Министерство науки и высшего образования РФ

Федеральное государственное автономное

образовательное учреждение высшего образования

«**СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

Институт космических и информационных технологий

Кафедра систем искусственного интеллекта

**ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №2**

по дисциплине «Анализ больших данных»

Тема: Составление базы данных

Преподаватель \_\_\_\_\_\_\_\_\_\_\_ А. С. Полякова

подпись, дата

Студент КИ21-11Б, 032155580 \_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_ И. С. Кожевников

подпись, дата

Красноярск, 2024

Заполненные таблицы представлены на рисунках 1-12.

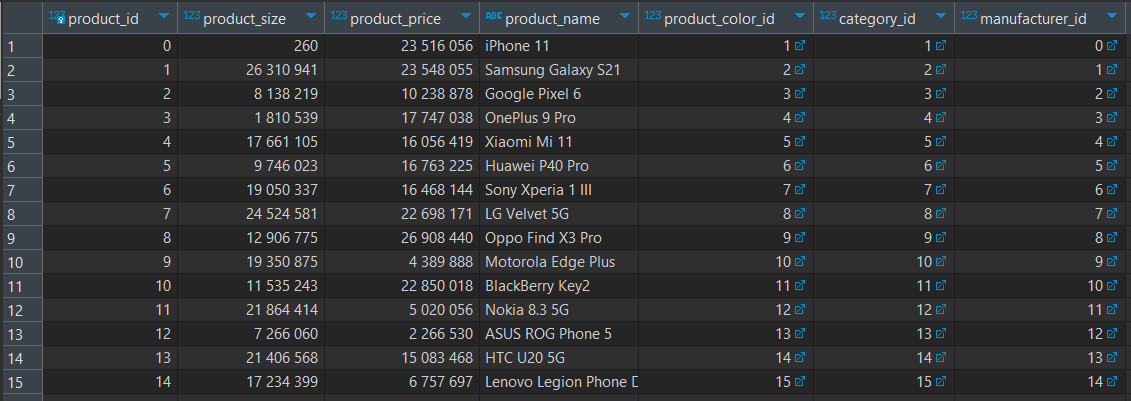


Рисунок 1 – Данные из таблицы Product

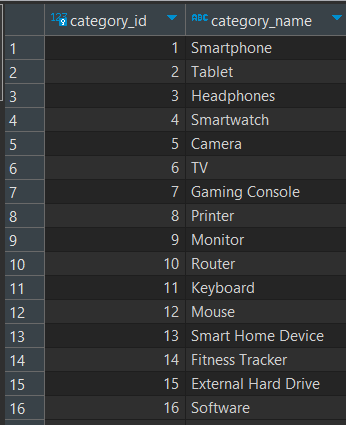


Рисунок 2 – Данные из таблицы Category

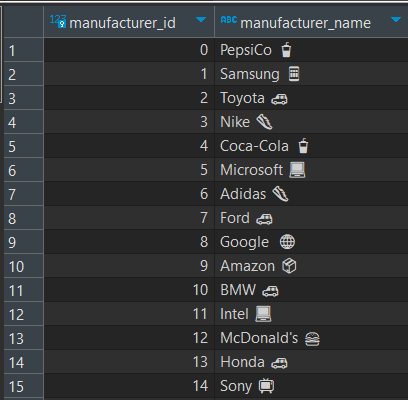


Рисунок 3 – Данные из таблицы Manufacturer

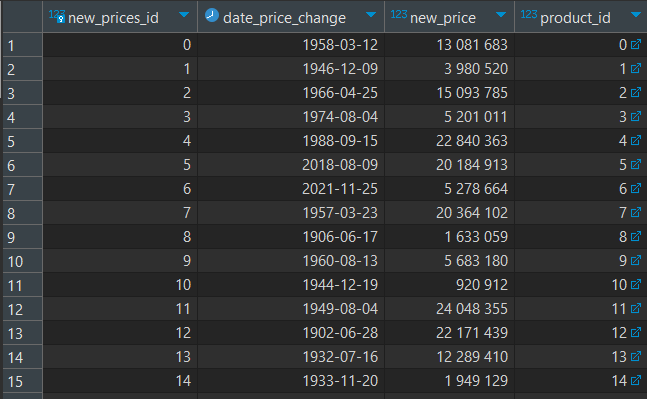


Рисунок 4 – Данные из таблицы NewPrices

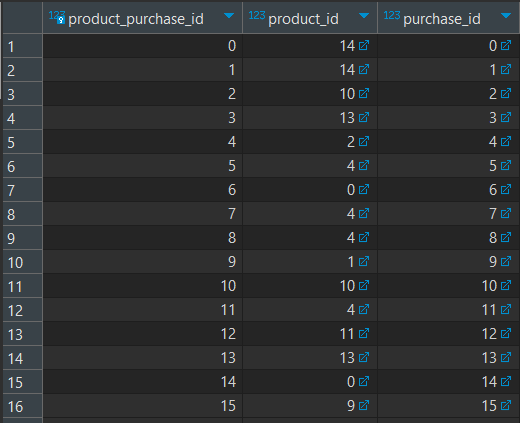


Рисунок 5 – Данные из таблицы ProductPurchase

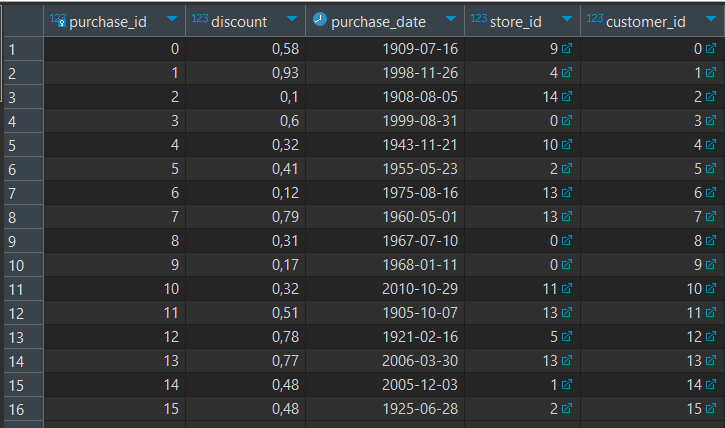


Рисунок 6 – Данные из таблицы Purchase

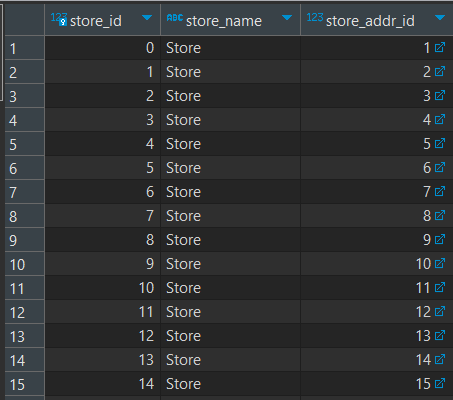


Рисунок 7 – Данные из таблицы Store

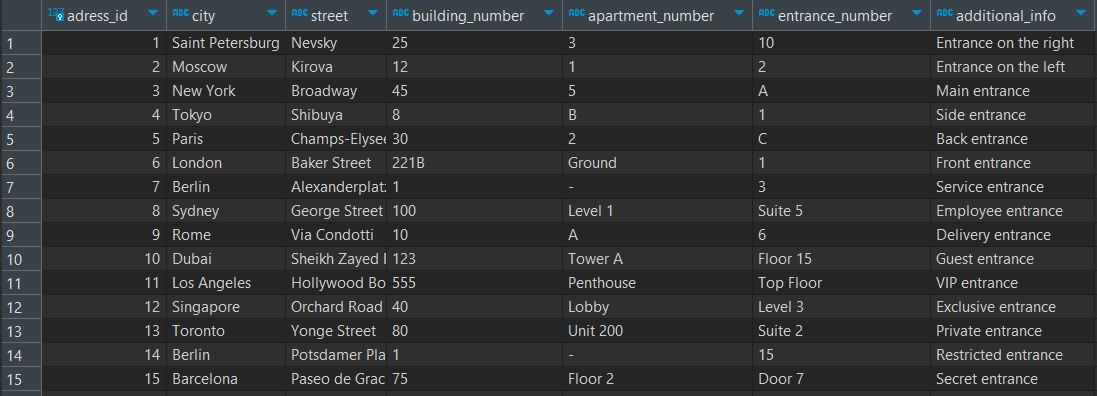


Рисунок 8 – Данные из таблицы Addresses

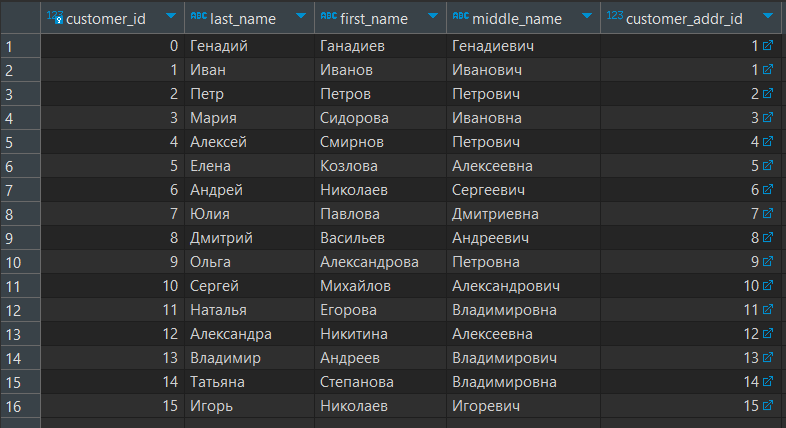


Рисунок 9 – Данные из таблицы Customers

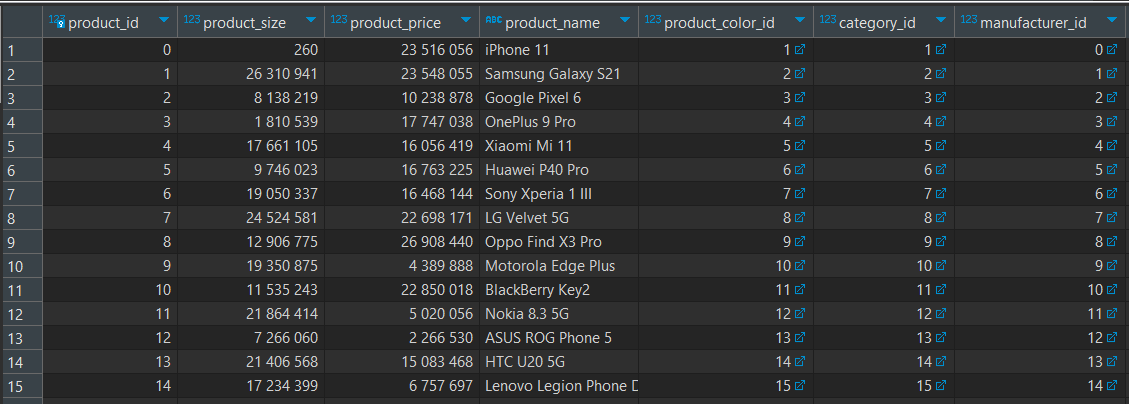


Рисунок 10 – Данные из таблицы Products

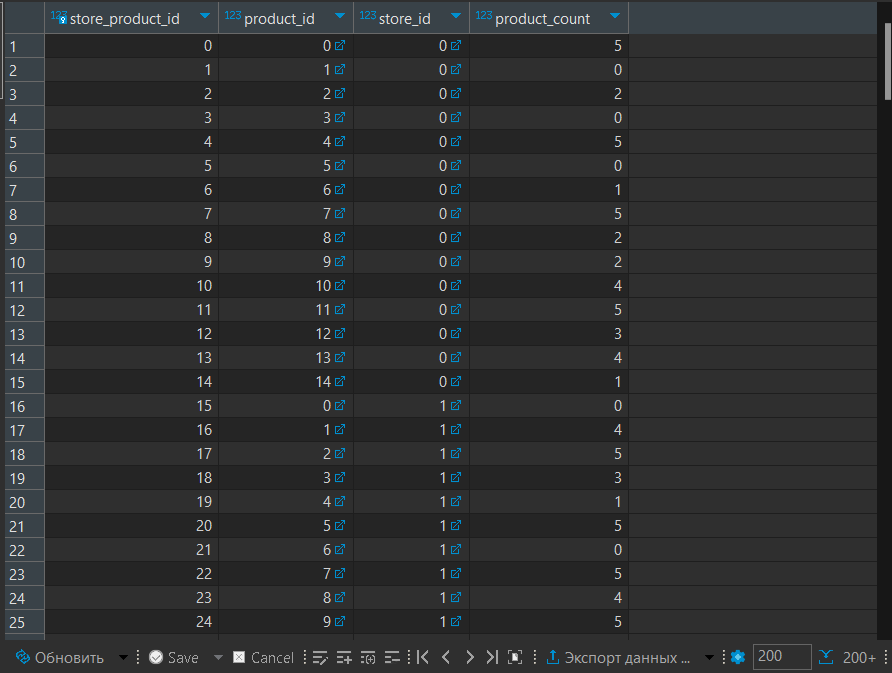


Рисунок 11 – Данные из таблицы StoreProducts

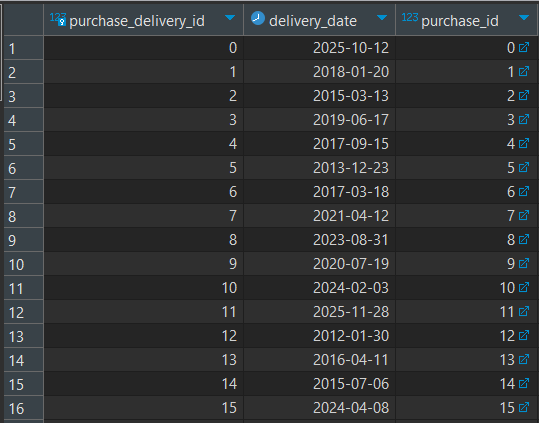


Рисунок 12 – Данные из таблицы PurchaseDelivery

На рисунках 13-16 продемонстрирован код на языке Python и SQL, который позволяет взаимодействовать с базой данных.



Рисунок 13 – Подключение к базе данных

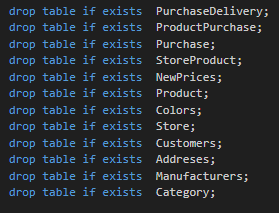


Рисунок 14 – Удаление таблиц

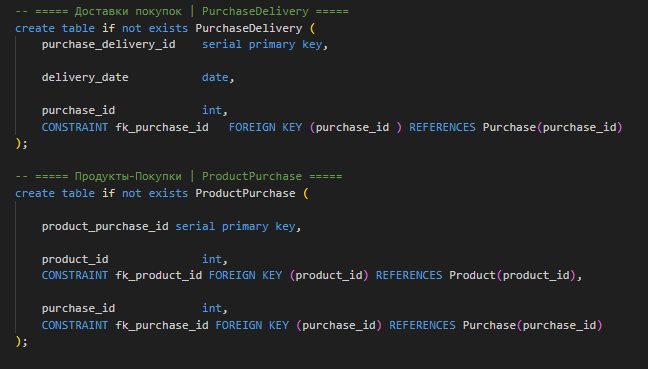


Рисунок 15 – Пример создания таблиц

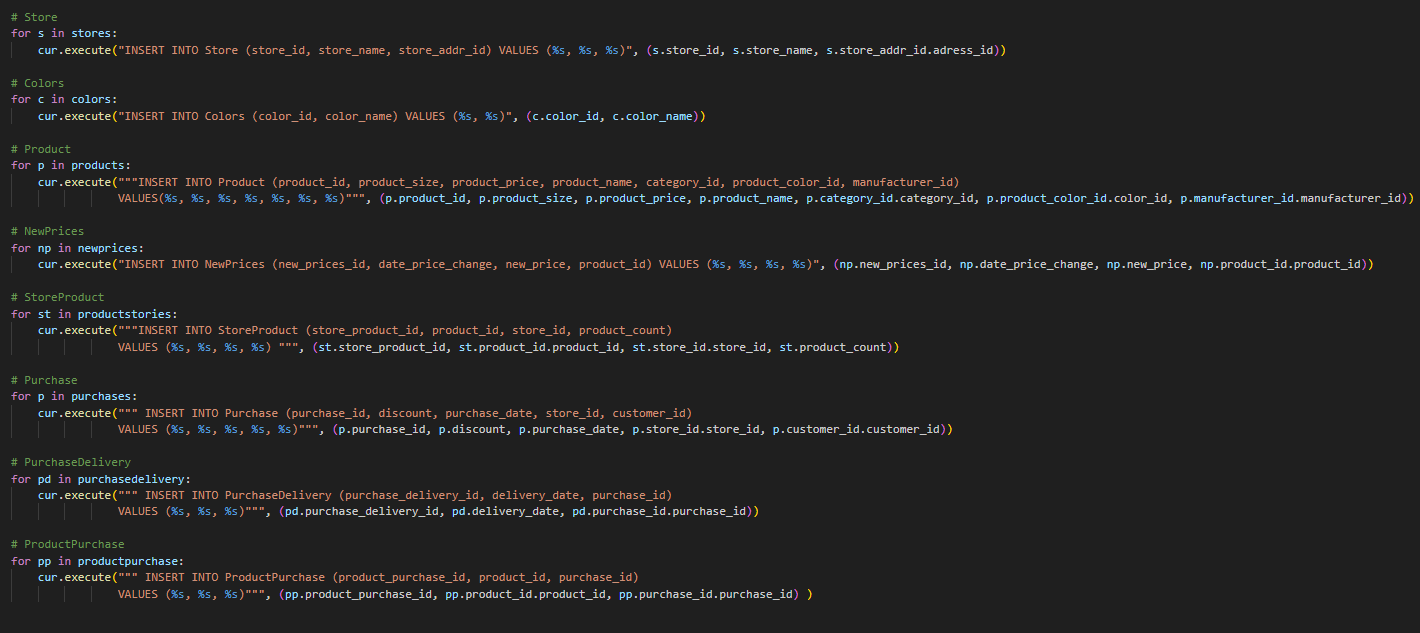


Рисунок 16 – Пример заполнения таблиц

Скрипт на заполнение базы данных python

# Адреса id [0 - 15]

addresses = [

    Addreses(1, "Saint Petersburg", "Nevsky", "25", "3", "10", "Entrance on the right"),

    Addreses(2, "Moscow", "Kirova", "12", "1", "2", "Entrance on the left"),

    Addreses(3, "New York", "Broadway", "45", "5", "A", "Main entrance"),

    Addreses(4, "Tokyo", "Shibuya", "8", "B", "1", "Side entrance"),

    Addreses(5, "Paris", "Champs-Elysees", "30", "2", "C", "Back entrance"),

    Addreses(6, "London", "Baker Street", "221B", "Ground", "1", "Front entrance"),

    Addreses(7, "Berlin", "Alexanderplatz", "1", "-", "3", "Service entrance"),

    Addreses(8, "Sydney", "George Street", "100", "Level 1", "Suite 5", "Employee entrance"),

    Addreses(9, "Rome", "Via Condotti", "10", "A", "6", "Delivery entrance"),

    Addreses(10, "Dubai", "Sheikh Zayed Road", "123", "Tower A", "Floor 15", "Guest entrance"),

    Addreses(11, "Los Angeles", "Hollywood Boulevard", "555", "Penthouse", "Top Floor", "VIP entrance"),

    Addreses(12, "Singapore", "Orchard Road", "40", "Lobby", "Level 3", "Exclusive entrance"),

    Addreses(13, "Toronto", "Yonge Street", "80", "Unit 200", "Suite 2", "Private entrance"),

    Addreses(14, "Berlin", "Potsdamer Platz", "1", "-", "15", "Restricted entrance"),

    Addreses(15, "Barcelona", "Paseo de Gracia", "75", "Floor 2", "Door 7", "Secret entrance")

]

# Производители id [0 - 15]

manufacturers = [

    Manufacturers(0, "PepsiCo 🥤"),

    Manufacturers(1, "Samsung 📱"),

    Manufacturers(2, "Toyota 🚗"),

    Manufacturers(3, "Nike 👟"),

    Manufacturers(4, "Coca-Cola 🥤"),

    Manufacturers(5, "Microsoft 💻"),

    Manufacturers(6, "Adidas 👟"),

    Manufacturers(7, "Ford 🚗"),

    Manufacturers(8, "Google  🌐"),

    Manufacturers(9, "Amazon 📦"),

    Manufacturers(10, "BMW 🚗"),

    Manufacturers(11, "Intel 💻"),

    Manufacturers(12, "McDonald's 🍔"),

    Manufacturers(13, "Honda 🚗"),

    Manufacturers(14, "Sony 📺"),

]

# Цвета id [0 - 15]

colors = [

    Colors(1, "red"),

    Colors(2, "blue"),

    Colors(3, "green"),

    Colors(4, "yellow"),

    Colors(5, "purple"),

    Colors(6, "orange"),

    Colors(7, "pink"),

    Colors(8, "brown"),

    Colors(9, "gray"),

    Colors(10, "white"),

    Colors(11, "cyan"),

    Colors(12, "magenta"),

    Colors(13, "teal"),

    Colors(14, "lime"),

    Colors(15, "indigo")

]

# Покупатели id [0 - 15]

customers = [

    Customers(0, "Генадий", "Ганадиев", "Генадиевич", addresses[0]),

    Customers(1, "Иван", "Иванов", "Иванович", addresses[0]),

    Customers(2, "Петр", "Петров", "Петрович", addresses[1]),

    Customers(3, "Мария", "Сидорова", "Ивановна", addresses[2]),

    Customers(4, "Алексей", "Смирнов", "Петрович", addresses[3]),

    Customers(5, "Елена", "Козлова", "Алексеевна", addresses[4]),

    Customers(6, "Андрей", "Николаев", "Сергеевич", addresses[5]),

    Customers(7, "Юлия", "Павлова", "Дмитриевна", addresses[6]),

    Customers(8, "Дмитрий", "Васильев", "Андреевич", addresses[7]),

    Customers(9, "Ольга", "Александрова", "Петровна", addresses[8]),

    Customers(10, "Сергей", "Михайлов", "Александрович", addresses[9]),

    Customers(11, "Наталья", "Егорова", "Владимировна", addresses[10]),

    Customers(12, "Александра", "Никитина", "Алексеевна", addresses[11]),

    Customers(13, "Владимир", "Андреев", "Владимирович", addresses[12]),

    Customers(14, "Татьяна", "Степанова", "Владимировна", addresses[13]),

    Customers(15, "Игорь", "Николаев", "Игоревич", addresses[14])

]

# Категории id [0 - 15]

categories = [

    Category(1, "Smartphone"),

    Category(2, "Tablet"),

    Category(3, "Headphones"),

    Category(4, "Smartwatch"),

    Category(5, "Camera"),

    Category(6, "TV"),

    Category(7, "Gaming Console"),

    Category(8, "Printer"),

    Category(9, "Monitor"),

    Category(10, "Router"),

    Category(11, "Keyboard"),

    Category(12, "Mouse"),

    Category(13, "Smart Home Device"),

    Category(14, "Fitness Tracker"),

    Category(15, "External Hard Drive"),

    Category(16, "Software")

]

# Магазины id [0 - 15]

stores = []

for i in range(15):

    store = Store(i, "Store", addresses[i])

    stores.append(store)

# Продукты id [0 - 15]

products = [

    Product(0, random.randint(1, 2700), random.randint(1, 27000000), "iPhone 11", categories[0], colors[0], manufacturers[0]),

    Product(1, random.randint(1, 27000000), random.randint(1, 27000000), "Samsung Galaxy S21", categories[1], colors[1], manufacturers[1]),

    Product(2, random.randint(1, 27000000), random.randint(1, 27000000), "Google Pixel 6", categories[2], colors[2], manufacturers[2]),

    Product(3, random.randint(1, 27000000), random.randint(1, 27000000), "OnePlus 9 Pro", categories[3], colors[3], manufacturers[3]),

    Product(4, random.randint(1, 27000000), random.randint(1, 27000000), "Xiaomi Mi 11", categories[4], colors[4], manufacturers[4]),

    Product(5, random.randint(1, 27000000), random.randint(1, 27000000), "Huawei P40 Pro", categories[5], colors[5], manufacturers[5]),

    Product(6, random.randint(1, 27000000), random.randint(1, 27000000), "Sony Xperia 1 III", categories[6], colors[6], manufacturers[6]),

    Product(7, random.randint(1, 27000000), random.randint(1, 27000000), "LG Velvet 5G", categories[7], colors[7], manufacturers[7]),

    Product(8, random.randint(1, 27000000), random.randint(1, 27000000), "Oppo Find X3 Pro", categories[8], colors[8], manufacturers[8]),

    Product(9, random.randint(1, 27000000), random.randint(1, 27000000), "Motorola Edge Plus", categories[9], colors[9], manufacturers[9]),

    Product(10, random.randint(1, 27000000), random.randint(1, 27000000), "BlackBerry Key2", categories[10], colors[10], manufacturers[10]),

    Product(11, random.randint(1, 27000000), random.randint(1, 27000000), "Nokia 8.3 5G", categories[11], colors[11], manufacturers[11]),

    Product(12, random.randint(1, 27000000), random.randint(1, 27000000), "ASUS ROG Phone 5", categories[12], colors[12], manufacturers[12]),

    Product(13, random.randint(1, 27000000), random.randint(1, 27000000), "HTC U20 5G", categories[13], colors[13], manufacturers[13]),

    Product(14, random.randint(1, 27000000), random.randint(1, 27000000), "Lenovo Legion Phone Duel 2", categories[14], colors[14], manufacturers[14])

]

# Новые цены id [0 - 15]

start\_date = datetime(1900, 1, 1)

end\_date = datetime(2023, 12, 31)

newprices = []

id = 0

for p in products:

    random\_days = random.randint(0, (end\_date - start\_date).days)

    random\_date = start\_date + timedelta(days=random\_days)

    newprices.append(NewPrices(id, f"{random\_date.year}-{random\_date.month}-{random\_date.day}", random.randint(1, 27000000), products[id]))

    id += 1

# Склады id [0 - 15]

id = 0

productstories = []

for s in stores:

    for p in products:

        productstories.append(StoreProduct(id, p, s, random.randint(0, 5)))

        id += 1

# Чеки id [0 - 15]

id = 0

start\_date = datetime(1900, 1, 1)

end\_date = datetime(2011, 12, 31)

purchases = []

for c in customers:

    random\_days = random.randint(0, (end\_date - start\_date).days)

    random\_date = start\_date + timedelta(days=random\_days)

    purchases.append(Purchase(id, random.randint(10, 100) / 100, f"{random\_date.year}-{random\_date.month}-{random\_date.day}", c, random.choice(stores)))

    id += 1

# Доставка id [0 - 15]

id = 0

start\_date = datetime(2011, 12, 31)

end\_date = datetime(2025, 12, 31)

purchasedelivery = []

for p in purchases:

    random\_days = random.randint(0, (end\_date - start\_date).days)

    random\_date = start\_date + timedelta(days=random\_days)

    purchasedelivery.append(PurchaseDelivery(id, f"{random\_date.year}-{random\_date.month}-{random\_date.day}", p))

    id += 1

# Покупки

id = 0

productpurchase = []

for p in purchases:

    productpurchase.append(ProductPurchase(id, random.choice(products), p))

    id += 1

# Connect to an existing database

conn = psycopg2.connect("dbname=postgres user=postgres password=rDx-Ckr-PjM-373 host=localhost port=5432")

# Open a cursor to perform database operations

cur = conn.cursor()

# Category

for c in categories:

    cur.execute("INSERT INTO Category (category\_id, category\_name) VALUES (%s, %s)", (c.category\_id, c.category\_name))

# Manufacturers

for m in manufacturers:

    cur.execute("INSERT INTO Manufacturers (manufacturer\_id, manufacturer\_name) VALUES (%s, %s)", (m.manufacturer\_id, m.manufacturer\_name))

# Addreses

for a in addresses:

    cur.execute("""INSERT INTO Addreses (adress\_id, city, street, building\_number, apartment\_number, entrance\_number, additional\_info)

                VALUES (%s, %s, %s, %s, %s, %s, %s)""", (a.adress\_id, a.city,  a.street, a.building\_number, a.apartment\_number, a.entrance\_number, a.additional\_info))

# Customers

for c in customers:

    cur.execute("""INSERT INTO Customers (customer\_id, last\_name, first\_name, middle\_name, customer\_addr\_id)

                VALUES (%s, %s, %s, %s, %s)""", (c.customer\_id, c.last\_name, c.first\_name, c.middle\_name, c.customer\_addr\_id.adress\_id))

# Store

for s in stores:

    cur.execute("INSERT INTO Store (store\_id, store\_name, store\_addr\_id) VALUES (%s, %s, %s)", (s.store\_id, s.store\_name, s.store\_addr\_id.adress\_id))

# Colors

for c in colors:

    cur.execute("INSERT INTO Colors (color\_id, color\_name) VALUES (%s, %s)", (c.color\_id, c.color\_name))

# Product

for p in products:

    cur.execute("""INSERT INTO Product (product\_id, product\_size, product\_price, product\_name, category\_id, product\_color\_id, manufacturer\_id)

                VALUES(%s, %s, %s, %s, %s, %s, %s)""", (p.product\_id, p.product\_size, p.product\_price, p.product\_name, p.category\_id.category\_id, p.product\_color\_id.color\_id, p.manufacturer\_id.manufacturer\_id))

# NewPrices

for np in newprices:

    cur.execute("INSERT INTO NewPrices (new\_prices\_id, date\_price\_change, new\_price, product\_id) VALUES (%s, %s, %s, %s)", (np.new\_prices\_id, np.date\_price\_change, np.new\_price, np.product\_id.product\_id))

# StoreProduct

for st in productstories:

    cur.execute("""INSERT INTO StoreProduct (store\_product\_id, product\_id, store\_id, product\_count)

                VALUES (%s, %s, %s, %s) """, (st.store\_product\_id, st.product\_id.product\_id, st.store\_id.store\_id, st.product\_count))

# Purchase

for p in purchases:

    cur.execute(""" INSERT INTO Purchase (purchase\_id, discount, purchase\_date, store\_id, customer\_id)

                VALUES (%s, %s, %s, %s, %s)""", (p.purchase\_id, p.discount, p.purchase\_date, p.store\_id.store\_id, p.customer\_id.customer\_id))

# PurchaseDelivery

for pd in purchasedelivery:

    cur.execute(""" INSERT INTO PurchaseDelivery (purchase\_delivery\_id, delivery\_date, purchase\_id)

                VALUES (%s, %s, %s)""", (pd.purchase\_delivery\_id, pd.delivery\_date, pd.purchase\_id.purchase\_id))

# ProductPurchase

for pp in productpurchase:

    cur.execute(""" INSERT INTO ProductPurchase (product\_purchase\_id, product\_id, purchase\_id)

                VALUES (%s, %s, %s)""", (pp.product\_purchase\_id, pp.product\_id.product\_id, pp.purchase\_id.purchase\_id) )

conn.commit()

cur.close()

conn.close()

Скрипт на развертывание базы данных SQL

-- ===== Ктегории | Category =====

create table if not exists Category (

    category\_id     serial primary key,

    category\_name   varchar(255)

);

-- ===== Производители | Manufacturer =====

create table if not exists Manufacturers (

    manufacturer\_id     serial primary key,

    manufacturer\_name   varchar(255)

);

-- ===== Адреса | Addreses =====

create table if not exists Addreses (

    adress\_id           serial primary key,

    city                varchar(100),       -- город

    street              varchar(100),       -- улица

    building\_number     varchar(10),        -- номер дома

    apartment\_number    varchar(10),        -- номер квартиры

    entrance\_number     varchar(10) null,   -- номер подъезда

    additional\_info     varchar(200)        -- доп информация

);

-- ===== Покупатели | Customer =====

create table if not exists Customers (

    customer\_id     serial primary key,

    last\_name        varchar(255),      -- имя

    first\_name       varchar(255),      -- фамилия

    middle\_name      varchar(255) null, -- отчество

    customer\_addr\_id int,

    CONSTRAINT fk\_customer\_addr\_id FOREIGN KEY (customer\_addr\_id) REFERENCES Addreses(adress\_id)

);

-- ===== Магазины | Store =====

create table if not exists Store (

    store\_id        serial primary key,

    store\_name      varchar(255),

    store\_addr\_id int,

    CONSTRAINT fk\_store\_addr\_id FOREIGN KEY (store\_addr\_id) REFERENCES Addreses(adress\_id)

);

-- ===== Цвета | Colors =====

create table if not exists Colors (

    color\_id        serial primary key,

    color\_name      varchar(255)

);

-- ===== Продукты | Product =====

create table if not exists Product (

    product\_id      serial primary key,

    product\_size        int,

--  product\_count       int,

    product\_price       numeric(15, 2),

    product\_name        varchar(255),

    -- store\_id             int,

    -- CONSTRAINT fk\_store\_id FOREIGN KEY (store\_id) REFERENCES Store(store\_id),

    product\_color\_id    int,

    CONSTRAINT fk\_product\_color\_id FOREIGN KEY (product\_color\_id) REFERENCES Colors(color\_id),

    category\_id         int,

    CONSTRAINT fk\_category\_id  FOREIGN KEY (category\_id) REFERENCES Category(category\_id),

    manufacturer\_id     int,

    CONSTRAINT fk\_manufacturer\_id  FOREIGN KEY (manufacturer\_id) REFERENCES Manufacturers(manufacturer\_id)

);

-- ===== Новая цена | NewPrices =====

create table if not exists NewPrices (

    new\_prices\_id       serial primary key,

    date\_price\_change   date,

    new\_price           numeric(15, 2),

    product\_id          int,

    CONSTRAINT fk\_product\_id  FOREIGN KEY (product\_id) REFERENCES Product(product\_id)

);

-- ===== Магазины-Продукты | StoreProduct =====

create table if not exists StoreProduct (

    store\_product\_id serial primary key,

    product\_id      int,

    CONSTRAINT fk\_product\_id FOREIGN KEY (product\_id) REFERENCES Product(product\_id),

    store\_id        int,

    CONSTRAINT fk\_store\_id FOREIGN KEY (store\_id) REFERENCES Store(store\_id),

    product\_count   int

);

-- ===== Покупки | Purchase =====

create table if not exists Purchase (

    purchase\_id         serial primary key,

    discount            numeric(5, 2),

    purchase\_date       date,

    store\_id            int,

    CONSTRAINT fk\_store\_id  FOREIGN KEY (store\_id) REFERENCES Store(store\_id),

    customer\_id         int,

    CONSTRAINT fk\_customer\_id  FOREIGN KEY (customer\_id) REFERENCES Customers(customer\_id)

);

-- ===== Доставки покупок | PurchaseDelivery =====

create table if not exists PurchaseDelivery (

    purchase\_delivery\_id    serial primary key,

    delivery\_date           date,

    purchase\_id             int,

    CONSTRAINT fk\_purchase\_id   FOREIGN KEY (purchase\_id ) REFERENCES Purchase(purchase\_id)

);

-- ===== Продукты-Покупки | ProductPurchase =====

create table if not exists ProductPurchase (

    product\_purchase\_id serial primary key,

    product\_id              int,

    CONSTRAINT fk\_product\_id FOREIGN KEY (product\_id) REFERENCES Product(product\_id),

    purchase\_id             int,

    CONSTRAINT fk\_purchase\_id FOREIGN KEY (purchase\_id) REFERENCES Purchase(purchase\_id)

);