ГУАП

КАФЕДРА № 41

ОТЧЕТ
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

Старший преподаватель		Б.К. Акопян
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

СВЯЗЬ СУБД MYSQL И PYTHON. ВИЗУАЛИЗАЦИЯ ДАННЫХ PYTHON

по курсу: БАЗЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4217		Д.М. Никитин
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2025

1. **Цель работы:** произвести связь базы данных в MySQL и Python, извлечь данные из таблиц базы данных и выполнить анализ данных в БД с помощью визуализации в Python.

2. **Вариант:** 24 – «Информационная система железнодорожной пассажирской станции».

Работников железнодорожной станции можно подразделить на водителей подвижного состава, диспетчеров, ремонтников подвижного состава, путей, кассиров, работников службы подготовки составов, справочной службы и других, которые административно относятся каждый к своему отделу. Каждая из перечисленных категорий работников имеет уникальные атрибуты-характеристики, определяемые профессиональной направленностью. В отделах существует разбиение работников на бригады. Отделы возглавляются начальниками, которые представляют собой администрацию железнодорожной станции. В функции администрации входит планирование маршрутов, составление расписаний, формирование кадрового состава железнодорожной станции. За каждым локомотивом закрепляется локомотивная бригада. За несколькими локомотивами закрепляется бригада техников-ремонтников, выполняющая рейсовый и плановый техосмотр (по определенному графику), ремонт, техническое обслуживание. Водители локомотивов обязаны проходить каждый год медосмотр, не прошедших медосмотр необходимо перевести на другую работу. Локомотив должен своевременно осматриваться техниками-ремонтниками и при необходимости ремонтироваться. Подготовка к рейсу включает в себя техническую часть (рейсовый техосмотр, мелкий ремонт) и обслуживающую часть (уборка вагонов, запас продуктов питания и т.п.).

В расписании указывается тип поезда (скорый, пассажирский . . .), номер поезда, дни и время отправления и прибытия, маршрут (начальный и конечный пункты назначения, основные узловые станции), стоимость билета. Билеты на поезд можно приобрести заранее или забронировать в железнодорожных кассах. До отправления поезда, если есть необходимость, билет можно

вернуть. Отправление поездов может быть задержано из-за опозданий поездов, погодных условий, технических неполадок.

Железнодорожные маршруты можно разделить на следующие категории: внутренние, международные, туристические, специальные маршруты. Пассажиры могут сдавать свои вещи в багажное отделение.

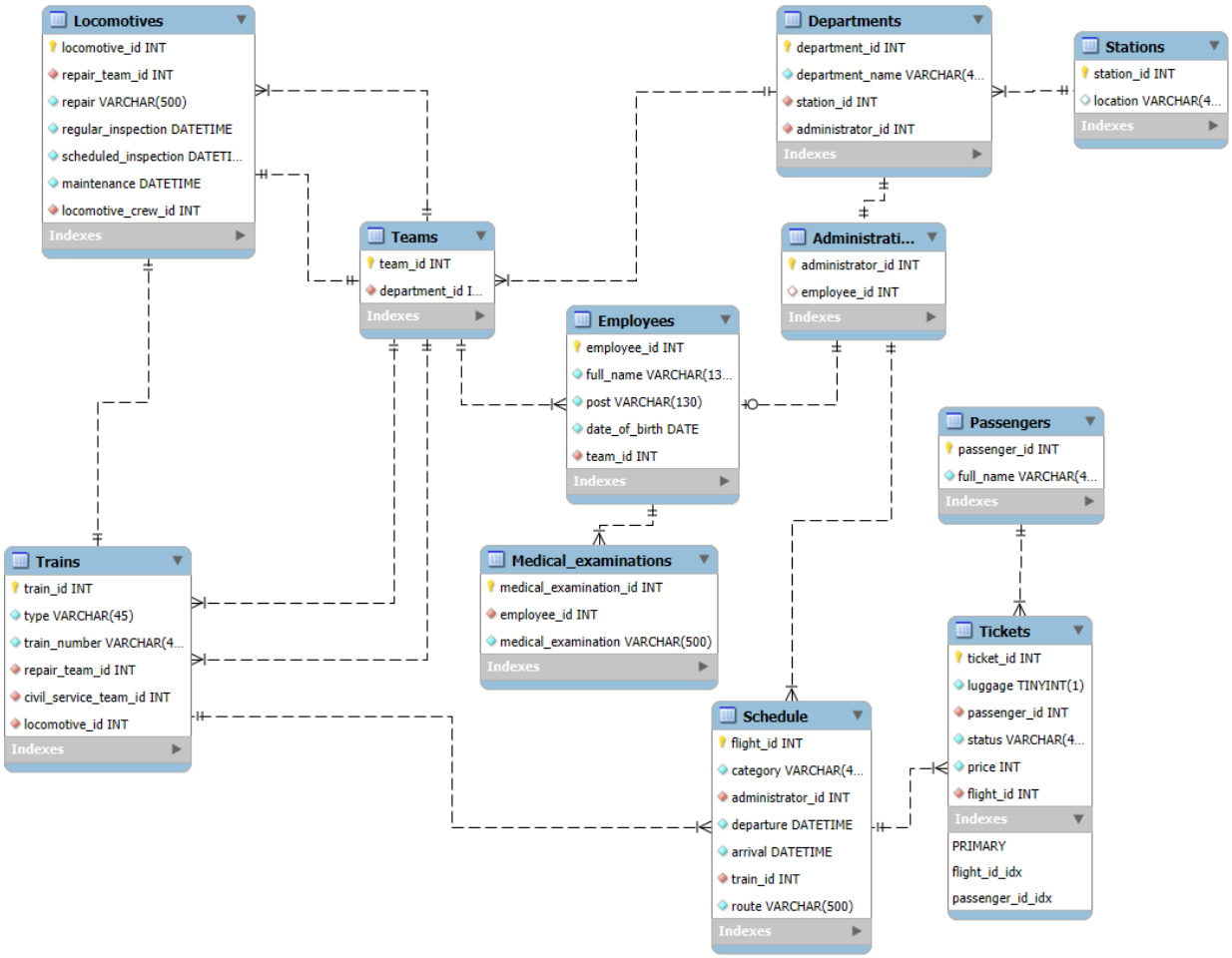


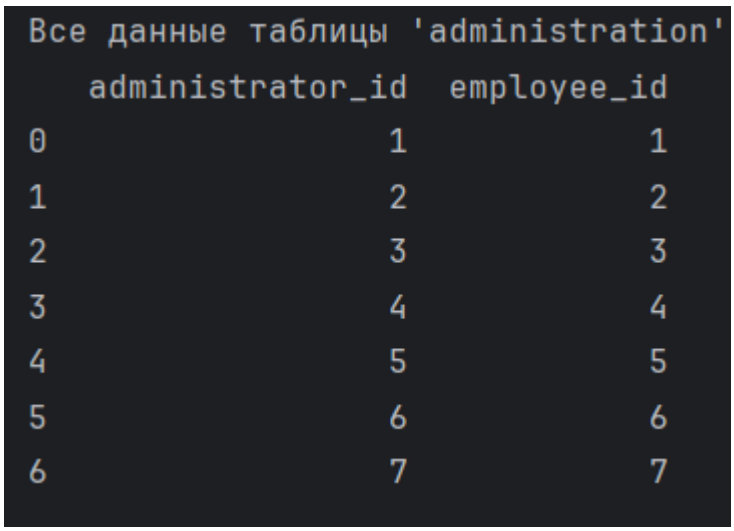
Рисунок 1 – Схема базы данных

3. Ход работы:

В процессе работы было выполнено 4 запроса к базе данных.

Сначала был выполнен общий запрос, который выводит все данные из таблицы administration. Код и результат запроса показан ниже.

```
with connection.cursor() as cursor:
    query: str = "SELECT * FROM `administration`;"
    cursor.execute(query)
    rows = cursor.fetchall()
    df: pd.DataFrame = pd.DataFrame(rows).reset_index(drop=True)
    print("Все данные таблицы 'administration'")
    print(df)
```



Все данные таблицы 'administration'

	administrator_id	employee_id
0	1	1
1	2	2
2	3	3
3	4	4
4	5	5
5	6	6
6	7	7

Рисунок 2 – Вывод запроса к таблице administration.

Далее был выполнен запрос с использованием агрегатных функций, который выводит таблицу с данными о том, сколько работников прошло медосмотр, а сколько – нет, к таблице, содержащей данные о медосмотрах – medical_examinations. Также была построена круговая диаграмма, описывающая ситуацию. Код, производящий действие, показан ниже.

```
with connection.cursor() as cursor:
    query: str = """
    SELECT SUM(CASE WHEN medical_examination='Пройден' THEN 1
ELSE 0 END) AS 'Пройден', SUM(CASE WHEN
medical_examination="Предстоит пройти" THEN 1 ELSE 0 END) AS
'Предстоит пройти' FROM `medical_examinations`;"""

    cursor.execute(query)
    rows = cursor.fetchall()

    df: pd.DataFrame = pd.DataFrame(rows).T.reset_index()
    df.columns = ["Статус", "Количество"]

    plt.figure(figsize=(8, 8))
    plt.pie(x=df["Количество"],
            labels=df["Статус"],
            startangle=140,
            colors=["lightgreen", "red"],
            autopct='%1.1f%%',
            shadow=True)
    plt.title(f"Статус медосмотра по количеству людей - \nвсего
{sum(df["Количество"])} записей", fontsize=20)
    plt.savefig("Графики/pie.png")
```

Результат запроса ниже. Только небольшая часть – 14.3% человек не прошло медосмотр. Так как в таблице 7 записей, следовательно – медосмотр

предстоит пройти только 1 человеку, что является неплохим показателем. Однако стоит поторопиться в продвижении к 100% пройденных медосмотров, так как это может повлиять на допуск к работе.

Рисунок 3 – Статус медосмотра по количеству людей

Далее был выполнен запрос с использованием группировки. Запрос выводит количество билетов, группируя по статусу.

```
with connection.cursor() as cursor:
    query: str = "SELECT status AS 'Статус', COUNT(*) AS 'Количество' FROM tickets GROUP BY status;"
    cursor.execute(query)
    rows = cursor.fetchall()

    df: pd.DataFrame = pd.DataFrame(rows)
```

```
plt.figure(figsize=(10, 8))
plt.bar(df["Статус"], df["Количество"])
plt.title('Количество билетов по статусу', fontsize=20)
plt.xlabel('Статус', fontsize=15)
plt.ylabel('Количество', fontsize=15)

plt.savefig("Графики/bars.png")
```

Далее представлена диаграмма, полученная после выполнения запроса. В системе зарегистрировано 3 билета: 2 в статусе забронирован, 1 в статусе отменён. Данных мало, скорее всего при наполнении базы билетами баланс отменённых и забронированных изменится.

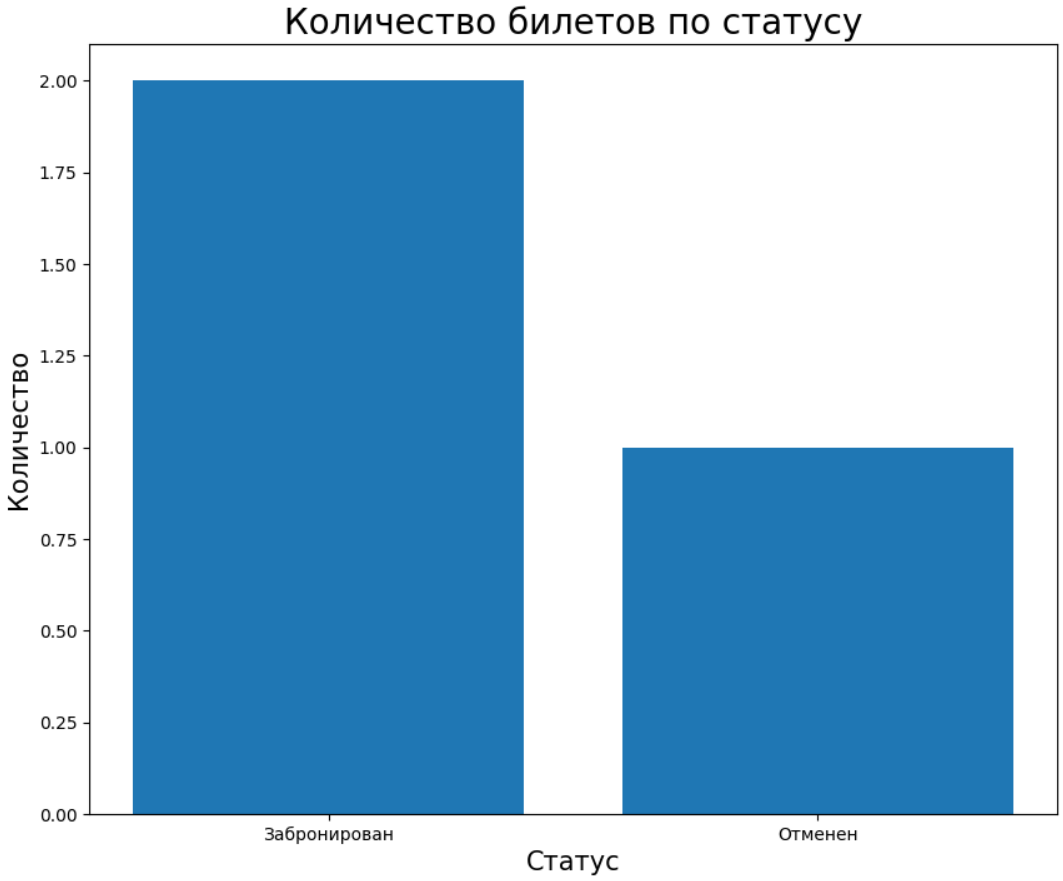


Рисунок 4 – Количество билетов по статусу

Далее был выполнен запрос с использованием сортировки по дате рождения и создан график распределения по возрасту. Далее представлен код запроса и результат его работы.

```
with connection.cursor() as cursor:
    query: str = "SELECT * FROM `employees` ORDER BY
```

```
date_of_birth;"
    cursor.execute(query)
    rows = cursor.fetchall()

    df: pd.DataFrame = pd.DataFrame(rows)

    df['date_of_birth'] = pd.to_datetime(df['date_of_birth'])
    df['age']       =       (datetime.today().date().year       -
df['date_of_birth'].dt.year)

    df.plot(x="age", y="age", kind="scatter")
    plt.xlabel("Возраст (лет)")
    plt.ylabel("Возраст (лет)")
    plt.title("Распределение людей по возрасту")
    plt.savefig("Графики/plot.png")
```

Программа получает из базы данных отсортированные по году рождения данные и строит график. См. рис. 5.

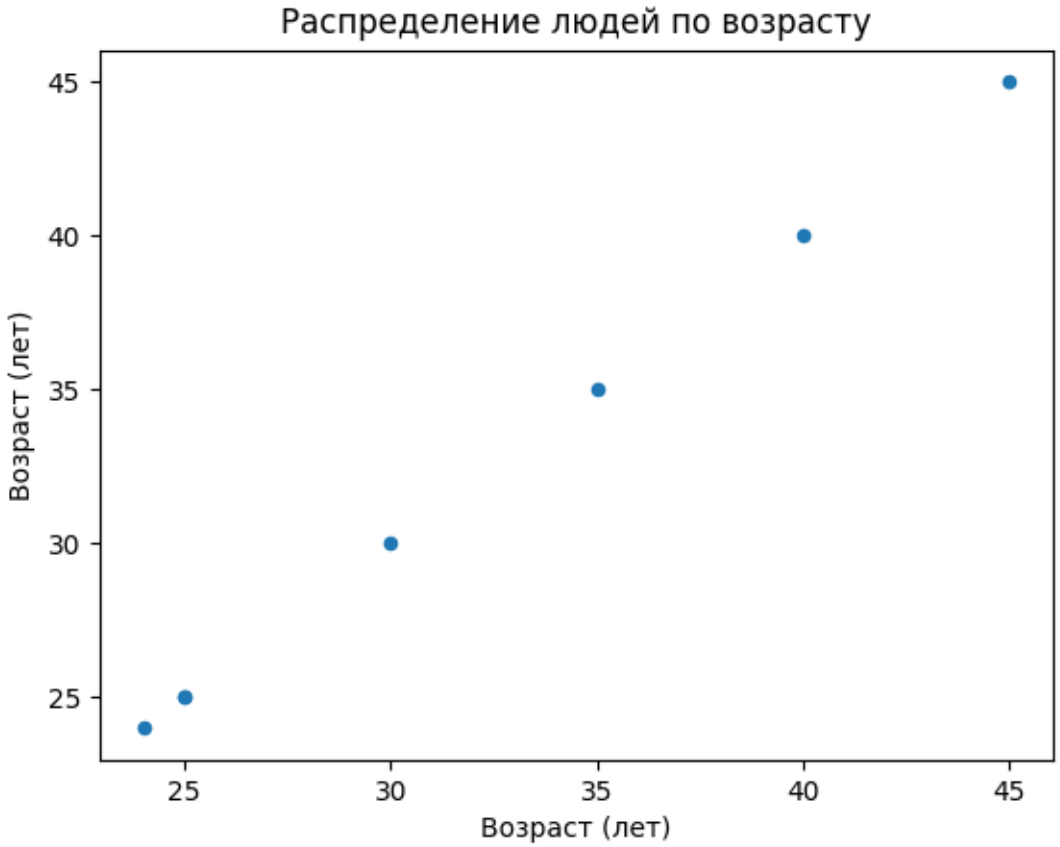


Рисунок 5 – Распределение людей по возрасту

Имеется равномерное распределение людей по возрасту, однако молодых лиц немного больше.

4. **Листинг кода:** см. приложение А.

5. **Вывод:**

В ходе работы была установлена связь между базой данных MySQL и Python, что позволило извлечь данные из таблиц базы данных и выполнить их анализ с помощью визуализации. Были выполнены различные SQL-запросы, включая агрегатные функции, группировки и сортировки, что позволило получить информацию о статусе медосмотров сотрудников, количестве билетов по статусу и распределении сотрудников по возрасту. Все результаты были визуализированы с помощью графиков и диаграмм, что позволило наглядно представить данные. Работа показала эффективность использования SQL и Python для анализа и визуализации данных из базы данных, а также возможности дальнейшего улучшения системы с увеличением объема данных.

Приложение А

Полный листинг программы

```
import pandas as pd
from datetime import datetime
import matplotlib.pyplot as plt
import pymysql.cursors


# Установка связи между Python и MySQL
connection: pymysql.connect = pymysql.connect(host="localhost",
                                              user="root",
                                              password="qwerty",
                                              db="train_station",
                                              charset="utf8mb4",

cursorclass=pymysql.cursors.DictCursor)


# Вывод с данными всей таблицы
with connection.cursor() as cursor:
    query: str = "SELECT * FROM `administration`;"
    cursor.execute(query)
    rows = cursor.fetchall()
    df: pd.DataFrame = pd.DataFrame(rows).reset_index(drop=True)
    print("Все данные таблицы 'administration'")
    print(df)


# Вывод данных с использованием агрегатной функции count, создание
круговой диаграммы
with connection.cursor() as cursor:
    query: str = """
    SELECT SUM(CASE WHEN medical_examination='Пройден' THEN 1 ELSE 0
END) AS 'Пройден', SUM(CASE WHEN medical_examination="Предстоит
пройти" THEN 1 ELSE 0 END) AS 'Предстоит пройти' FROM
`medical_examinations`;"""
```

```
    cursor.execute(query)
    rows = cursor.fetchall()

    df: pd.DataFrame = pd.DataFrame(rows).T.reset_index()
    df.columns = ["Статус", "Количество"]

    plt.figure(figsize=(8, 8))
    plt.pie(x=df["Количество"],
            labels=df["Статус"],
            startangle=140,
            colors=["lightgreen", "red"],
            autopct='%1.1f%%',
            shadow=True)
    plt.title(f"Статус медосмотра по количеству людей - \nвсего
{sum(df["Количество"])} записей", fontsize=20)
    plt.savefig("Графики/pie.png")


# Вывод данных с использованием группировки, создание столбчатой
диаграммы
with connection.cursor() as cursor:
    query: str = "SELECT status AS 'Статус', COUNT(*) AS 'Количество'
FROM tickets GROUP BY status;"
    cursor.execute(query)
    rows = cursor.fetchall()

    df: pd.DataFrame = pd.DataFrame(rows)

    plt.figure(figsize=(10, 8))
    plt.bar(df["Статус"], df["Количество"])
    plt.title('Количество билетов по статусу', fontsize=20)
    plt.xlabel('Статус', fontsize=15)
    plt.ylabel('Количество', fontsize=15)

    plt.savefig("Графики/bars.png")


# Вывод данных с использованием сортировки
with connection.cursor() as cursor:
    query: str = "SELECT * FROM `employees` ORDER BY date_of_birth;"
    cursor.execute(query)
    rows = cursor.fetchall()

    df: pd.DataFrame = pd.DataFrame(rows)

    df['date_of_birth'] = pd.to_datetime(df['date_of_birth'])
    df['age'] = (datetime.today().date().year -
df['date_of_birth'].dt.year)

    df.plot(x="age", y="age", kind="scatter")
    plt.xlabel("Возраст (лет)")
    plt.ylabel("Возраст (лет)")
    plt.title("Распределение людей по возрасту")
    plt.savefig("Графики/plot.png")
```