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

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Базы данных»
Тема: Реализация базы данных с использованием ORM.

Студент гр. 3384 Копасова К. А.
Преподаватель Михайлова С. А.

Санкт-Петербург
2025

Цель работы: реализовать базу данных с использованием ORM.

Задание:

Пусть требуется создать программную систему, предназначенную для работников справочной службы кинотеатров города. Такая система должна обеспечивать хранение сведений о кинотеатрах города, о фильмах, которые в них демонстрируются, о сеансах и билетах на эти сеансы. Сведения о кинотеатре — это его название, район города, где расположен кинотеатр, категория, вместимость. Сведения о фильме — это название фильма, режиссёр, оператор, актёры, сыгравшие главные роли, жанр; производство, наличие призов кинофестивалей, продолжительность сеанса, кадр из фильма для рекламы. Кроме того, должна храниться информация о репертуаре кинотеатров на месяц, то есть о том какие фильмы, когда и где демонстрируются, о ценах на билеты и о количестве свободных мест на тот или иной сеанс. На разных сеансах в одном кинотеатре могут идти разные фильмы, а если в кинотеатре несколько залов, то и на одном. Кинотеатр может ввести новый фильм в репертуар или убрать фильм из репертуара. Работник справочной службы может корректировать перечень фильмов, находящихся в прокате – добавлять новые фильмы и снимать с проката, а также перечень кинотеатров, поскольку кинотеатры могут открываться или закрываться, причём иногда временно, например, на ремонт. Цена билета определяется прокатной стоимостью копии фильма, сеансом и категорией кинотеатра. Справочной службе могут потребоваться следующие сведения о текущем состоянии проката фильмов в городе:

1. Репертуар кинотеатра?

2. Адрес и район кинотеатра ?

3. Число свободных мест на данный сеанс в указанном кинотеатре?

4. Цена билетов на данный сеанс в указанном кинотеатре?

5. Жанр, производство и режиссёр данного фильма ?

6. Какие фильмы имеют награды, когда и в каких кинотеатрах они демонстрируются?

7. В каких кинотеатрах в указанный день на указанных сеансах демонстрируется комедия?

В данной лабораторной работе рекомендуется использовать Sequelize (Node.js). В случае, если не знаете как с ним работать, то можно воспользоваться примером из репозитория.

Вы можете использовать другой ORM по вашему выбору по согласованию с преподавателем, принимающим у вас практики.

Необходимо развернуть Sequelize на своем ПК и выполнить следующие задачи:

- Описать в виде моделей Sequelize 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)
- Запушить в репозиторий исходный код проекта, соблюсти .gitignore, убрать исходную базу из проекта (или иные нагенерированные данные бд если они есть).
- Описать процесс запуска: команды, зависимости
- В отчете описать цель, текст задания в соответствии с вариантом, выбранную ORM, инструкцию по запуску, скриншоты (код) моделей ORM, скриншоты на каждый запрос (или группу запросов) на изменение/таблицы с выводом результатов (ответ), ссылку на PR в приложении, вывод

Выполнение работы

Для выполнении работы была выбрана ORM SQLAlchemy на языке Python. Перед запуском проета необходимо установить следующие библиотеки:

```
pip install sqlalchemy psycopg2 tabulate
```

Также нужно создать базу данных в PostgreeSQL и убедиться, что строка подключения в файле db_config.py (переменная DATABASE_URL) соответствует созданной базе данных. После можно запустить файл main.py, который продемонстрирует работу всех файлов проекта.

Описание файлов проекта:

1) *db_config.py* - хранит строку подключения к базе данных и создает engine и session для работы с базой.

2) *models.py* - описывает все таблицы базы данных через ORM SQLAlchemy (Base и классы таблиц).

3) *create_table.py* - создает все таблицы из *model.py* в базе данных.

4) *insert_data.py* - заполняет таблицы текстовыми данными.

5) *show_table.py* - выводит содержимое таблиц в табличном виде.

6) *questions.py* - содержит все ORM-запросы по заданию лабораторной работы.

7) *delete_table.py* - позволяет удалить таблицу по имени или сразу все таблицы.

8) *main.py* - основной скрипт, который вызывает функции из *delete_table.py*, *create_table.py*, *insert_data.py*, *show_table.py* и *questions.py* для демонстрации работы проета.

Продемонстрируем работу проекта. Создадим все таблицы и заполним их данными. Результат создания таблиц представлен на рис. 1, результат вставки данных в таблицы представлен на рис. 2. Таблица cinema представлена на рис. 3.

Таблица films представлена на рис. 4. Таблица festival представлена на рис. 5. Таблица cinema_halls представлена на рис. 6. Таблица sessions представлена на рис. 7. Таблица tickets представлена на рис. 8. Таблица prizes представлена на рис. 9.

```
таблица 'cinema' создана (0 строк)
таблица 'films' создана (0 строк)
таблица 'festival' создана (0 строк)
таблица 'cinema_halls' создана (0 строк)
таблица 'sessions' создана (0 строк)
таблица 'tickets' создана (0 строк)
таблица 'prizes' создана (0 строк)
все таблицы созданы
```

Рисунок 1 - Создание таблиц.

```
таблица Cinema заполнена
таблица Films заполнена
таблица Festival заполнена
таблица CinemaHalls заполнена
таблица Sessions заполнена
таблица Tickets заполнена
таблица Prizes заполнена
все данные успешно вставлены в базу
```

Рисунок 2 - Вставка данных в таблицы.

таблица: cinema

cinema_id	title	city_region	address_cinema	category	total_capacity
1	Аврора	Центральный	Невский проспект, 60	Премиум	600
2	Формула Кино Галерея	Центральный	Лиговский проспект, 30	Мультиплекс	800
3	Каро 11 Охта Молл	Красногвардейский	Брантовская дорога, 3	Мультиплекс	1200
4	7D	Адмиралтейский	ул. Марата, 86	Классический	280
5	Ленфильм	Петроградский	Каменноостровский проспект, 10	Премиум	450
6	Дом кино	Центральный	ул. Караванная, 12	Исторический	200
7	Синема парк Питер Радуга	Московский	проспект Космонавтов, 14	Стандарт	550
8	Формула Кино Сити Молл	Приморский	Коломяжский проспект, 17	Стандарт	320

Рисунок 3 - Таблица cinema.

таблица: films

film_id	title	director	operator	main_actors	genre	production	session_duration	shot_advertising
1	Аватар: Путь воды	Джеймс Кэмерон	Расселл Карпентер	Сэм Уортингтон, Зои Салдана, Стивен Лэнг, Кейт Уинслет	Фантастика	США	192	avatar_water_poster.jpg
2	Оппенгеймер	Кристофер Нолан	Хойте ван Хойтема	Киллиан Мерфи, Эмили Блант, Мэтт Дэймон, Роберт Дауни-мл.	Исторический	США	180	oppenheimer_poster.jpg
3	Дюна: Часть вторая	Дени Вильнёв	Грег Фрейзер	Тимоти Шаламе, Зендея, Ребекка Фергюсон, Остин Батлер	Фантастика	США	166	dune2_poster.jpg
4	Барби	Грета Гервиг	Родриго Прието	Марго Робби, Райан Гослинг, Америка Феррера, Кейт МакКиннон	Комедия	США	114	barbie_poster.jpg
5	Вызов	Клим Шипенко	Клим Шипенко	Юлия Пересильд, Владимир Машков, Михаил Трофимов	Драма	Россия	165	challenge_poster.jpg
6	Человек-паук: Паутина вселенных	Жуакин Душ Сантуш	Джастин К. Томпсон	Шмеик Мур, Хейли Стайнфелд, Оскар Айзек	Мультфильм	США	140	spiderverse_poster.jpg
7	Сергий против нечисти	Святослав Подгаевский	Святослав Подгаевский	Тихон Жизневский, Алексей Розин, Софья Райзман	Ужасы	Россия	112	sergiy_poster.jpg
8	Гарри Поттер и Дары Смерти: Часть 2	Дэвид Йейтс	Эдуарду Серра	Дэниел Рэдклифф, Эмма Уотсон, Руперт Гринт, Рэйф Файнс	Фэнтези	Великобритания	130	hp_deathly_hallows.jpg

Рисунок 4 - Таблица films.

таблица: festival

festival_id	title
1	Каннский кинофестиваль
2	Оскар
3	Венецианский кинофестиваль
4	Берлинале
5	Кинотавр
6	Московский международный кинофестиваль
7	Сандэнс
8	Золотой орёл

Рисунок 5 - Таблица festival.

таблица: cinema_halls

hall_id	cinema_id	title	capacity
1	1	Зал 1	125
2	1	Зал 2	75
3	2	Красный зал	200
4	2	Синий зал	100
5	2	Зеленый зал	35
6	3	Основной зал	150
7	4	Зал VIP	15
8	4	Зал MEDIUM	75

Рисунок 6 - Таблица cinema_halls.

таблица: sessions

session_id	hall_id	film_id	date_session	start_time	end_time
1	1	1	2025-10-20	10:00:00	13:15:00
2	1	2	2025-10-22	14:00:00	16:28:00
3	1	7	2025-10-22	17:00:00	19:15:00
4	3	3	2025-10-20	11:30:00	14:00:00
5	4	6	2025-10-27	11:30:00	14:45:00
6	5	4	2025-10-21	15:00:00	17:35:00
7	5	4	2025-10-21	18:15:00	20:50:00
8	6	5	2025-10-25	12:00:00	13:40:00

Рисунок 7 - Таблица sessions.

таблица: tickets

ticket_id	session_id	row_number	place_number	sold_out	cost
1	1	1	5	0	350
2	1	1	6	1	270
3	2	2	3	0	400
4	3	3	15	1	250
5	4	3	9	0	370
6	5	2	13	0	250
7	6	1	12	0	500
8	7	1	7	0	1000

Рисунок 8 - Таблица tickets.

таблица: prizes

prizes_id	festival_id	film_id	title
1	2	1	Лучшие визуальные эффекты
2	1	4	Золотая пальмовая ветвь
3	3	3	Лучшая режиссура
4	2	2	Лучший актер
5	4	5	Приз зрительских симпатий
6	5	6	Главный приз
7	6	7	Лучший фильм
8	2	8	Лучшая операторская работа

Рисунок 9 - Таблица prizes.

Выполним запросы из задания. Ответ на запрос 1 представлен на рис. 10. Ответ на запрос 2 представлен на рис. 11. Ответ на запрос 3 представлен на рис. 12. Ответ на запрос 4 представлен на рис. 13. Ответ на запрос 5 представлен на рис. 14. Ответ на запрос 6 представлен на рис. 15. Ответ на запрос 7 представлен на рис. 16.

1. Репертуар кинотеатра? (кинотеатр Аврора)

Название фильма	Жанр	Режиссер	Продолжительность (мин)
Аватар: Путь воды	Фантастика	Джеймс Кэмерон	192
Оппенгеймер	Исторический	Кристофер Нолан	180
Сергий против нечисти	Ужасы	Святослав Подгаевский	112

Рисунок 10 - Репертуар кинотеатра Аврора.

2. Адрес и район кинотеатра? (кинотеатр Аврора)

Кинотеатр	Район	Адрес
Аврора	Центральный	Невский проспект, 60

Рисунок 11 - Адрес и район кинотеатра Аврора.

3. Число свободных мест на данный сеанс в указанном кинотеатре? (кинотеатр Аврора, сеанс 1)

Свободные места
1

Рисунок 12 - Число свободных мест на сеанс 1 кинотеатра Аврора.

4. Цена билетов на данный сеанс в указанном кинотеатре? (кинотеатр Аврора, сеанс 1)

Цена билета
270
350

Рисунок 13 - Цена билетов на сеанс 1 кинотеатра Аврора.

5. Жанр, производство и режиссер данного фильма? (Барби)

Фильм	Жанр	Производство	Режиссер
Барби	Комедия	США	Грета Гервиг

Рисунок 14 - Жанр, производство и режиссер фильма Барби.

6. Какие фильмы имеют награды, когда и в каких кинотеатрах они демонстрируются?

Фильм	Награда	Фестиваль	Дата сеанса	Время начала	Кинотеатр	Зал	Адрес
Аватар: Путь воды	Лучшие визуальные эффекты	Оскар	2025-10-20	10:00:00	Аврора	Зал 1	Невский проспект, 60
Барби	Золотая пальмовая ветвь	Каннский кинофестиваль	2025-10-21	15:00:00	Формула Кино Галерея	Зеленый зал	Лиговский проспект, 30
Барби	Золотая пальмовая ветвь	Каннский кинофестиваль	2025-10-21	18:15:00	Формула Кино Галерея	Зеленый зал	Лиговский проспект, 30
Вызов	Приз зрительских симпатий	Берлинале	2025-10-25	12:00:00	Каро 11 Охта Молл	Основной зал	Брантовская дорога, 3
Дюна: Часть вторая	Лучшая режиссура	Венецианский кинофестиваль	2025-10-20	11:30:00	Формула Кино Галерея	Красный зал	Лиговский проспект, 30
Оппенгеймер	Лучший актер	Оскар	2025-10-22	14:00:00	Аврора	Зал 1	Невский проспект, 60
Сергий против нечисти	Лучший фильм	Московский международный кинофестиваль	2025-10-22	17:00:00	Аврора	Зал 1	Невский проспект, 60
Человек-паук: Паутина вселенных	Главный приз	Кинотавр	2025-10-27	11:30:00	Формула Кино Галерея	Синий зал	Лиговский проспект, 30

Рисунок 15 - Награды фильмов, когда и в каких кинотеатрах они демонстрируются.

7. В каких кинотеатрах в указанный день на указанных сеансах демонстрируется комедия? (день 2025-10-21, сеансы 1, 2, 3, 6 и 7)

Кинотеатр	Фильм	Жанр	Сеанс	Зал	Дата	Начало	Окончание
Формула Кино Галерея	Барби	Комедия	6	Зеленый зал	2025-10-21	15:00:00	17:35:00
Формула Кино Галерея	Барби	Комедия	7	Зеленый зал	2025-10-21	18:15:00	20:50:00

Рисунок 16 - Кинотеатры, в которых 2025-10-21 на сеансах 1, 2, 3, 6, 7 демонстрируется комедия.

Полученные результаты запросов совпадают с результатами, полученными в прошлой лабораторной работе при написании SQL запросов. Весь написанный код и ссылка на pull request приведены в Приложении 1.

Вывод

В ходе лабораторной работы была изучена ORM SQLAlchemy и реализована база данных кинотеатров на основе данных из предыдущей лабораторной работы.

Были созданы таблицы с соблюдением связей между ними, написан скрипт заполнения таблиц тестовыми данными. Также были реализованы ORM-запросы, отвечающие на вопросы из первой лабораторной работы, с выводом результатов в читаемом табличном виде. Все функции и скрипты проверены на корректность работы.

ПРИЛОЖЕНИЕ 1

Ссылка на pull request: https://github.com/moevm/sql-2025-3384/pull/33.

Файл *db_config.py*:

```
from sqlalchemy import create_engine
from sqlalchemy.orm import sessionmaker

DATABASE_URL = "postgresql+psycopg2://postgres:000000@localhost:5432/lb3"

engine = create_engine(DATABASE_URL)

Session = sessionmaker(bind=engine)
session = Session()
```

Файл *models.py*:

```
from sqlalchemy import Column, Integer, String, Date, Time, Text, ForeignKey, CheckConstraint
from sqlalchemy.orm import declarative_base, relationship


Base = declarative_base()


class Cinema(Base):
    __tablename__ = 'cinema'


    cinema_id = Column(Integer, primary_key=True, autoincrement=True)
    title = Column(String(100), nullable=False)
    city_region = Column(String(100), nullable=False)
    address_cinema = Column(String(100), nullable=False)
    category = Column(String(30))
    total_capacity = Column(Integer, nullable=False)


    __table_args__ = (CheckConstraint('total_capacity > 0'),)


    halls = relationship("CinemaHalls", back_populates="cinema", cascade="all, delete")


class Films(Base):
    __tablename__ = 'films'


    film_id = Column(Integer, primary_key=True, autoincrement=True)
    title = Column(String(100), nullable=False)
    director = Column(String(100), nullable=False)
```

```
    operator = Column(String(100))
    main_actors = Column(Text)
    genre = Column(String(30))
    production = Column(String(100))
    session_duration = Column(Integer, nullable=False)
    shot_advertising = Column(String(255))

    __table_args__ = (CheckConstraint('session_duration > 0'),)

    sessions = relationship("Sessions", back_populates="film", cascade="all, delete")
    prizes = relationship("Prizes", back_populates="film", cascade="all, delete")

class Festival(Base):
    __tablename__ = 'festival'

    festival_id = Column(Integer, primary_key=True, autoincrement=True)
    title = Column(String(100), nullable=False)

    prizes = relationship("Prizes", back_populates="festival", cascade="all, delete")

class CinemaHalls(Base):
    __tablename__ = 'cinema_halls'

    hall_id = Column(Integer, primary_key=True, autoincrement=True)
    cinema_id = Column(Integer, ForeignKey('cinema.cinema_id', ondelete='CASCADE'), nullable=False)
    title = Column(String(100), nullable=False)
    capacity = Column(Integer, nullable=False)

    __table_args__ = (CheckConstraint('capacity > 0'),)

    cinema = relationship("Cinema", back_populates="halls")
    sessions = relationship("Sessions", back_populates="hall", cascade="all, delete")

class Sessions(Base):
    __tablename__ = 'sessions'
```

```
    session_id = Column(Integer, primary_key=True, autoincrement=True)

    hall_id = Column(Integer, ForeignKey('cinema_halls.hall_id',
ondelete='CASCADE'), nullable=False)

    film_id = Column(Integer, ForeignKey('films.film_id', ondelete='CASCADE'),
nullable=False)

    date_session = Column(Date, nullable=False)

    start_time = Column(Time, nullable=False)

    end_time = Column(Time, nullable=False)


    hall = relationship("CinemaHalls", back_populates="sessions")

    film = relationship("Films", back_populates="sessions")

    tickets = relationship("Tickets", back_populates="session", cascade="all,
delete")


class Tickets(Base):

    __tablename__ = 'tickets'


    ticket_id = Column(Integer, primary_key=True, autoincrement=True)

    session_id = Column(Integer, ForeignKey('sessions.session_id',
ondelete='CASCADE'), nullable=False)

    row_number = Column(Integer, nullable=False)

    place_number = Column(Integer, nullable=False)

    sold_out = Column(String(1), default='0')

    cost = Column(Integer, nullable=False)


    __table_args__ = (
        CheckConstraint('row_number > 0'),
        CheckConstraint('place_number > 0'),
        CheckConstraint('cost >= 0'),
    )


    session = relationship("Sessions", back_populates="tickets")


class Prizes(Base):

    __tablename__ = 'prizes'


    prizes_id = Column(Integer, primary_key=True, autoincrement=True)

    festival_id = Column(Integer, ForeignKey('festival.festival_id',
ondelete='CASCADE'), nullable=False)
```

```
    film_id = Column(Integer, ForeignKey('films.film_id', ondelete='CASCADE'),
nullable=False)
    title = Column(String(100), nullable=False)


    festival = relationship("Festival", back_populates="prizes")
    film = relationship("Films", back_populates="prizes")
```

Файл *create_table.py*:

```
from sqlalchemy import text
from models import Base
from db_config import *

def create_db():
    Base.metadata.create_all(engine)

    with engine.connect() as conn:
        for table_name in Base.metadata.tables.keys():
            result = conn.execute(text(f"SELECT COUNT(*) FROM {table_name};"))
            count = result.scalar()
            print(f"таблица '{table_name}' создана ({count} строк)")

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

Файл *insert_data.py*:

```
from models import *
from db_config import *

def insert_cinemas():
    # Добавление кинотеатров
    cinemas = [
        Cinema(title='Аврора',                    city_region='Центральный',
address_cinema='Невский проспект, 60', category='Премиум', total_capacity=600),
        Cinema(title='Формула Кино Галерея', city_region='Центральный',
address_cinema='Лиговский проспект, 30', category='Мультиплекс',
total_capacity=800),
        Cinema(title='Каро 11 Охта Молл', city_region='Красногвардейский',
address_cinema='Брантовская дорога, 3', category='Мультиплекс',
total_capacity=1200),
```

```
        Cinema(title='7D', city_region='Адмиралтейский', address_cinema='ул. Марата, 86', category='Классический', total_capacity=280),

        Cinema(title='Ленфильм', city_region='Петроградский', address_cinema='Каменноостровский проспект, 10', category='Премиум', total_capacity=450),

        Cinema(title='Дом кино', city_region='Центральный', address_cinema='ул. Караванная, 12', category='Исторический', total_capacity=200),

        Cinema(title='Синема парк Питер Радуга', city_region='Московский', address_cinema='проспект Космонавтов, 14', category='Стандарт', total_capacity=550),

        Cinema(title='Формула Кино Сити Молл', city_region='Приморский', address_cinema='Коломяжский проспект, 17', category='Стандарт', total_capacity=320),

    ]

    session.add_all(cinemas)

    session.commit()

    print("таблица Cinema заполнена")


def insert_films():

    films = [

        Films(title='Аватар: Путь воды', director='Джеймс Кэмерон', operator='Расселл Карпентер', main_actors='Сэм Уортингтон, Зои Салдана, Стивен Лэнг, Кейт Уинслет', genre='Фантастика', production='США', session_duration=192, shot_advertising='avatar_water_poster.jpg'),

        Films(title='Оппенгеймер', director='Кристофер Нолан', operator='Хойте ван Хойтема', main_actors='Киллиан Мерфи, Эмили Блант, Мэтт Дэймон, Роберт Дауни-мл.', genre='Исторический', production='США', session_duration=180, shot_advertising='oppenheimer_poster.jpg'),

        Films(title='Дюна: Часть вторая', director='Дени Вильнёв', operator='Грег Фрейзер', main_actors='Тимоти Шаламе, Зендея, Ребекка Фергюсон, Остин Батлер', genre='Фантастика', production='США', session_duration=166, shot_advertising='dune2_poster.jpg'),

        Films(title='Барби', director='Грета Гервиг', operator='Родриго Прието', main_actors='Марго Робби, Райан Гослинг, Америка Феррера, Кейт МакКиннон', genre='Комедия', production='США', session_duration=114, shot_advertising='barbie_poster.jpg'),

        Films(title='Вызов', director='Клим Шипенко', operator='Клим Шипенко', main_actors='Юлия Пересильд, Владимир Машков, Михаил Трофимов', genre='Драма', production='Россия', session_duration=165, shot_advertising='challenge_poster.jpg'),

        Films(title='Человек-паук: Паутина вселенных', director='Жуакин Душ Сантуш', operator='Джастин К. Томпсон', main_actors='Шамеик Мур, Хейли Стайнфелд, Оскар Айзек', genre='Мультфильм', production='США', session_duration=140, shot_advertising='spiderverse_poster.jpg'),

        Films(title='Сергий против нечисти', director='Святослав Подгаевский', operator='Святослав Подгаевский', main_actors='Тихон Жизневский, Алексей Розин, Софья Райзман', genre='Ужасы', production='Россия', session_duration=112, shot_advertising='sergiy_poster.jpg'),

        Films(title='Гарри Поттер и Дары Смерти: Часть 2', director='Дэвид Йейтс', operator='Эдуарду Серра', main_actors='Дэниел Рэдклифф, Эмма Уотсон,
```

```
Руперт Гринт, Рэйф Файнс', genre='Фэнтези', production='Великобритания',
session_duration=130, shot_advertising='hp_deathly_hallows.jpg')
        ]
        session.add_all(films)
        session.commit()
        print("таблица Films заполнена")


    def insert_festivals():
        festivals = [
            Festival(title='Каннский кинофестиваль'),
            Festival(title='Оскар'),
            Festival(title='Венецианский кинофестиваль'),
            Festival(title='Берлинале'),
            Festival(title='Кинотавр'),
            Festival(title='Московский международный кинофестиваль'),
            Festival(title='Сандэнс'),
            Festival(title='Золотой орёл')
        ]
        session.add_all(festivals)
        session.commit()
        print("таблица Festival заполнена")


    def insert_cinema_halls():
        halls = [
            CinemaHalls(cinema_id=1, title='Зал 1', capacity=125),
            CinemaHalls(cinema_id=1, title='Зал 2', capacity=75),
            CinemaHalls(cinema_id=2, title='Красный зал', capacity=200),
            CinemaHalls(cinema_id=2, title='Синий зал', capacity=100),
            CinemaHalls(cinema_id=2, title='Зеленый зал', capacity=35),
            CinemaHalls(cinema_id=3, title='Основной зал', capacity=150),
            CinemaHalls(cinema_id=4, title='Зал VIP', capacity=15),
            CinemaHalls(cinema_id=4, title='Зал MEDIUM', capacity=75)
        ]
        session.add_all(halls)
        session.commit()
        print("таблица CinemaHalls заполнена")


    def insert_sessions():
```

```
      sessions_list = [
          Sessions(hall_id=1,        film_id=1,        date_session='2025-10-20',
start_time='10:00', end_time='13:15'),
          Sessions(hall_id=1,        film_id=2,        date_session='2025-10-22',
start_time='14:00', end_time='16:28'),
          Sessions(hall_id=1,        film_id=7,        date_session='2025-10-22',
start_time='17:00', end_time='19:15'),
          Sessions(hall_id=3,        film_id=3,        date_session='2025-10-20',
start_time='11:30', end_time='14:00'),
          Sessions(hall_id=4,        film_id=6,        date_session='2025-10-27',
start_time='11:30', end_time='14:45'),
          Sessions(hall_id=5,        film_id=4,        date_session='2025-10-21',
start_time='15:00', end_time='17:35'),
          Sessions(hall_id=5,        film_id=4,        date_session='2025-10-21',
start_time='18:15', end_time='20:50'),
          Sessions(hall_id=6,        film_id=5,        date_session='2025-10-25',
start_time='12:00', end_time='13:40')
      ]
      session.add_all(sessions_list)
      session.commit()
      print("таблица Sessions заполнена")


  def insert_tickets():
      tickets_list = [
          Tickets(session_id=1,   row_number=1,   place_number=5,   sold_out='0',
cost=350),
          Tickets(session_id=1,   row_number=1,   place_number=6,   sold_out='1',
cost=270),
          Tickets(session_id=2,   row_number=2,   place_number=3,   sold_out='0',
cost=400),
          Tickets(session_id=3,   row_number=3,   place_number=15,   sold_out='1',
cost=250),
          Tickets(session_id=4,   row_number=3,   place_number=9,   sold_out='0',
cost=370),
          Tickets(session_id=5,   row_number=2,   place_number=13,   sold_out='0',
cost=250),
          Tickets(session_id=6,   row_number=1,   place_number=12,   sold_out='0',
cost=500),
          Tickets(session_id=7,   row_number=1,   place_number=7,   sold_out='0',
cost=1000)
      ]
      session.add_all(tickets_list)
      session.commit()
      print("таблица Tickets заполнена")
```

```
def insert_prizes():
    prizes_list = [
        Prizes(festival_id=2, film_id=1, title='Лучшие визуальные эффекты'),
        Prizes(festival_id=1, film_id=4, title='Золотая пальмовая ветвь'),
        Prizes(festival_id=3, film_id=3, title='Лучшая режиссура'),
        Prizes(festival_id=2, film_id=2, title='Лучший актер'),
        Prizes(festival_id=4, film_id=5, title='Приз зрительских симпатий'),
        Prizes(festival_id=5, film_id=6, title='Главный приз'),
        Prizes(festival_id=6, film_id=7, title='Лучший фильм'),
        Prizes(festival_id=2, film_id=8, title='Лучшая операторская работа')
    ]
    session.add_all(prizes_list)
    session.commit()
    print("таблица Prizes заполнена")

if __name__ == "__main__":
    insert_cinemas()
    insert_films()
    insert_festivals()
    insert_cinema_halls()
    insert_sessions()
    insert_tickets()
    insert_prizes()
    print("все данные успешно вставлены в базу lb3")
```

Файл *show_table.py:*

```
from sqlalchemy import inspect, text, MetaData
from tabulate import tabulate
from models import *
from db_config import *

metadata = MetaData()

def show_table(table_name):
    inspector = inspect(engine)
    tables = inspector.get_table_names()
```

```
    table_name = table_name.lower()

    if table_name not in tables:
        print(f"таблица '{table_name}' не найдена")
        return

    query = text(f"SELECT * FROM {table_name}")
    result = session.execute(query).fetchall()
    columns = [col["name"] for col in inspector.get_columns(table_name)]

    print(f"таблица: {table_name}")
    print(tabulate(result, headers=columns, tablefmt="fancy_grid",
disable_numparse=True))


def show_all_table():
    show_table("cinema")
    show_table("films")
    show_table("festival")
    show_table("cinema_halls")
    show_table("sessions")
    show_table("tickets")
    show_table("prizes")
```

Файл *questions.py:*

```
from sqlalchemy import func, distinct
from sqlalchemy.orm import sessionmaker
from tabulate import tabulate
from models import *
from db_config import *

Session = sessionmaker(bind=engine)
session = Session()

def repertoire(cinema_name):
    result = (session.query(distinct(Films.title), Films.genre, Films.director,
Films.session_duration)
              .join(Sessions, Films.film_id == Sessions.film_id)
```

```
                .join(CinemaHalls, Sessions.hall_id == CinemaHalls.hall_id)
                .join(Cinema, CinemaHalls.cinema_id == Cinema.cinema_id)
                .filter(Cinema.title == cinema_name)
                .order_by(Films.title)
                .all())

    headers = ["Название фильма", "Жанр", "Режиссер", "Продолжительность (мин)"]
    return tabulate(result, headers=headers, tablefmt="fancy_grid")

def cinema_address(cinema_name):
    result = session.query(Cinema.title, Cinema.city_region, Cinema.address_cinema)\
        .filter(Cinema.title == cinema_name).first()

    headers = ["Кинотеатр", "Район", "Адрес"]
    return tabulate([result], headers=headers, tablefmt="fancy_grid")

def free_places(cinema_name, session_id):
    count = (session.query(func.count(Tickets.ticket_id))
             .join(Sessions)
             .join(CinemaHalls)
             .join(Cinema)
             .filter(Cinema.title == cinema_name, Sessions.session_id == session_id, Tickets.sold_out == '0')
             .scalar())
    headers = ["Свободные места"]
    return tabulate([[count]], headers=headers, tablefmt="fancy_grid")

def ticket_prices(cinema_name, session_id):
    result = (session.query(distinct(Tickets.cost))
              .join(Sessions, Tickets.session_id == Sessions.session_id)
              .join(CinemaHalls, Sessions.hall_id == CinemaHalls.hall_id)
              .join(Cinema, CinemaHalls.cinema_id == Cinema.cinema_id)
              .filter(Cinema.title == cinema_name, Sessions.session_id == session_id)
              .order_by(Tickets.cost)
              .all())
```

```
        headers = ["Цена билета"]
        return tabulate(result, headers=headers, tablefmt="fancy_grid")


    def film_info(film_title):
        result = session.query(Films.title, Films.genre, Films.production,
Films.director)\
            .filter(Films.title == film_title).first()


        headers = ["Фильм", "Жанр", "Производство", "Режиссер"]
        return tabulate([result], headers=headers, tablefmt="fancy_grid")


    def films_with_prizes():
        result = (session.query(Films.title, Prizes.title.label("Награда"),
Festival.title.label("Фестиваль"),
                                Sessions.date_session, Sessions.start_time,
                                Cinema.title.label("Кинотеатр"),
CinemaHalls.title.label("Зал"),
                                Cinema.address_cinema)
                  .join(Prizes, Films.film_id == Prizes.film_id)
                  .join(Festival, Prizes.festival_id == Festival.festival_id)
                  .join(Sessions, Films.film_id == Sessions.film_id)
                  .join(CinemaHalls, Sessions.hall_id == CinemaHalls.hall_id)
                  .join(Cinema, CinemaHalls.cinema_id == Cinema.cinema_id)
                  .order_by(Films.title, Sessions.date_session,
Sessions.start_time)
                  .all())


        headers = ["Фильм", "Награда", "Фестиваль", "Дата сеанса", "Время начала",
"Кинотеатр", "Зал", "Адрес"]
        return tabulate(result, headers=headers, tablefmt="fancy_grid")


    def comedy_on_day(date, session_ids):
        result = (session.query(Cinema.title.label("Кинотеатр"),
Films.title.label("Фильм"), Films.genre,
                                Sessions.session_id,
CinemaHalls.title.label("Зал"),
                                Sessions.date_session, Sessions.start_time,
Sessions.end_time)
                  .join(Films, Sessions.film_id == Films.film_id)
                  .join(CinemaHalls, Sessions.hall_id == CinemaHalls.hall_id)
                  .join(Cinema, CinemaHalls.cinema_id == Cinema.cinema_id)
```

```
                .filter(Films.genre == 'Комедия', Sessions.date_session == date,
                        Sessions.session_id.in_(session_ids))
                .all())

    headers = ["Кинотеатр", "Фильм", "Жанр", "Сеанс", "Зал", "Дата", "Начало",
"Окончание"]
    return tabulate(result, headers=headers, tablefmt="fancy_grid")
```

Файл *delete_table.py:*

```
from sqlalchemy import inspect, text, MetaData
from db_config import *

metadata = MetaData()

def delete_table(table_name):
    inspector = inspect(engine)
    tables = inspector.get_table_names()

    table_name = table_name.lower()

    if table_name not in tables:
        print(f"таблица '{table_name}' не найдена")
        return

    try:
        session.execute(text(f'DROP TABLE IF EXISTS "{table_name}" CASCADE'))
        session.commit()
        print(f"таблица '{table_name}' успешно удалена")
    except Exception as e:
        session.rollback()
        print(f"ошибка при удалении таблицы '{table_name}': {e}")

def delete_all_tables():
    # порядок важен, чтобы сначала удалить зависимые таблицы
    tables = [
        "tickets",
        "sessions",
        "cinema_halls",
```

```
        "prizes",
        "festival",
        "films",
        "cinema"
    ]

    for t in tables:
        delete_table(t)
```

Файл *main.py*

```
from create_table import *
from insert_data import *
from questions import *
from show_table import *
from delete_table import *

def main():
    # удаляем таблицы
    delete_all_tables()

    # создаём таблицы
    create_db()
    print("все таблицы созданы")

    # # # показываем таблицы
    # show_all_table()

    # вставляем данные
    insert_cinemas()
    insert_films()
    insert_festivals()
    insert_cinema_halls()
    insert_sessions()
    insert_tickets()
    insert_prizes()
    print("все данные успешно вставлены в базу")

    # показываем таблицы
    show_all_table()

    # примеры запросов
    print("\n1. Репертуар кинотеатра? (кинотеатр Аврора)")
    print(repertoire("Аврора"))

    print("\n2. Адрес и район кинотеатра? (кинотеатр Аврора)")
    print(cinema_address("Аврора"))

    print("\n3. Число свободных мест на данный сеанс в указанном кинотеатре?
(кинотеатр Аврора, сеанс 1)")
    print(free_places("Аврора", 1))

    print("\n4. Цена билетов на данный сеанс в указанном кинотеатре?
(кинотеатр Аврора, сеанс 1)")
    print(ticket_prices("Аврора", 1))

    print("\n5. Жанр, производство и режиссер данного фильма? (Барби)")
```

```
        print(film_info("Барби"))

        print("\n6. Какие фильмы имеют награды, когда и в каких кинотеатрах они демонстрируются?")
        print(films_with_prizes())

        print("\n7.  В каких кинотеатрах в указанный день на указанных сеансах демонстрируется комедия? (день 2025-10-21, время с 10:00 до 20:00)")
        print(comedy_on_day("2025-10-21", [1,2,3,6,7]))

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