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 №7

ВИЗУАЛИЗАЦИЯ ДАННЫХ ИЗ СУБД POSTGRESQL В PYTHON

по курсу: БАЗЫ ДАННЫХ

РАБОТУ ВЫПОЛНИЛ

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

Санкт-Петербург 2025

Цель работы

Произвести связь базы данных в PostgreSQL и Python, изучить операции по манипулированию с данными БД, выполнить анализ данных в БД с помощью визуализации в Python.

Вариант

Вариант 4.

Решение

Сначала был сделан скриншот схемы базы данных. См. рис. 1.

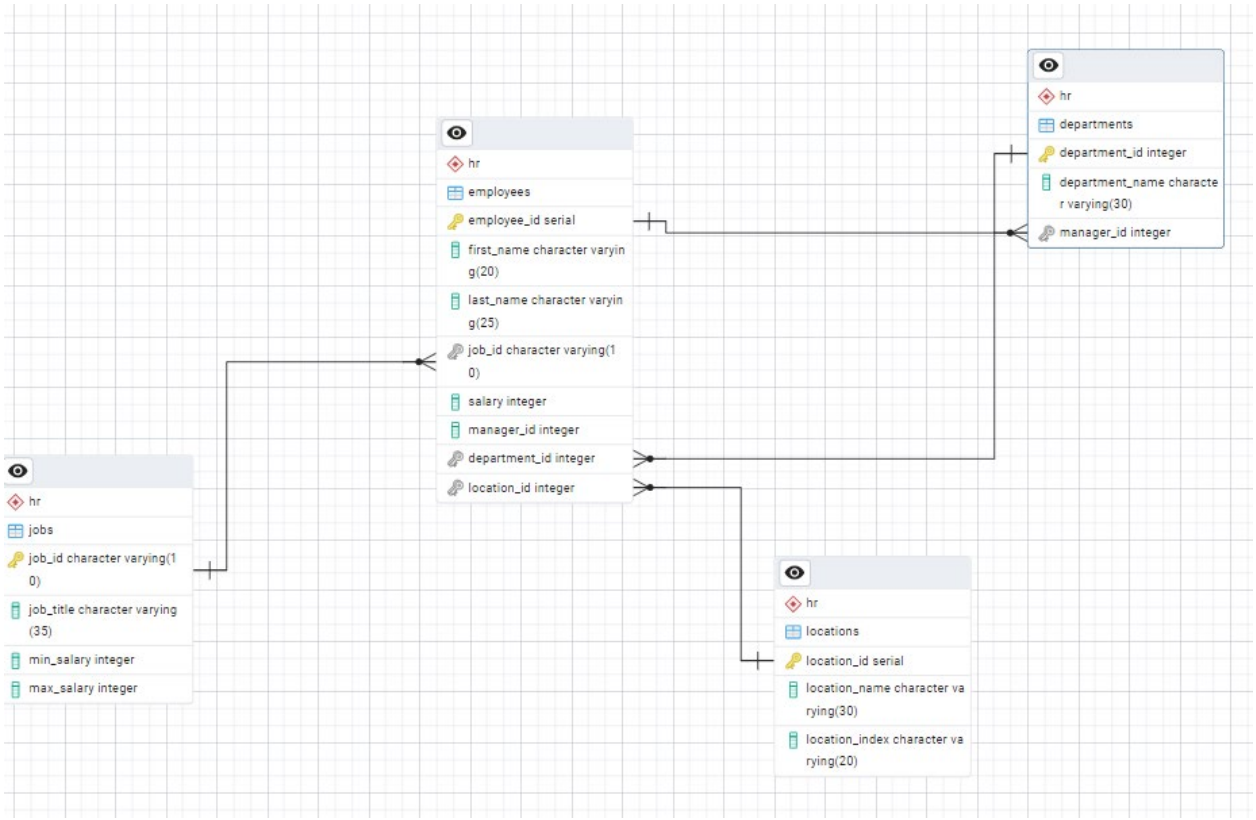

Рисунок 1 – Схема базы данных

Далее были сделаны графики. См. рис. 2-5.

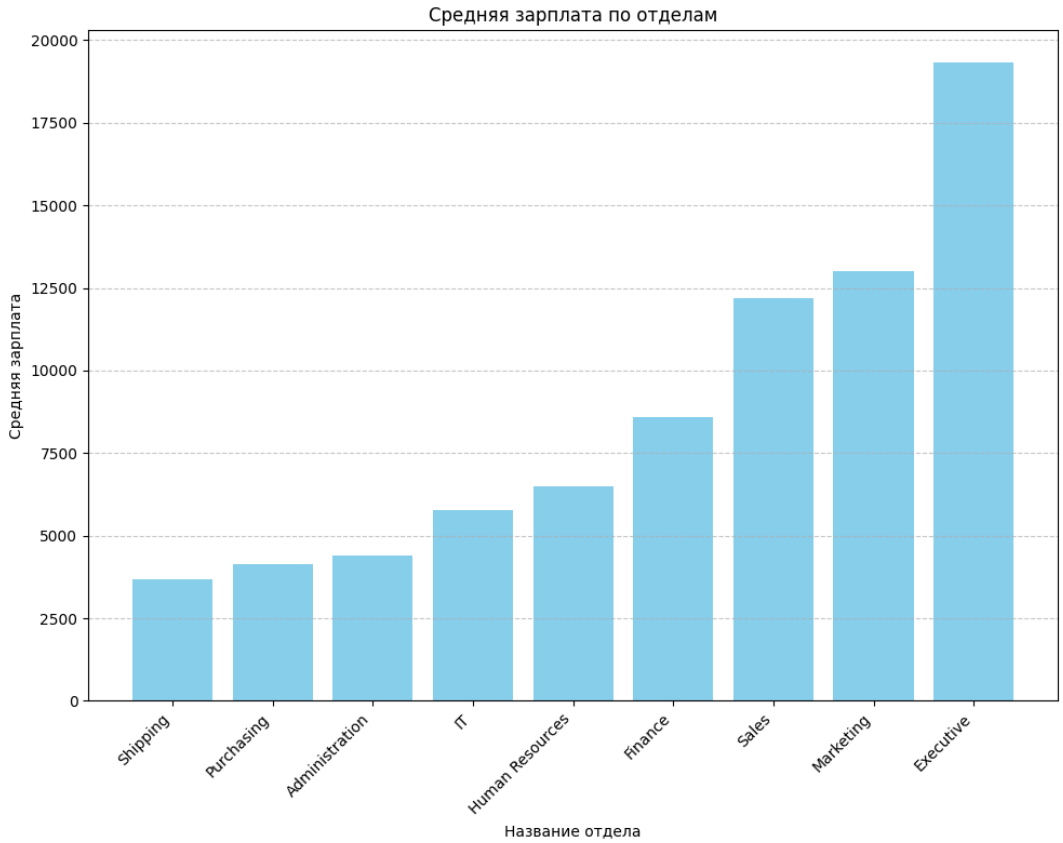

Рисунок 2 – Вертикальный график средней зарплаты

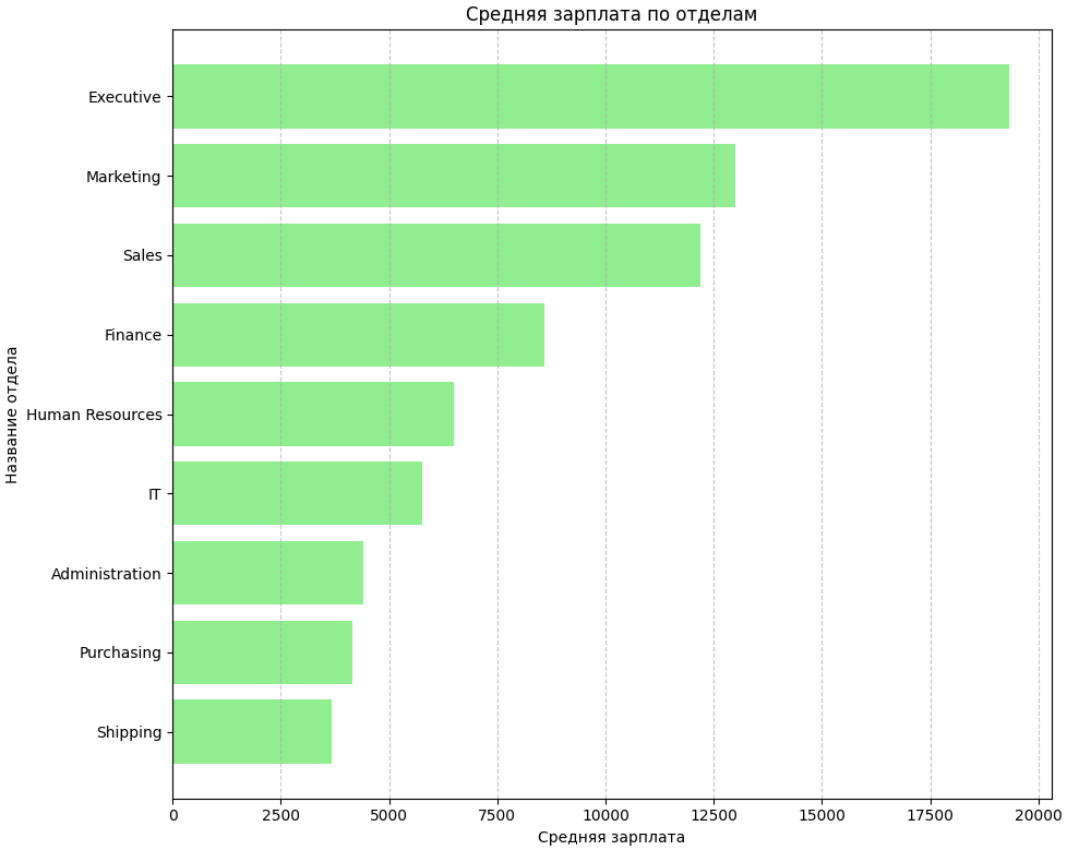

Рисунок 3 – Горизонтальный график средней зарплаты

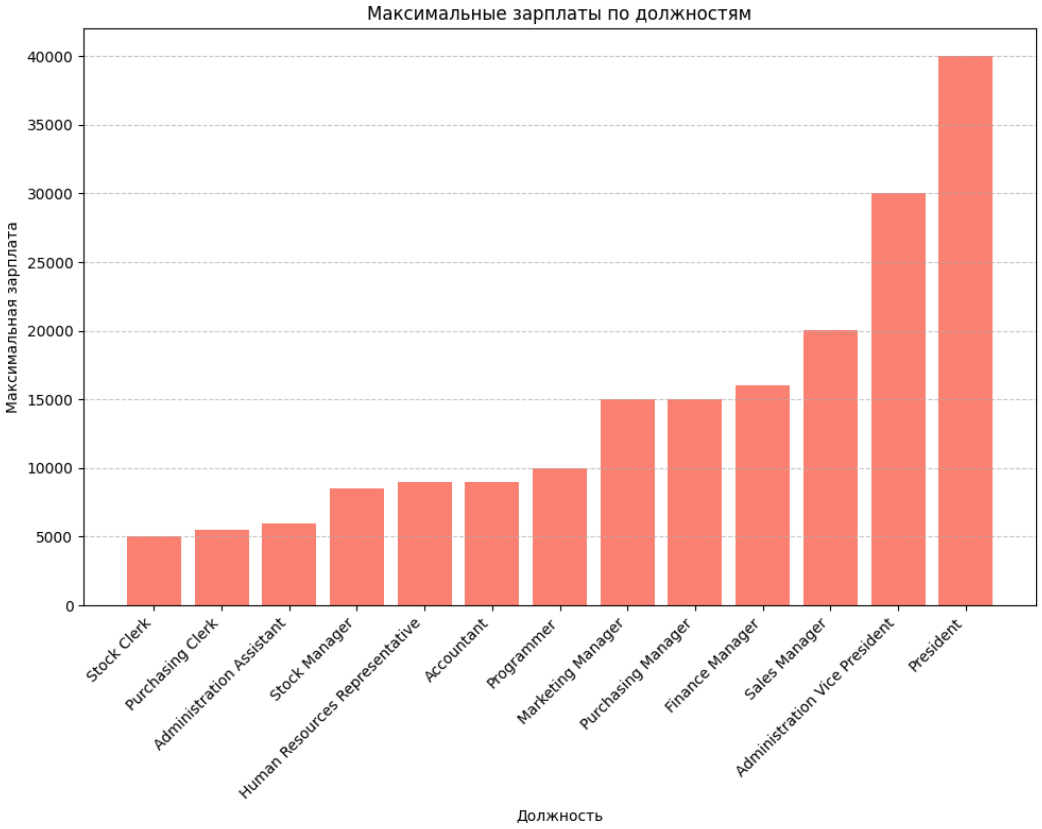

Рисунок 4 – Вертикальный график максимальной зарплаты

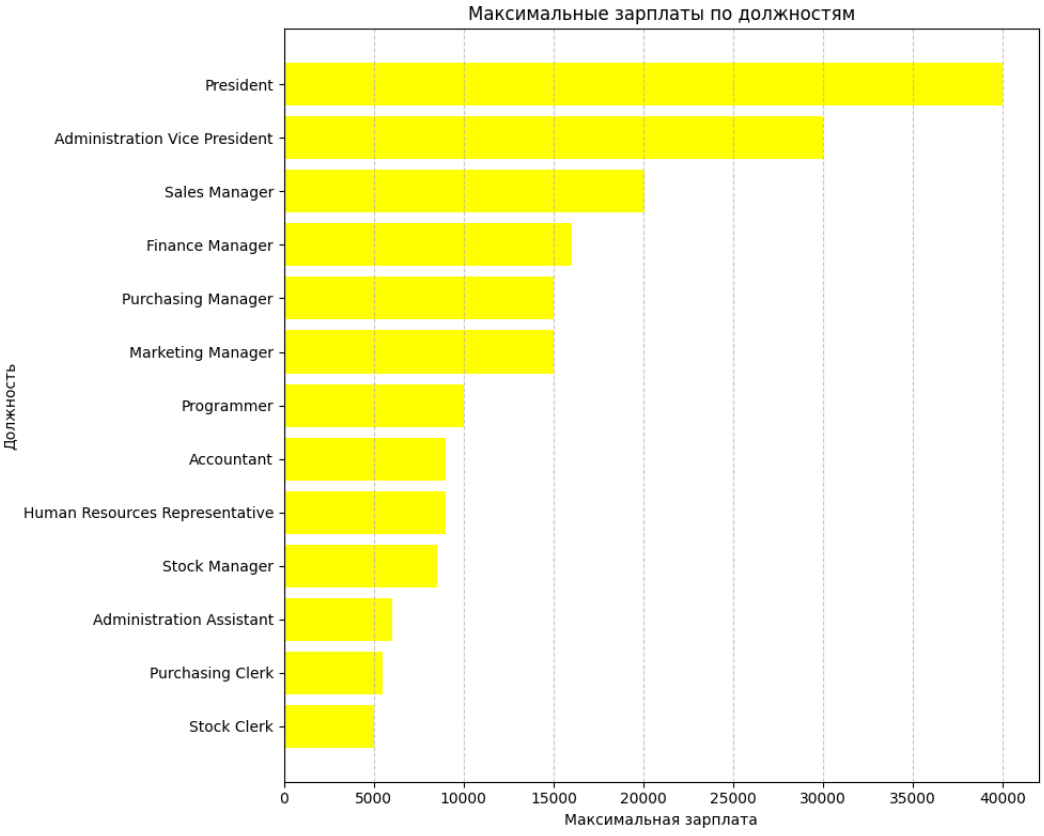

Рисунок 5 – Горизонтальный график максимальной зарплаты

Далее были добавлены диапазоны по зарплате для этих графиков. См. рис. 6-9.

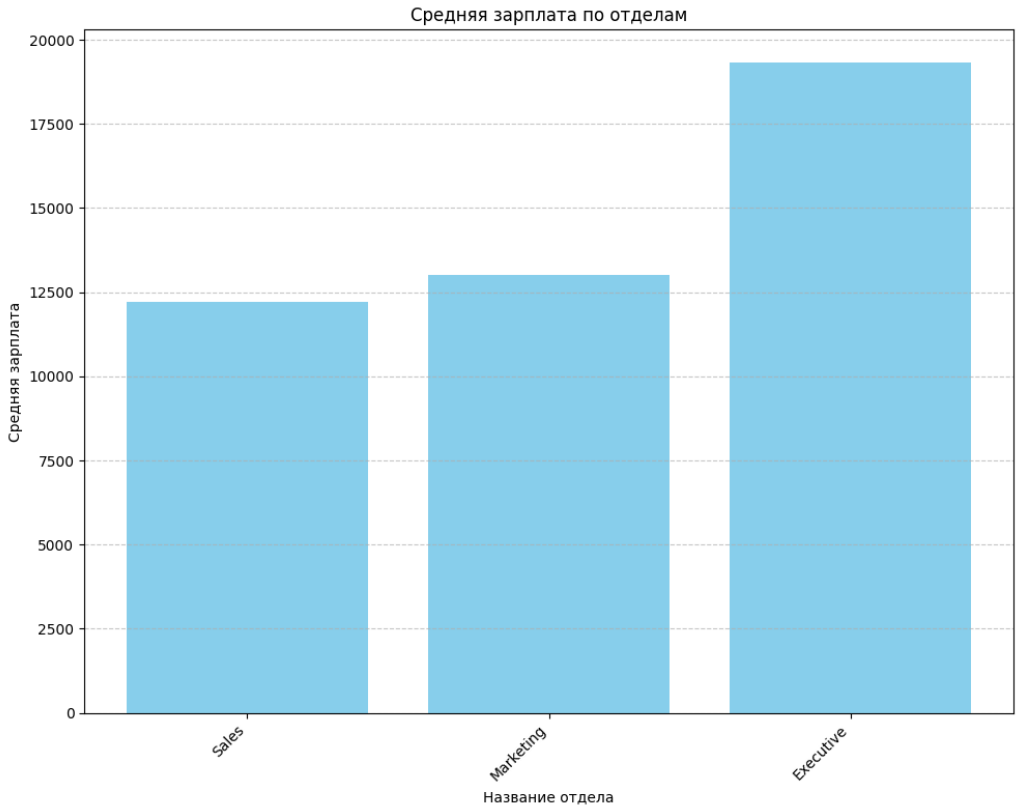

Рисунок 6 – Средние зарплаты от 10000 до 20000 вертикальный

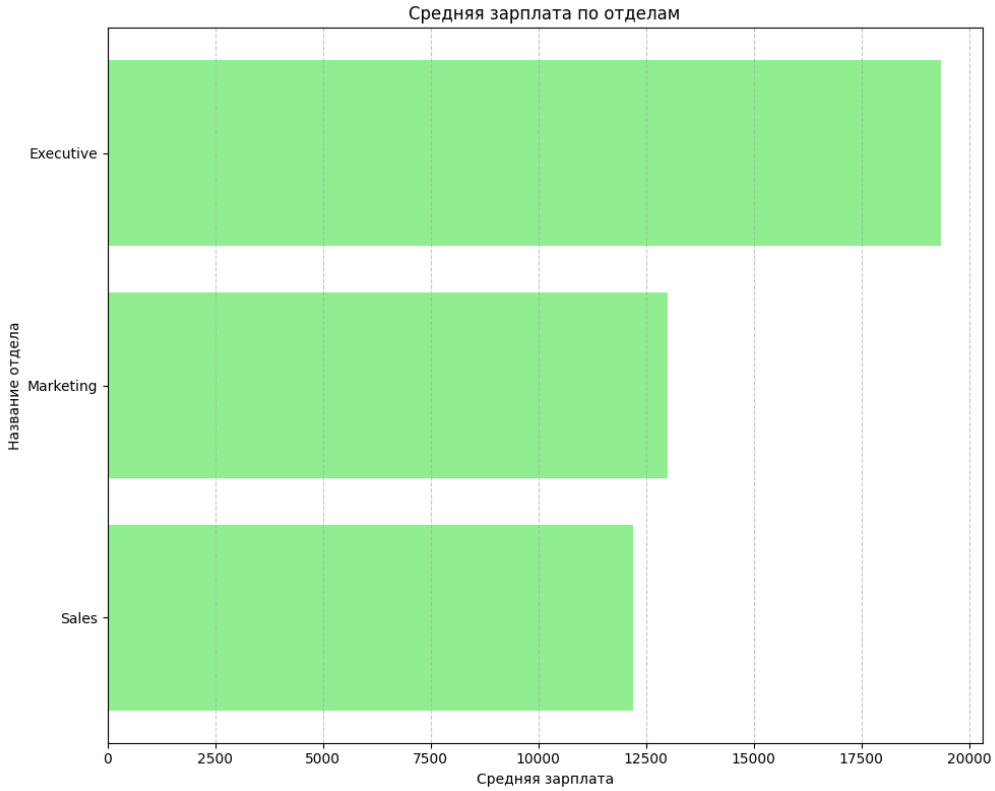

Рисунок 7 – Средние зарплаты от 10000 до 20000 горизонтальный

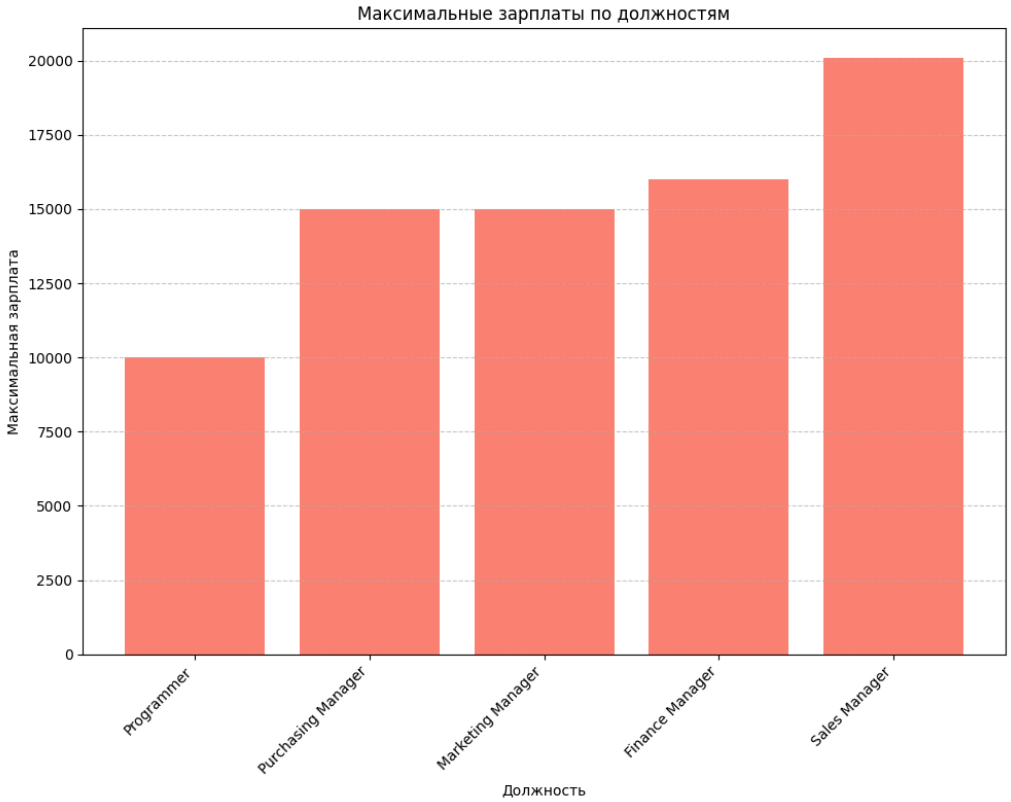

Рисунок 8 – Максимальные зарплаты по должности от 10000 до 25000 вертикальный

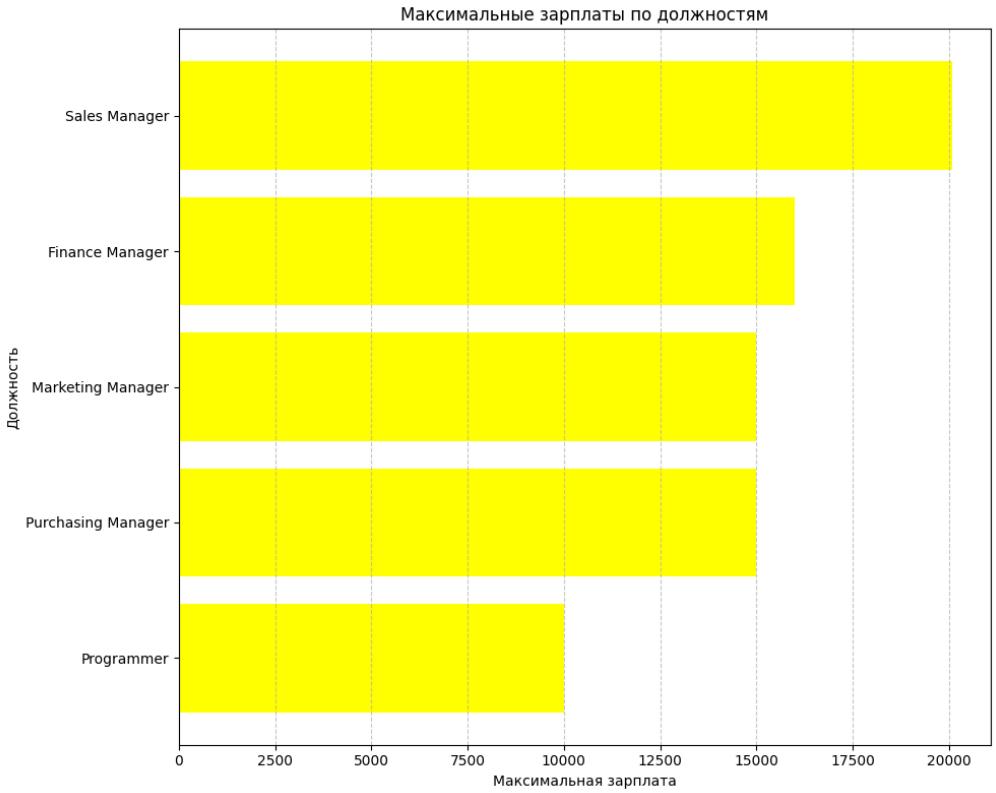

Рисунок 9 – Максимальные зарплаты по должности от 10000 до 25000 горизонтальный

Далее была создана круговая диаграмма с распределением работников по различным локациям. См. рис. 10.

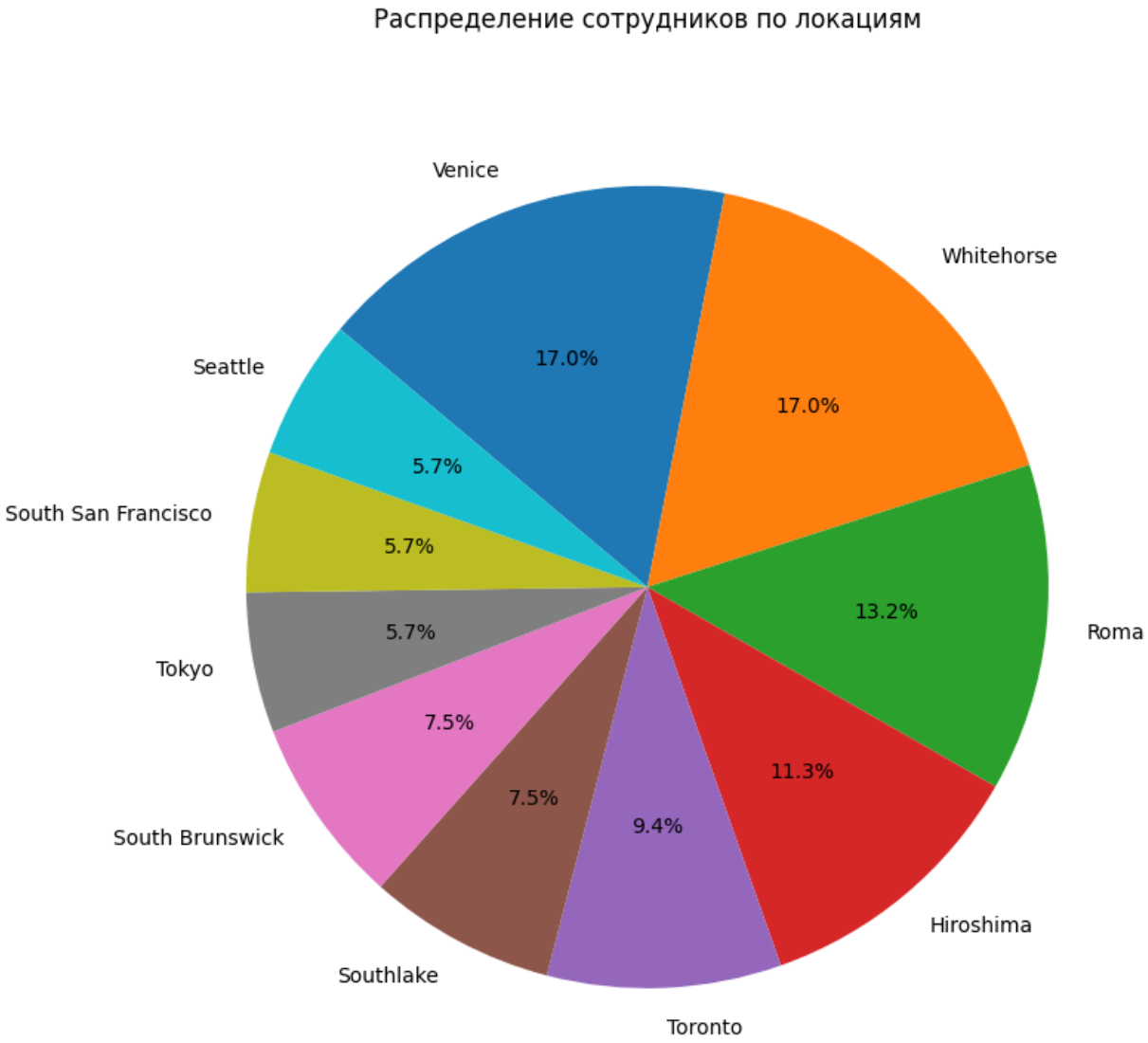

Рисунок 10 – Работники по локациям

Листинг кода

Необходимые для запуска компоненты:

```
python==3.13
contourpy==1.3.2
cycler==0.12.1
dotenv==0.9.9
fonttools==4.58.0
greenlet==3.2.2
kiwisolver==1.4.8
matplotlib==3.10.3
numpy==2.2.6
```

```
packaging==25.0
pandas==2.2.3
pillow==11.2.1
psycopg2==2.9.10
pyparsing==3.2.3
python-dateutil==2.9.0.post0
python-dotenv==1.1.0
pytz==2025.2
six==1.17.0
SQLAlchemy==2.0.41
typing_extensions==4.13.2
tzdata==2025.2
```

См. приложение А.

Вывод

В ходе лабораторной работы была выполнена визуализация данных из базы данных PostgreSQL средствами языка Python с использованием библиотеки SQLAlchemy и pandas, и инструментария визуализации Matplotlib. Были построены столбчатые, горизонтальные и круговые диаграммы, отражающие:

1. средние и максимальные значения зарплат по отделам и должностям;
2. распределение сотрудников по различным локациям.

Были реализованы фильтры по диапазонам зарплат, что позволило провести более детальный анализ. Полученные графики визуально подтверждают закономерности в структуре данных и позволяют быстро выявить отклонения и ключевые особенности.

Работа продемонстрировала практическое применение связки Python + PostgreSQL для анализа данных, а также навыки работы с ORM, SQL-запросами и средствами визуализации. Поставленная цель была достигнута в полном объёме.

Приложение А

«Код программы»

```
from sqlalchemy import create_engine
from sqlalchemy.orm import declarative_base, sessionmaker
from sqlalchemy import Column, Integer, String, ForeignKey, func
from dotenv import load_dotenv
from os import getenv
import pandas as pd
import matplotlib.pyplot as plt

Base = declarative_base()


class Employee(Base):
    __tablename__ = 'employees'
    __table_args__ = {'schema': 'hr'}
    employee_id = Column(Integer, primary_key=True,
nullable=False)
    first_name = Column(String(20), nullable=False)
    last_name = Column(String(25), nullable=False)
    job_id = Column(String(10), ForeignKey("jobs.job_id"),
nullable=False)
    salary = Column(Integer, nullable=True)
    manager_id = Column(Integer)
    department_id = Column(Integer,
ForeignKey("departments.department_id"), nullable=False)
    location_id = Column(Integer,
ForeignKey("locations.location_id"), nullable=False)

    def __repr__(self):
        return f"employee_id: {self.employee_id}, first_name:
{self.first_name}, last_name: {self.last_name}, job_id:
{self.job_id}, salary: {self.salary}, manager_id:
{self.manager_id}, department_id: {self.department_id}"


class Department(Base):
    __tablename__ = 'departments'
    __table_args__ = {'schema': 'hr'}
    department_id = Column(Integer, primary_key=True,
nullable=False)
    department_name = Column(String(30), nullable=False)
    manager_id = Column(Integer,
ForeignKey("employees.employee_id"))


class Job(Base):
    __tablename__ = 'jobs'
    __table_args__ = {'schema': 'hr'}
    job_id = Column(String(10), primary_key=True,
nullable=False)
```

```
    job_title = Column(String(35), nullable=False)
    min_salary = Column(Integer)
    max_salary = Column(Integer)


class Location(Base):
    __tablename__ = 'locations'
    __table_args__ = {'schema': 'hr'}

    location_id = Column(Integer, primary_key=True,
nullable=False)
    location_name = Column(String(30), nullable=False)
    location_index = Column(String(20), nullable=False)


def draw_employees_by_location_pie() -> None:
    with Session() as session:
        # Получаем количество сотрудников в каждой локации
        query = (
            session.query(
                Location.location_name.label("location"),

func.count(Employee.employee_id).label("employee_count")
            )
            .join(Employee, Employee.location_id ==
Location.location_id)
            .group_by(Location.location_name)
            .order_by(func.count(Employee.employee_id).desc())
        )

        df = pd.read_sql_query(query.statement, session.bind)

        # Построение круговой диаграммы
        plt.figure(figsize=(8, 8))
        plt.title("Распределение сотрудников по локациям")

        plt.pie(
            df["employee_count"],
            labels=df["location"],
            autopct="%1.1f%%",
            startangle=140,
            counterclock=False
        )

        plt.axis("equal")  # Делает круг ровным
        plt.tight_layout()
        plt.savefig("graphics/employees_by_location_pie.png")
        plt.show()


def draw_avg_salary(horizontal: bool = False, min_salary: float
= 0, max_salary: float = float('inf')) -> None:
    """
```

```
    Строит диаграмму средней зарплаты по отделам.
    Можно указать, горизонтальная ли диаграмма, и диапазон
средней зарплаты для фильтрации.

    :param horizontal: Отображать горизонтально (True) или
вертикально (False)
    :param min_salary: Минимальная средняя зарплата для
включения в график
    :param max_salary: Максимальная средняя зарплата для
включения в график
    """
    with Session() as session:
        subquery = (
            session.query(

Department.department_name.label("department_name"),
                func.round(func.avg(Employee.salary),
2).label("avg_salary")
            )
            .join(Department, Employee.department_id ==
Department.department_id)
            .group_by(Department.department_name)

.having(func.avg(Employee.salary).between(min_salary,
max_salary))
            .order_by(func.avg(Employee.salary))
        )

        result = subquery.all()

    df = pd.DataFrame(
        [{
            "Department_Name": department.department_name,
            "AVG_Salary": department.avg_salary
        } for department in result]
    ).reset_index(drop=True)

    if df.empty:
        print("Нет данных для отображения в заданном
диапазоне.")
        return

    # Построение графика
    plt.figure(figsize=(10, 8))
    plt.title("Средняя зарплата по отделам")

    if not horizontal:
        plt.bar(df["Department_Name"], df["AVG_Salary"],
color='skyblue')
        plt.xlabel("Название отдела")
        plt.ylabel("Средняя зарплата")
        plt.xticks(rotation=45, ha='right')
        plt.grid(axis='y', linestyle='--', alpha=0.7)
```

```
    else:
        plt.barh(df["Department_Name"], df["AVG_Salary"],
color='lightgreen')
        plt.xlabel('Средняя зарплата')
        plt.ylabel('Название отдела')
        plt.grid(axis='x', linestyle='--', alpha=0.7)

    plt.tight_layout()
    plt.savefig(

f"graphics/avg_salary_horizontal_{str(horizontal)}_with_range_{s
tr(min_salary)}_to_{str(max_salary)}.png")
    plt.show()


def draw_max_salary(horizontal: bool = False, min_total: float =
0, max_total: float = float('inf')) -> None:
    """
    Строит график максимальной зарплаты по должностям, фильтруя
по суммарной зарплате сотрудников на этих должностях.

    :param horizontal: Отображать график горизонтально (True)
или вертикально (False)
    :param min_total: Минимальная суммарная зарплата по
должности для включения
    :param max_total: Максимальная суммарная зарплата по
должности для включения
    """
    with Session() as session:
        query = (
            session.query(Job.job_title, Job.max_salary)
            .filter(Job.max_salary.between(min_total,
max_total))
            .order_by(Job.max_salary)
        )

        df = pd.read_sql_query(query.statement,
query.session.bind)

    if df.empty:
        print("Нет данных для отображения в заданном диапазоне
суммарных зарплат.")
        return

    # Визуализация
    plt.figure(figsize=(10, 8))
    plt.title("Максимальные зарплаты по должностям")

    if not horizontal:
        plt.bar(df["job_title"], df["max_salary"],
color='salmon')
        plt.xlabel("Должность")
        plt.ylabel("Максимальная зарплата")
```

```
        plt.xticks(rotation=45, ha='right')
        plt.grid(axis='y', linestyle='--', alpha=0.7)
    else:
        plt.barh(df["job_title"], df["max_salary"],
color='yellow')
        plt.xlabel("Максимальная зарплата")
        plt.ylabel("Должность")
        plt.grid(axis='x', linestyle='--', alpha=0.7)

    plt.tight_layout()

plt.savefig(f"graphics/max_salary_horizontal_{str(horizontal)}_w
ith_range_{str(min_total)}_to_{str(max_total)}.png")
    plt.show()


if __name__ == '__main__':
    load_dotenv()  # Загрузка секретных переменных
    # Секретное формирование ссылки для подключения к БД
    postgres_link =
f"{getenv("DBTYPE")}+{getenv("PSQLDRIVER")}://{getenv("DBUSERNAM
E")}:{getenv("PASSWORD")}@{getenv("HOST")}:{getenv("PORT")}/{get
env("DATABASE")}"
    engine = create_engine(postgres_link)  # Создание движка для
работы с БД
    Base.metadata.create_all(engine)  # Создание всех таблиц
    Session = sessionmaker(bind=engine)  # Фабрика сессий

    draw_avg_salary()
    draw_avg_salary(horizontal=True)
    draw_avg_salary(horizontal=False, min_salary=10000,
max_salary=20000)
    draw_avg_salary(horizontal=True, min_salary=10000,
max_salary=20000)
    draw_max_salary()
    draw_max_salary(horizontal=True)
    draw_max_salary(horizontal=False, min_total=10000,
max_total=25000)
    draw_max_salary(horizontal=True, min_total=10000,
max_total=25000)
    draw_employees_by_location_pie()
```