МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МО ЭВМ

ОТЧЕТ

по лабораторной работе №3

по дисциплине «Базы данных»

Тема: Реализация базы данных с использованием ORM

Студент гр. 3384 _____ Козьмин Н.В.

Преподаватель _____ Михайлова С.В.

Санкт-Петербург

2025

Цель работы.

Развернуть PostgreSQL локально, выполнить следующие задачи:

- Описать в виде моделей таблицы из 1-й лабораторной работы
- Написать скрипт заполнения тестовыми данными: 5-10 строк на каждую таблицу, обязательно наличие связи между ними, данные приближены к реальности.
- Написать запросы к БД, отвечающие на вопросы из 1-й лабораторной работы с использованием ORM. Вывести результаты в консоль (или иной человеко-читабельный вывод)
- Описать процесс запуска: команды, зависимости.

Задание.

Вариант 8. Пусть требуется создать программную систему, предназначенную для директора продовольственного магазина. Такая система должна обеспечивать хранение сведений о магазине, об имеющихся в нем товарах, о торговых базах и товарах, хранящихся на этих базах. Магазин осуществляет закупку товаров на разных базах, предпочитая при этом закупать одни виды товара на одних базах, а другие на других. Магазин характеризуется классом, номером и имеет несколько отделов. Каждый товар в каждом магазине продается, по крайней мере, в одном отделе. Каждый отдел имеет заведующего. Товары, имеющиеся в магазине и хранящиеся на базах, характеризуются ценой, сортом и количеством. Розничные цены в магазине зависят от класса магазина. Директор магазина должен иметь возможность изменить цену товара по своему усмотрению, осуществить закупку недостающего товара на базе. Он может также закрыть один из отделов или открыть новый, при этом товары могут перемещаться из отдела в отдел. Директору могут потребоваться следующие сведения:

- Какие товары имеются в магазине (на базе)?
- Какие отсутствующие товары может заказать магазин на базе?
- Какие товары, и в каком количестве имеются в отделе магазина?

- Список заведующих отделами магазина?
- Суммарная стоимость товара в каждом отделе?
- На каких базах, и в каких количествах есть товар нужного наименования?

Выполнение.

В работе использован SQLAlchemy на Python 3.12.2. Код, указанный в приложении Б, может быть запущен и на более ранних версиях. Зависимости для pip указаны в requirements.txt. Команды для поднятия и подключения PostgreSQL находятся в commands.sh. После их запуска нужно установить пароль для пользователя и создать пустую базу данных, используя create.sql для подключенного PostgreSQL (через psql). После этого можно запускать main.py.

SQLAlchemy как ORM позволяет работать с данными, как с объектами, выявляя ошибки на этапе разработки; иметь возможность легко сменить СУБД; работать со встроенными механизмами, например, с транзакциями. Тем самым повышается простота, ускорение и безопасность разработки. В целом выбрана SQLAlchemy, так как это популярное решение, которое предназначено как раз для поставленной задачи без лишних дополнений, на языке программирования, выбранным по личным предпочтениям.

Для таблиц SQL эквивалентны классы, создаваемые в файле models.py, которые наследуются от базового и описываются также в декларативном стиле.

В populate_data.py идет требуемое заполнение данных. Для удаление используется TRUNCATE, а не delete из SQLAlchemy, так как требуется перезаписывать номера добавляемых строк, чтобы потом правильно на них ссылаться. Для добавления используется session.add_all(…) или session.add(…) с указанием, что добавляем в виде экземпляров классов. Также используется session.flush() для того, чтобы подгружались изменения, но не завершалась транзакция, как в session.commit(). Это позволяет откатить полностью все изменения при ошибке.

Для запросов ответов на вопросы используется файл queries.py. В нем используется session.query(…).all() для получения всех ответов на отдельный запрос, где в запросе используются методы, соответствующие инструкциям SQL.

В запускаемом main`e берутся значения для подключения к PostgreSQL из переменных окружения среды, как это делается обычно в рабочей практике. Затем устанавливается соединение через create_engine и создаются таблицы из базового класса. После создается сессия, которая предоставляет интерфейс для взаимодействия с базой данных. Далее пытаемся добавить данные и выполнить запросы, при исключении откатываемся и в конце заканчиваем сеанс.

Приведем ссылки на результаты работы в приложении А. Выводы результатов со скриншотами см. в приложении В.

Выводы.

В ходе выполнения лабораторной работы изучены новые методы взаимодействия с базами данных и выполнены поставленные задачи, связанные с ними: Реализовано объектно-реляционное отображение средствами SQLAlchemy с декларативным определением моделей и связей между ними. Созданы скрипты для автоматического развертывания структуры БД и заполнения тестовыми данными. Разработана функция для аналитических запросов ответов на вопросы. Решение собрано в один файл, который реализует необходимую логику. Система успешно выполняет основные требования.

ПРИЛОЖЕНИЕ А
ССЫЛКИ НА РЕЗУЛЬТАТЫ

Ссылка на pr:

https://github.com/moevm/sql-2025-3384/pull/36

ПРИЛОЖЕНИЕ Б
ИСХОДНЫЙ КОД

requirements.txt:

```
sqlalchemy>=1.4.0
psycopg2-binary>=2.9.0
```

commands.sh:

```
sudo apt update && sudo apt upgrade -y
sudo apt install postgresql postgresql-contrib -y

# psql --version
sudo service postgresql start
# sudo service postgresql status
# sudo systemctl enable postgresql # Для автозапуска

sudo -u postgres psql # Пользователь postgres
```

create.sql:

```
ALTER USER postgres WITH PASSWORD 'postgres';
CREATE DATABASE university_store_database;
```

models.py:

```
from sqlalchemy import create_engine, Column, Integer, String, Text, ForeignKey, DECIMAL
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import relationship, sessionmaker

Base = declarative_base()

class StoreClass(Base):
    __tablename__ = 'store_class'

    store_class_id = Column(Integer, primary_key=True)
    name = Column(String(50), nullable=False)
    description = Column(Text)

    stores = relationship("Store", back_populates="store_class")
    product_prices = relationship("ProductPrice", back_populates="store_class")

class TradingBase(Base):
    __tablename__ = 'trading_base'

    trading_base_id = Column(Integer, primary_key=True)
    name = Column(String(100), nullable=False)
    description = Column(Text)

    warehouse_products = relationship("WarehouseProduct", back_populates="trading_base")
    warehouse_priorities = relationship("WarehousePriority", back_populates="trading_base")

class Employee(Base):
    __tablename__ = 'employee'
```

```
        employee_id = Column(Integer, primary_key=True)
        first_name = Column(String(50), nullable=False)
        last_name = Column(String(50), nullable=False)

        managed_stores = relationship("Store", back_populates="director")
        managed_departments = relationship("Department",
back_populates="manager")

    class Store(Base):
        __tablename__ = 'store'

        store_id = Column(Integer, primary_key=True)
        name = Column(String(100), nullable=False)
        description = Column(Text)
        store_class_id = Column(Integer,
ForeignKey('store_class.store_class_id'))
        director_id = Column(Integer, ForeignKey('employee.employee_id'))

        store_class = relationship("StoreClass", back_populates="stores")
        director = relationship("Employee", back_populates="managed_stores")
        departments = relationship("Department", back_populates="store")
        warehouse_priorities = relationship("WarehousePriority",
back_populates="store")

    class Department(Base):
        __tablename__ = 'department'

        department_id = Column(Integer, primary_key=True)
        store_id = Column(Integer, ForeignKey('store.store_id'))
        manager_id = Column(Integer, ForeignKey('employee.employee_id'))
        name = Column(String(100), nullable=False)

        store = relationship("Store", back_populates="departments")
        manager = relationship("Employee",
back_populates="managed_departments")
        department_products = relationship("DepartmentProduct",
back_populates="department")

    class Product(Base):
        __tablename__ = 'product'

        article = Column(Integer, primary_key=True)
        name = Column(String(100), nullable=False)
        sort = Column(String(50))

        department_products = relationship("DepartmentProduct",
back_populates="product")
        warehouse_products = relationship("WarehouseProduct",
back_populates="product")
        product_prices = relationship("ProductPrice", back_populates="product")
        warehouse_priorities = relationship("WarehousePriority",
back_populates="product")

    class DepartmentProduct(Base):
        __tablename__ = 'department_product'

        department_id = Column(Integer, ForeignKey('department.department_id'),
primary_key=True)
        article = Column(Integer, ForeignKey('product.article'),
primary_key=True)
        count = Column(Integer, default=0)
```

```
        department = relationship("Department",
back_populates="department_products")
        product = relationship("Product", back_populates="department_products")

    class WarehouseProduct(Base):
        __tablename__ = 'warehouse_product'

        trading_base_id = Column(Integer,
ForeignKey('trading_base.trading_base_id'), primary_key=True)
        article = Column(Integer, ForeignKey('product.article'),
primary_key=True)
        count = Column(Integer, default=0)
        price = Column(DECIMAL(10, 2), nullable=False)

        trading_base = relationship("TradingBase",
back_populates="warehouse_products")
        product = relationship("Product", back_populates="warehouse_products")

    class ProductPrice(Base):
        __tablename__ = 'product_price'

        store_class_id = Column(Integer,
ForeignKey('store_class.store_class_id'), primary_key=True)
        article = Column(Integer, ForeignKey('product.article'),
primary_key=True)
        price = Column(DECIMAL(10, 2), nullable=False)

        store_class = relationship("StoreClass",
back_populates="product_prices")
        product = relationship("Product", back_populates="product_prices")

    class WarehousePriority(Base):
        __tablename__ = 'warehouse_priority'

        article = Column(Integer, ForeignKey('product.article'),
primary_key=True)
        store_id = Column(Integer, ForeignKey('store.store_id'),
primary_key=True)
        trading_base_id = Column(Integer,
ForeignKey('trading_base.trading_base_id'), primary_key=True)
        priority = Column(Integer, default=5)

        product = relationship("Product",
back_populates="warehouse_priorities")
        store = relationship("Store", back_populates="warehouse_priorities")
        trading_base = relationship("TradingBase",
back_populates="warehouse_priorities")
```

populate_data.py:

```
    from sqlalchemy import create_engine, text
    from sqlalchemy.orm import sessionmaker
    from models import Base, StoreClass, TradingBase, Employee, Store,
Department, Product, DepartmentProduct, WarehouseProduct, ProductPrice,
WarehousePriority

    def populate_test_data(session):
        # Очистка существующих данных
        truncate_sql = """
        TRUNCATE TABLE
            department_product,
            department,
            warehouse_priority,
```

```
            store,
            product_price,
            store_class,
            warehouse_product,
            trading_base,
            employee,
            product
        RESTART IDENTITY;
        """

        session.execute(text(truncate_sql))

        # Классы магазинов
        store_classes = [
            StoreClass(name='Элитный', description='Магазины премиум-класса с широким ассортиментом'),
            StoreClass(name='Стандарт', description='Магазины среднего ценового сегмента'),
            StoreClass(name='Эконом', description='Бюджетные магазины'),
            StoreClass(name='Специализированный', description='Магазины узкой направленности'),
            StoreClass(name='Гипермаркет', description='Крупные торговые центры')
        ]
        session.add_all(store_classes)
        session.flush()

        # Торговые базы
        trading_bases = [
            TradingBase(name='Центральная база "Северная"', description='Крупнейший распределительный центр в северном регионе'),
            TradingBase(name='Южный распределительный центр', description='Современный логистический комплекс южного направления'),
            TradingBase(name='Западный складской комплекс', description='Складские помещения с системой климат-контроля'),
            TradingBase(name='Восточная база снабжения', description='База снабжения для розничных сетей восточного региона'),
            TradingBase(name='Центральный логистический центр', description='Основной хаб для федеральных сетей')
        ]
        session.add_all(trading_bases)
        session.flush()

        # Сотрудники
        employees = [
            Employee(first_name='Иван', last_name='Петров'),
            Employee(first_name='Мария', last_name='Сидорова'),
            Employee(first_name='Алексей', last_name='Козлов'),
            Employee(first_name='Ольга', last_name='Николаева'),
            Employee(first_name='Сергей', last_name='Васильев'),
            Employee(first_name='Елена', last_name='Федорова'),
            Employee(first_name='Дмитрий', last_name='Орлов'),
            Employee(first_name='Анна', last_name='Морозова'),
            Employee(first_name='Павел', last_name='Семенов'),
            Employee(first_name='Ирина', last_name='Волкова')
        ]
        session.add_all(employees)
        session.flush()

        # Магазины
        stores = [
            Store(name='Супермаркет "Восток"', description='Крупный супермаркет в центре города',
```

```
                    store_class_id=store_classes[1].store_class_id,
director_id=employees[0].employee_id),
                Store(name='Гипермаркет "Мега"', description='Торговый центр с
полным ассортиментом',
                    store_class_id=store_classes[4].store_class_id,
director_id=employees[3].employee_id),
                Store(name='Магазин "Эконом"', description='Бюджетный магазин для
ежедневных покупок',
                    store_class_id=store_classes[2].store_class_id,
director_id=employees[5].employee_id),
                Store(name='Премиум маркет "Люкс"', description='Магазин премиум-
класса',
                    store_class_id=store_classes[0].store_class_id,
director_id=employees[8].employee_id),
                Store(name='Спецмагазин                                    "Техника"',
description='Специализированный магазин электроники',
                    store_class_id=store_classes[3].store_class_id,
director_id=employees[3].employee_id)
            ]
            session.add_all(stores)
            session.flush()

            # Отделы
            departments = [
                Department(store_id=stores[0].store_id,
manager_id=employees[1].employee_id, name='Молочный отдел'),
                Department(store_id=stores[0].store_id,
manager_id=employees[2].employee_id, name='Хлебный отдел'),
                Department(store_id=stores[1].store_id,
manager_id=employees[4].employee_id, name='Овощной отдел'),
                Department(store_id=stores[1].store_id,
manager_id=employees[6].employee_id, name='Мясной отдел'),
                Department(store_id=stores[2].store_id,
manager_id=employees[7].employee_id, name='Бакалея'),
                Department(store_id=stores[3].store_id,
manager_id=employees[9].employee_id, name='Гастрономия'),
                Department(store_id=stores[3].store_id,
manager_id=employees[1].employee_id, name='Кондитерский отдел'),
                Department(store_id=stores[4].store_id,
manager_id=employees[2].employee_id, name='Бытовая техника'),
                Department(store_id=stores[4].store_id,
manager_id=employees[4].employee_id, name='Электроника'),
                Department(store_id=stores[0].store_id,
manager_id=employees[6].employee_id, name='Напитки')
            ]
            session.add_all(departments)
            session.flush()

            # Товары
            products = [
                Product(name='Молоко 2,5%', sort='Пастеризованное'),
                Product(name='Хлеб Бородинский', sort='Ржаной'),
                Product(name='Картофель', sort='Отборный'),
                Product(name='Курица охлажденная', sort='Бройлер'),
                Product(name='Рис круглый', sort='Высший сорт'),
                Product(name='Кофе молотый', sort='Арабика'),
                Product(name='Шоколад горький', sort='Премиум'),
                Product(name='Чай черный', sort='Цейлон'),
                Product(name='Сок яблочный', sort='Осветленный'),
                Product(name='Телевизор LED', sort='Smart TV'),
                Product(name='Смартфон', sort='Флагман'),
                Product(name='Йогурт натуральный', sort='Без добавок'),
                Product(name='Сыр Российский', sort='Полутвердый'),
```

```
    Product(name='Колбаса докторская', sort='Вареная'),
    Product(name='Печенье овсяное', sort='С шоколадом')
]
session.add_all(products)
session.flush()

# Товары в отделах
department_products = [
    (1, 1, 0), (1, 12, 30), (1, 13, 25),
    (2, 2, 40), (2, 15, 45),
    (3, 3, 100), (3, 5, 60),
    (4, 4, 35), (4, 14, 28),
    (5, 5, 55), (5, 6, 20), (5, 7, 35),
    (6, 7, 15), (6, 8, 25), (6, 13, 18),
    (7, 7, 30), (7, 15, 40),
    (8, 10, 8), (9, 11, 12),
    (10, 1, 25), (10, 9, 35)
]

for dept_id, art, cnt in department_products:
    session.add(DepartmentProduct(
        department_id=departments[dept_id-1].department_id,
        article=products[art-1].article,
        count=cnt
    ))

# Товары на складах баз
warehouse_products = [
    (1, 1, 200, 85.50), (1, 2, 150, 45.00), (1, 3, 500, 35.00),
    (2, 4, 180, 320.00), (2, 5, 300, 95.00), (2, 6, 120, 450.00),
    (3, 7, 90, 180.00), (3, 8, 200, 120.00), (3, 9, 150, 110.00),
    (4, 10, 25, 25000.00), (4, 11, 40, 45000.00),
    (5, 12, 100, 65.00), (5, 13, 80, 580.00), (5, 14, 60, 420.00),
    (1, 15, 120, 85.00), (2, 1, 180, 82.00)
]

for base_id, art, cnt, price in warehouse_products:
    session.add(WarehouseProduct(
        trading_base_id=trading_bases[base_id-1].trading_base_id,
        article=products[art-1].article,
        count=cnt,
        price=price
    ))

# Цены товаров по классам магазинов
product_prices = [
    (1, 1, 120.00), (1, 2, 65.00), (1, 3, 50.00),
    (2, 1, 95.00), (2, 2, 48.00), (2, 3, 38.00),
    (3, 1, 80.00), (3, 2, 40.00), (3, 3, 30.00),
    (1, 7, 250.00), (2, 7, 200.00), (3, 7, 150.00),
    (4, 10, 28000.00), (5, 10, 27000.00)
]

for class_id, art, price in product_prices:
    session.add(ProductPrice(
        store_class_id=store_classes[class_id-1].store_class_id,
        article=products[art-1].article,
        price=price
    ))

# Приоритеты поставок
warehouse_priorities = [
    (1, 1, 1, 1), (1, 2, 1, 2), (1, 3, 1, 1),
```

```
            (2, 1, 1, 2), (2, 2, 1, 3), (2, 3, 1, 2),
            (3, 1, 1, 1), (3, 2, 1, 1), (3, 3, 1, 1),
            (10, 4, 4, 1), (10, 5, 4, 1), (11, 5, 4, 1),
            (7, 4, 3, 2), (7, 1, 3, 3),
            (1, 1, 2, 2), (2, 1, 2, 3)
        ]

        for art, store_id, base_id, priority in warehouse_priorities:
            session.add(WarehousePriority(
                article=products[art-1].article,
                store_id=stores[store_id-1].store_id,
                trading_base_id=trading_bases[base_id-1].trading_base_id,
                priority=priority
            ))

        session.commit()
        print("Тестовые данные успешно добавлены!")
```

queries.py:

```
    from sqlalchemy import func, and_, or_, not_, case, select
    from sqlalchemy.orm import aliased
    from models import WarehousePriority, ProductPrice, WarehouseProduct,
DepartmentProduct, Product, Department, Store, Employee, TradingBase, StoreClass

    def execute_queries(session):
        print("=" * 80)
        print("ЗАПРОСЫ К БАЗЕ ДАННЫХ УНИВЕРСИТЕТСКОГО МАГАЗИНА")
        print("=" * 80)

        # 1.1 Какие товары имеются в магазине?
        print("\n1.1 ТОВАРЫ В МАГАЗИНЕ 'Супермаркет \"Восток\"' (ID=1):")
        print("-" * 60)

        store_products = (session.query(
            Store.name.label('store_name'),
            Department.name.label('department_name'),
            Product.article,
            Product.name.label('product_name'),
            Product.sort.label('product_sort'),
            DepartmentProduct.count.label('quantity'),
            ProductPrice.price.label('current_price'),
            (DepartmentProduct.count * ProductPrice.price).label('total_value')
        )
        .select_from(Store)
        .join(Department, Store.store_id == Department.store_id)
                .join(DepartmentProduct,     Department.department_id     ==
DepartmentProduct.department_id)
        .join(Product, DepartmentProduct.article == Product.article)
        .join(ProductPrice, and_(
            Product.article == ProductPrice.article,
            Store.store_class_id == ProductPrice.store_class_id
        ))
        .filter(Store.store_id == 1)
        .order_by(Department.name, Product.name)
        .all())

        for product in store_products:
                    print(f"Отдел:    {product.department_name:20}    Товар:
{product.product_name:25} "
                                    f"Кол-во:    {product.quantity:3}    Цена:
{float(product.current_price):8.2f} "
                  f"Сумма: {float(product.total_value):10.2f}")
```

```
        # 1.2 Какие товары имеются на базе?
        print("\n1.2 ТОВАРЫ НА ТОРГОВОЙ БАЗЕ 'Южный распределительный центр' (ID=2):")
        print("-" * 60)

        base_products = (session.query(
            TradingBase.name.label('trading_base_name'),
            Product.article,
            Product.name.label('product_name'),
            Product.sort.label('product_sort'),
            WarehouseProduct.count.label('available_quantity'),
            WarehouseProduct.price.label('base_price'),
                                        (WarehouseProduct.count * WarehouseProduct.price).label('total_value')
        )
        .select_from(TradingBase)
                .join(WarehouseProduct, TradingBase.trading_base_id == WarehouseProduct.trading_base_id)
        .join(Product, WarehouseProduct.article == Product.article)
        .filter(TradingBase.trading_base_id == 2)
        .filter(WarehouseProduct.count > 0)
        .order_by(Product.name)
        .all())

        for product in base_products:
                        print(f"Товар:  {product.product_name:25}  Кол-во: {product.available_quantity:4} "
                              f"Цена:  {float(product.base_price):8.2f}  Сумма: {float(product.total_value):12.2f}")

        # 2. Какие отсутствующие товары может заказать магазин на базе?
        print("\n2. ОТСУТСТВУЮЩИЕ ТОВАРЫ ДЛЯ ЗАКАЗА МАГАЗИНОМ 'Супермаркет \"Восток\"' (ID=1):")
        print("-" * 60)

        # Подзапрос для товаров с нулевым количеством в магазине
        zero_products_subquery = select(DepartmentProduct.article).\
                        join(Department, DepartmentProduct.department_id == Department.department_id).\
            where(and_(
                Department.store_id == 1,
                DepartmentProduct.count == 0
            )).\
            scalar_subquery()

        orderable_products = (session.query(
            Store.name.label('store_name'),
            Product.name.label('product_name'),
            TradingBase.name.label('trading_base'),
            WarehouseProduct.count.label('available_quantity'),
            WarehouseProduct.price.label('base_price'),
            WarehousePriority.priority
        )
        .select_from(Product)
        .join(WarehouseProduct, Product.article == WarehouseProduct.article)
                .join(TradingBase, WarehouseProduct.trading_base_id == TradingBase.trading_base_id)
        .join(Store, Store.store_id == 1)
        .join(WarehousePriority, and_(
            WarehousePriority.store_id == Store.store_id,
            WarehousePriority.article == Product.article,
            WarehousePriority.trading_base_id == TradingBase.trading_base_id
```

```
        ))
        .filter(Product.article.in_(zero_products_subquery))
        .filter(WarehouseProduct.count >= 1)
        .order_by(Product.name, WarehousePriority.priority)
        .all())

        for product in orderable_products:
                        print(f"Товар:    {product.product_name:25}    База:
{product.trading_base:30} "
                              f"Доступно:  {product.available_quantity:3}  Цена:
{float(product.base_price):8.2f} "
                  f"Приоритет: {product.priority}")

         # 2.доп Какие отсутствующие товары может заказать магазин на базе (в
том числе, которых не было, но для которых есть приоритет)?
        print("\n2.доп ОТСУТСТВУЮЩИЕ ТОВАРЫ ДЛЯ ЗАКАЗА МАГАЗИНОМ (включая новые
товары с приоритетами) 'Супермаркет \"Восток\"' (ID=1):")
        print("-" * 60)

        # Подзапрос для товаров, которые есть в магазине
        store_products_subquery = select(DepartmentProduct.article).\
                        join(Department,   DepartmentProduct.department_id   ==
Department.department_id).\
            where(Department.store_id == 1)

        # Подзапрос для товаров с нулевым количеством
        zero_count_subquery = select(DepartmentProduct.article).\
                        join(Department,   DepartmentProduct.department_id   ==
Department.department_id).\
            where(and_(
                Department.store_id == 1,
                DepartmentProduct.count == 0
            ))

        extended_orderable_products = (session.query(
            Store.name.label('store_name'),
            Product.name.label('product_name'),
            TradingBase.name.label('trading_base'),
            WarehouseProduct.count.label('available_quantity'),
            WarehouseProduct.price.label('base_price'),
            WarehousePriority.priority,
            case(
                (Product.article.in_(zero_count_subquery), 'ЗАКОНЧИЛСЯ'),
                   (not_(Product.article.in_(store_products_subquery)), 'НОВЫЙ
ТОВАР'),
                else_='ДРУГОЙ'
            ).label('status')
        )
        .select_from(Product)
        .join(WarehouseProduct, Product.article == WarehouseProduct.article)
                .join(TradingBase,     WarehouseProduct.trading_base_id     ==
TradingBase.trading_base_id)
        .join(Store, Store.store_id == 1)
        .join(WarehousePriority, and_(
            WarehousePriority.store_id == Store.store_id,
            WarehousePriority.article == Product.article,
            WarehousePriority.trading_base_id == TradingBase.trading_base_id
        ))
        .filter(or_(
            Product.article.in_(zero_count_subquery),
            not_(Product.article.in_(store_products_subquery))
        ))
        .filter(WarehouseProduct.count >= 0)
```

```
            .order_by(Product.name, WarehousePriority.priority)
            .all())

            if extended_orderable_products:
                for product in extended_orderable_products:
                     status_text = "ЗАКОНЧИЛСЯ" if product.status == 'ЗАКОНЧИЛСЯ'
else "НОВЫЙ ТОВАР (может быть добавлен)"
                              print(f"Товар:   {product.product_name:25}   Статус:
{status_text:25} "
                                    f"База:  {product.trading_base:25}  Приоритет:
{product.priority}")
            else:
                print("Нет товаров для заказа")

            # 3. Какие товары, и в каком количестве имеются в отделе магазина?
            print("\n3. ТОВАРЫ В ОТДЕЛЕ 'Молочный отдел' (ID=1):")
            print("-" * 60)

            department_products = (session.query(
                Store.name.label('store_name'),
                Department.name.label('department_name'),
                Product.name.label('product_name'),
                DepartmentProduct.count.label('quantity')
            )
            .select_from(Store)
            .join(Department, Store.store_id == Department.store_id)
                     .join(DepartmentProduct,     Department.department_id     ==
DepartmentProduct.department_id)
            .join(Product, DepartmentProduct.article == Product.article)
            .filter(Department.department_id == 1)
            .order_by(Product.name)
            .all())

            for product in department_products:
                            print(f"Магазин:    {product.store_name:25}    Отдел:
{product.department_name:20} "
                                    f"Товар:   {product.product_name:25}   Кол-во:
{product.quantity}")

            # 4. Список заведующих отделами магазина
              print("\n4.  ЗАВЕДУЮЩИЕ  ОТДЕЛАМИ  МАГАЗИНА  'Супермаркет  \"Восток\"'
(ID=1):")
            print("-" * 60)

            department_managers = (session.query(
                Store.name.label('store_name'),
                Department.name.label('department_name'),
                                   func.concat(Employee.first_name,      '      ',
Employee.last_name).label('manager_name')
            )
            .select_from(Store)
            .join(Department, Store.store_id == Department.store_id)
            .join(Employee, Department.manager_id == Employee.employee_id)
            .filter(Store.store_id == 1)
            .order_by(Department.name)
            .all())

            for manager in department_managers:
                        print(f"Отдел:   {manager.department_name:20}   Заведующий:
{manager.manager_name}")

            # 5. Суммарная стоимость товара в каждом отделе
            print("\n5. СУММАРНАЯ СТОИМОСТЬ ТОВАРОВ ПО ОТДЕЛАМ:")
```

```
        print("-" * 60)

        department_values = (session.query(
            Store.name.label('store_name'),
            Department.name.label('department_name'),
                                        func.sum(DepartmentProduct.count       *
ProductPrice.price).label('total_value')
        )
        .select_from(Store)
        .join(Department, Store.store_id == Department.store_id)
                 .join(DepartmentProduct,      Department.department_id      ==
DepartmentProduct.department_id)
        .join(ProductPrice, and_(
            DepartmentProduct.article == ProductPrice.article,
            Store.store_class_id == ProductPrice.store_class_id
        ))
        .group_by(Store.name, Department.name)
                           .order_by(func.sum(DepartmentProduct.count          *
ProductPrice.price).desc())
        .all())

        for dept in department_values:
                            print(f"Магазин:     {dept.store_name:25}     Отдел:
{dept.department_name:20} "
                  f"Сумма: {float(dept.total_value):12.2f}")

          # 6. На каких базах, и в каких количествах есть товар нужного
наименования?
        print("\n6. ПОИСК ТОВАРА 'Молоко' ПО БАЗАМ:")
        print("-" * 60)

        search_products = (session.query(
            Product.name.label('product_name'),
            TradingBase.name.label('trading_base'),
            TradingBase.description.label('base_description'),
            WarehouseProduct.count.label('available_quantity'),
            WarehouseProduct.price.label('price_per_unit')
        )
        .select_from(Product)
        .join(WarehouseProduct, Product.article == WarehouseProduct.article)
                 .join(TradingBase,     WarehouseProduct.trading_base_id     ==
TradingBase.trading_base_id)
        .filter(Product.name.ilike('%молоко%'))
        .order_by(WarehouseProduct.count.desc())
        .all())

        for product in search_products:
                           print(f"Товар:     {product.product_name:25}     База:
{product.trading_base:35} "
                                f"Кол-во:   {product.available_quantity:4}   Цена:
{float(product.price_per_unit):8.2f}")
```

main.py:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker
from models import Base
from populate_data import populate_test_data
from queries import execute_queries
import os

def main():
    # Параметры подключения к PostgreSQL
```

```
        db_host = os.getenv('DB_HOST', 'localhost')
        db_port = os.getenv('DB_PORT', '5432')
        db_name = os.getenv('DB_NAME', 'university_store_database')
        db_user = os.getenv('DB_USER', 'postgres')
        db_password = os.getenv('DB_PASSWORD', 'postgres')

        # Создание подключения к PostgreSQL
        database_url                                                    =
f'postgresql://{db_user}:{db_password}@{db_host}:{db_port}/{db_name}'
        engine = create_engine(
            database_url,
            echo=False,
        )

        # Создание таблиц
        print("Создание таблиц в PostgreSQL...")
        Base.metadata.create_all(engine)

        # Создание сессии
        Session = sessionmaker(bind=engine)
        session = Session()

        try:
            # Заполнение тестовыми данными
            print("Заполнение базы данных тестовыми данными...")
            populate_test_data(session)

            # Выполнение запросов
            execute_queries(session)

        except Exception as e:
            print(f"Произошла ошибка: {e}")
            session.rollback()
        finally:
            session.close()

    if __name__ == "__main__":
        main()
```

ПРИЛОЖЕНИЕ В
ВЫВОДЫ РЕЗУЛЬТАТОВ

Терминальный вывод:

```
python3 main.py
Создание таблиц в PostgreSQL...
Заполнение базы данных тестовыми данными...
Тестовые данные успешно добавлены!
================================================================================
ЗАПРОСЫ К БАЗЕ ДАННЫХ УНИВЕРСИТЕТСКОГО МАГАЗИНА
================================================================================

1.1 ТОВАРЫ В МАГАЗИНЕ 'Супермаркет "Восток"' (ID=1):
----------------------------------------------------------------
Отдел: Молочный отдел        Товар: Молоко 2,5%                Кол-во:    0 Цена:
95.00 Сумма:       0.00
Отдел: Напитки               Товар: Молоко 2,5%                Кол-во:  25 Цена:
95.00 Сумма:    2375.00
Отдел: Хлебный отдел         Товар: Хлеб Бородинский           Кол-во:  40 Цена:
48.00 Сумма:    1920.00

1.2 ТОВАРЫ НА ТОРГОВОЙ БАЗЕ 'Южный распределительный центр' (ID=2):
----------------------------------------------------------------
Товар: Кофе молотый              Кол-во:  120 Цена:   450.00 Сумма:     54000.00
Товар: Курица охлажденная        Кол-во:  180 Цена:   320.00 Сумма:     57600.00
Товар: Молоко 2,5%               Кол-во:  180 Цена:    82.00 Сумма:     14760.00
Товар: Рис круглый               Кол-во:  300 Цена:    95.00 Сумма:     28500.00

2. ОТСУТСТВУЮЩИЕ ТОВАРЫ ДЛЯ ЗАКАЗА МАГАЗИНОМ 'Супермаркет "Восток"' (ID=1):
----------------------------------------------------------------
Товар: Молоко 2,5%                База: Центральная база "Северная"    Доступно:
200 Цена:    85.50 Приоритет: 1
Товар: Молоко 2,5%                База: Южный распределительный центр  Доступно:
180 Цена:    82.00 Приоритет: 2

2.доп ОТСУТСТВУЮЩИЕ ТОВАРЫ ДЛЯ ЗАКАЗА МАГАЗИНОМ (включая новые товары с
приоритетами) 'Супермаркет "Восток"' (ID=1):
----------------------------------------------------------------
Товар: Картофель                  Статус: НОВЫЙ ТОВАР (может быть добавлен) База:
Центральная база "Северная" Приоритет: 1
Товар: Молоко 2,5%             Статус: ЗАКОНЧИЛСЯ              База: Центральная
база "Северная" Приоритет: 1
Товар: Молоко 2,5%                Статус: ЗАКОНЧИЛСЯ                 База: Южный
распределительный центр Приоритет: 2
Товар: Шоколад горький            Статус: НОВЫЙ ТОВАР (может быть добавлен) База:
Западный складской комплекс Приоритет: 3

3. ТОВАРЫ В ОТДЕЛЕ 'Молочный отдел' (ID=1):
----------------------------------------------------------------
Магазин: Супермаркет "Восток"        Отдел: Молочный отдел         Товар: Йогурт
натуральный        Кол-во: 30
Магазин: Супермаркет "Восток"      Отдел: Молочный отдел       Товар: Молоко 2,5%
Кол-во: 0
Магазин: Супермаркет "Восток"     Отдел: Молочный отдел      Товар: Сыр Российский
Кол-во: 25

4. ЗАВЕДУЮЩИЕ ОТДЕЛАМИ МАГАЗИНА 'Супермаркет "Восток"' (ID=1):
----------------------------------------------------------------
Отдел: Молочный отдел       Заведующий: Мария Сидорова
Отдел: Напитки              Заведующий: Дмитрий Орлов
```

```
Отдел: Хлебный отдел          Заведующий: Алексей Козлов

5. СУММАРНАЯ СТОИМОСТЬ ТОВАРОВ ПО ОТДЕЛАМ:
------------------------------------------------------------
Магазин: Спецмагазин "Техника"     Отдел: Бытовая техника      Сумма:    224000.00
Магазин: Премиум маркет "Люкс"     Отдел: Кондитерский отдел   Сумма:      7500.00
Магазин: Магазин "Эконом"          Отдел: Бакалея              Сумма:      5250.00
Магазин: Премиум маркет "Люкс"     Отдел: Гастрономия          Сумма:      3750.00
Магазин: Супермаркет "Восток"      Отдел: Напитки              Сумма:      2375.00
Магазин: Супермаркет "Восток"      Отдел: Хлебный отдел        Сумма:      1920.00
Магазин: Супермаркет "Восток"      Отдел: Молочный отдел       Сумма:         0.00

6. ПОИСК ТОВАРА 'Молоко' ПО БАЗАМ:
------------------------------------------------------------
Товар: Молоко 2,5%                 База: Центральная база "Северная"          Кол-
во:  200 Цена:    85.50
Товар: Молоко 2,5%                 База: Южный распределительный центр        Кол-
во:  180 Цена:    82.00
```

```
kozmi@ideapad3-82rn:/mnt/c/Users/kozmi/GoogleDrive/Development/Databases/sql-2025-3384/3384_Kozmin/lab3$ python3 main.py
Создание таблиц в PostgreSQL...
Заполнение базы данных тестовыми данными...
Тестовые данные успешно добавлены!
================================================================================
ЗАПРОСЫ К БАЗЕ ДАННЫХ УНИВЕРСИТЕТСКОГО МАГАЗИНА
================================================================================

1.1 ТОВАРЫ В МАГАЗИНЕ 'Супермаркет "Восток"' (ID=1):
------------------------------------------------------------
Отдел: Молочный отдел       Товар: Молоко 2,5%             Кол-во:   0 Цена:    95.00 Сумма:       0.00
Отдел: Напитки              Товар: Молоко 2,5%             Кол-во:  25 Цена:    95.00 Сумма:    2375.00
Отдел: Хлебный отдел        Товар: Хлеб Бородинский        Кол-во:  40 Цена:    48.00 Сумма:    1920.00

1.2 ТОВАРЫ НА ТОРГОВОЙ БАЗЕ 'Южный распределительный центр' (ID=2):
------------------------------------------------------------
Товар: Кофе молотый             Кол-во:  120 Цена:   450.00 Сумма:     54000.00
Товар: Курица охлажденная       Кол-во:  180 Цена:   320.00 Сумма:     57600.00
Товар: Молоко 2,5%              Кол-во:  180 Цена:    82.00 Сумма:     14760.00
Товар: Рис круглый              Кол-во:  300 Цена:    95.00 Сумма:     28500.00

2. ОТСУТСТВУЮЩИЕ ТОВАРЫ ДЛЯ ЗАКАЗА МАГАЗИНОМ 'Супермаркет "Восток"' (ID=1):
------------------------------------------------------------
Товар: Молоко 2,5%              База: Центральная база "Северная"    Доступно: 200 Цена:    85.50 Приоритет: 1
Товар: Молоко 2,5%              База: Южный распределительный центр  Доступно: 180 Цена:    82.00 Приоритет: 2

2.доп ОТСУТСТВУЮЩИЕ ТОВАРЫ ДЛЯ ЗАКАЗА МАГАЗИНОМ (включая новые товары с приоритетами) 'Супермаркет "Восток"' (ID=1):
------------------------------------------------------------
Товар: Картофель                Статус: НОВЫЙ ТОВАР (может быть добавлен) База: Центральная база "Северная" Приоритет: 1
Товар: Молоко 2,5%              Статус: ЗАКОНЧИЛСЯ                 База: Центральная база "Северная" Приоритет: 1
Товар: Молоко 2,5%              Статус: ЗАКОНЧИЛСЯ                 База: Южный распределительный центр Приоритет: 2
Товар: Шоколад горький          Статус: НОВЫЙ ТОВАР (может быть добавлен) База: Западный складской комплекс Приоритет: 3

3. ТОВАРЫ В ОТДЕЛЕ 'Молочный отдел' (ID=1):
------------------------------------------------------------
Магазин: Супермаркет "Восток"     Отдел: Молочный отдел     Товар: Йогурт натуральный       Кол-во: 30
Магазин: Супермаркет "Восток"     Отдел: Молочный отдел     Товар: Молоко 2,5%              Кол-во: 0
Магазин: Супермаркет "Восток"     Отдел: Молочный отдел     Товар: Сыр Российский           Кол-во: 25
```

Рисунок 1 – Скриншот вывода задач 1-3.

```
4. ЗАВЕДУЮЩИЕ ОТДЕЛАМИ МАГАЗИНА 'Супермаркет "Восток"' (ID=1):
------------------------------------------------------------
Отдел: Молочный отдел       Заведующий: Мария Сидорова
Отдел: Напитки              Заведующий: Дмитрий Орлов
Отдел: Хлебный отдел        Заведующий: Алексей Козлов

5. СУММАРНАЯ СТОИМОСТЬ ТОВАРОВ ПО ОТДЕЛАМ:
------------------------------------------------------------
Магазин: Спецмагазин "Техника"     Отдел: Бытовая техника      Сумма:    224000.00
Магазин: Премиум маркет "Люкс"     Отдел: Кондитерский отдел   Сумма:      7500.00
Магазин: Магазин "Эконом"          Отдел: Бакалея              Сумма:      5250.00
Магазин: Премиум маркет "Люкс"     Отдел: Гастрономия          Сумма:      3750.00
Магазин: Супермаркет "Восток"      Отдел: Напитки              Сумма:      2375.00
Магазин: Супермаркет "Восток"      Отдел: Хлебный отдел        Сумма:      1920.00
Магазин: Супермаркет "Восток"      Отдел: Молочный отдел       Сумма:         0.00

6. ПОИСК ТОВАРА 'Молоко' ПО БАЗАМ:
------------------------------------------------------------
Товар: Молоко 2,5%              База: Центральная база "Северная"        Кол-во:  200 Цена:    85.50
Товар: Молоко 2,5%              База: Южный распределительный центр      Кол-во:  180 Цена:    82.00
```

Рисунок 2 – Скриншот вывода задач 4-6.