate_part_types()
```

```
    save_to_csv(part_types, 'tables/part_types.csv', ['parttype_id', 'type_name',  
'description'])
```

```
print('Generating suppliers...')
```

```
suppliers = generate_suppliers()
```

```
    save_to_csv(suppliers, 'tables/suppliers.csv', ['supplier_id', 'supplier_name',  
'contact_phone', 'email'])
```

```
print('Generating parts...')
```

```
parts = generate_parts()
```

```
    save_to_csv(parts, 'tables/parts.csv', ['part_id', 'material', 'weight_kg', 'price_usd',  
'parttype_id', 'quantity_in_stock', 'supplier_id', 'min_stock_level', 'is_active'])
```

```
print('Generating employees...')
```

```
employees = generate_employees()
```

```
    save_to_csv(employees, 'tables/employees.csv', ['employee_id', 'first_name',  
'second_name', 'last_name', 'position', 'hire_date', 'age'])
```

```
print('Generating invoices...')
```

```
invoices = generate_invoices()
```

```
print('Generating invoice lines...')
```

```
invoice_lines = generate_invoice_lines(invoices, parts)
```

```
print('Generating invoice payments...')
```

```
payments = generate_payments(invoices)
```

```
    save_to_csv(invoice_lines, 'tables/invoice_lines.csv', ['invoiceline_id', 'invoice_id',  
'part_id', 'quantity', 'unit_price', 'line_total'])
```

```
    save_to_csv(payments, 'tables/payments.csv', ['payment_id', 'invoice_id',  
'payment_date', 'amount', 'payment_method'])
```

```
    save_to_csv(invoices, 'tables/invoices.csv', ['invoice_id', 'invoice_date',  
'total_amount', 'customer_id', 'employee_id', 'payment_status'])
```

```
if __name__ == '__main__':
```

```
    main()
```

ПРИЛОЖЕНИЕ 3