

# Пример работы с библиотекой SQLAlchemy

Убираем предупреждения для красоты.

```
import warnings
warnings.filterwarnings('ignore')
```

Немного увеличиваем уже существующую БД.

```
import pandas as pd
from sqlalchemy import create_engine, Column, Integer, String, Date
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
from datetime import date

# 1. Подключение к базе данных PostgreSQL
DATABASE_URL =
"postgresql://postgres:Arseniy228@localhost:5432/postgres"

# Создание движка подключения
engine = create_engine(DATABASE_URL)

# 2. Определение базового класса для моделей
Base = declarative_base()

# 3. Определение модели User
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True, autoincrement=True)
    name = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    date_of_birth = Column(Date)

    def __repr__(self):
        return f"<User(id={self.id}, name={self.name},\n\
email={self.email}, date_of_birth={self.date_of_birth})>"

# 4. Создание сессии для работы с БД
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)
session = SessionLocal()

# 5. Список пользователей, которых мы хотим добавить
users_to_add = [
    ('Arseniy', 'arseniyl@example.com', date(1995, 4, 23)),
    ('Ivan', 'ivan@example.com', date(1992, 3, 15)),
    ('Elena', 'elena@example.com', date(1988, 9, 12)),
```

```

('Dmitry', 'dmitry@example.com', date(1990, 6, 25)),
('Maria', 'maria@example.com', date(1996, 2, 14)),
('Alexey', 'alexey@example.com', date(1993, 8, 30)),
('Nina', 'nina@example.com', date(1997, 11, 7)),
('Oleg', 'oleg@example.com', date(1989, 4, 22)),
('Svetlana', 'svetlana@example.com', date(1991, 12, 5)),
('Igor', 'igor@example.com', date(1994, 7, 18)),
('Anna', 'anna@example.com', date(1990, 10, 21)),
('Sergey', 'sergey@example.com', date(1992, 1, 16)),
('Tatyana', 'tatyana@example.com', date(1995, 9, 11)),
('Maxim', 'maxim@example.com', date(1987, 5, 25)),
('Olga', 'olga@example.com', date(1996, 4, 30)),
('Yulia', 'yulia@example.com', date(1993, 12, 12)),
('Vladimir', 'vladimir@example.com', date(1990, 3, 17)),
('Valentina', 'valentina@example.com', date(1992, 2, 8)),
('Pavel', 'pavel@example.com', date(1994, 11, 19)),
('Ekaterina', 'ekaterina@example.com', date(1991, 6, 13))
]

```

*# 6. Добавление пользователей в базу данных*

```

try:
    for name, email, dob in users_to_add:
        user = User(name=name, email=email, date_of_birth=dob)
        session.add(user)

    # Сохраняем изменения в базе данных
    session.commit()
    print("Пользователи успешно добавлены!")

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

finally:
    # Закрытие сессии
    session.close()

```

Пользователи успешно добавлены!

Выведем данные таблицы для проверки.

```

from sqlalchemy import create_engine, Column, Integer, String, Date
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker

# 1. Подключение к базе данных PostgreSQL
DATABASE_URL =
"postgresql://postgres:Arseuiy228@localhost:5432/postgres"

# Создание движка подключения

```

```

engine = create_engine(DATABASE_URL)

# 2. Определение базового класса для моделей
Base = declarative_base()

# 3. Определение таблицы
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    date_of_birth = Column(Date)

    def __repr__(self):
        return f"<User(id={self.id}, name={self.name}, email={self.email}, date_of_birth={self.date_of_birth})>"

# 4. Создание таблиц в базе данных (если они не существуют)
Base.metadata.create_all(engine)

# 5. Создание сессии для работы с БД
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
session = SessionLocal()

# 6. Выполнение запроса для получения данных
users = session.query(User).all()
for user in users:
    print(user)

# 7. Закрытие сессии
session.close()

<User(id=1, name=Alice, email=None, date_of_birth=None)>
<User(id=2, name=Bob, email=None, date_of_birth=None)>
<User(id=3, name=Charlie, email=None, date_of_birth=None)>
<User(id=4, name=Diana, email=None, date_of_birth=None)>
<User(id=7, name=John Doe, email=johndoe@example.com, date_of_birth=1990-01-01)>
<User(id=10, name=Alice, email=alice@example.com, date_of_birth=1992-06-15)>
<User(id=11, name=Bob, email=bob@example.com, date_of_birth=1985-12-02)>
<User(id=12, name=Charlie, email=charlie@example.com, date_of_birth=1995-03-23)>
<User(id=14, name=Arseniy, email=ars@example.com, date_of_birth=1995-04-23)>
<User(id=16, name=Arseniy, email=arseniy1@example.com, date_of_birth=1995-04-23)>

```

```
<User(id=17, name=Ivan, email=ivan@example.com, date_of_birth=1992-03-15)>
<User(id=18, name=Elena, email=elena@example.com, date_of_birth=1988-09-12)>
<User(id=19, name=Dmitry, email=dmitry@example.com, date_of_birth=1990-06-25)>
<User(id=20, name=Maria, email=maria@example.com, date_of_birth=1996-02-14)>
<User(id=21, name=Alexey, email=alexey@example.com, date_of_birth=1993-08-30)>
<User(id=22, name=Nina, email=nina@example.com, date_of_birth=1997-11-07)>
<User(id=23, name=Oleg, email=oleg@example.com, date_of_birth=1989-04-22)>
<User(id=24, name=Svetlana, email=svetlana@example.com, date_of_birth=1991-12-05)>
<User(id=25, name=Igor, email=igor@example.com, date_of_birth=1994-07-18)>
<User(id=26, name=Anna, email=anna@example.com, date_of_birth=1990-10-21)>
<User(id=27, name=Sergey, email=sergey@example.com, date_of_birth=1992-01-16)>
<User(id=28, name=Tatyana, email=tatyana@example.com, date_of_birth=1995-09-11)>
<User(id=29, name=Maxim, email=maxim@example.com, date_of_birth=1987-05-25)>
<User(id=30, name=Olga, email=olga@example.com, date_of_birth=1996-04-30)>
<User(id=31, name=Yulia, email=yulia@example.com, date_of_birth=1993-12-12)>
<User(id=32, name=Vladimir, email=vladimir@example.com, date_of_birth=1990-03-17)>
<User(id=33, name=Valentina, email=valentina@example.com, date_of_birth=1992-02-08)>
<User(id=34, name=Pavel, email=pavel@example.com, date_of_birth=1994-11-19)>
<User(id=35, name=Ekaterina, email=ekaterina@example.com, date_of_birth=1991-06-13)>
```

Сделаем столбчатую диаграмму распределения пользователей по годам рождения.

```
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import create_engine, Column, Integer, String, Date, func
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
```

```

# 1. Подключение к базе данных PostgreSQL
DATABASE_URL =
"postgresql://postgres:Arseniy228@localhost:5432/postgres"

# Создание движка подключения
engine = create_engine(DATABASE_URL)

# 2. Определение базового класса для моделей
Base = declarative_base()

# 3. Определение модели User
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    date_of_birth = Column(Date)

    def __repr__(self):
        return f"<User(id={self.id}, name={self.name},
email={self.email})>"

# 4. Создание сессии для работы с БД
SessionLocal = sessionmaker(autocommit=False, autoflush=False,
bind=engine)
session = SessionLocal()

# 5. Выполнение запроса для получения данных о годах рождения
# Извлекаем год из даты рождения пользователей
result = session.query(
    func.extract('year', User.date_of_birth).label('birth_year'),
    func.count().label('num_users')
).group_by('birth_year').order_by('birth_year').all()

# Преобразуем результат в DataFrame для удобства
df = pd.DataFrame(result, columns=['birth_year', 'num_users'])

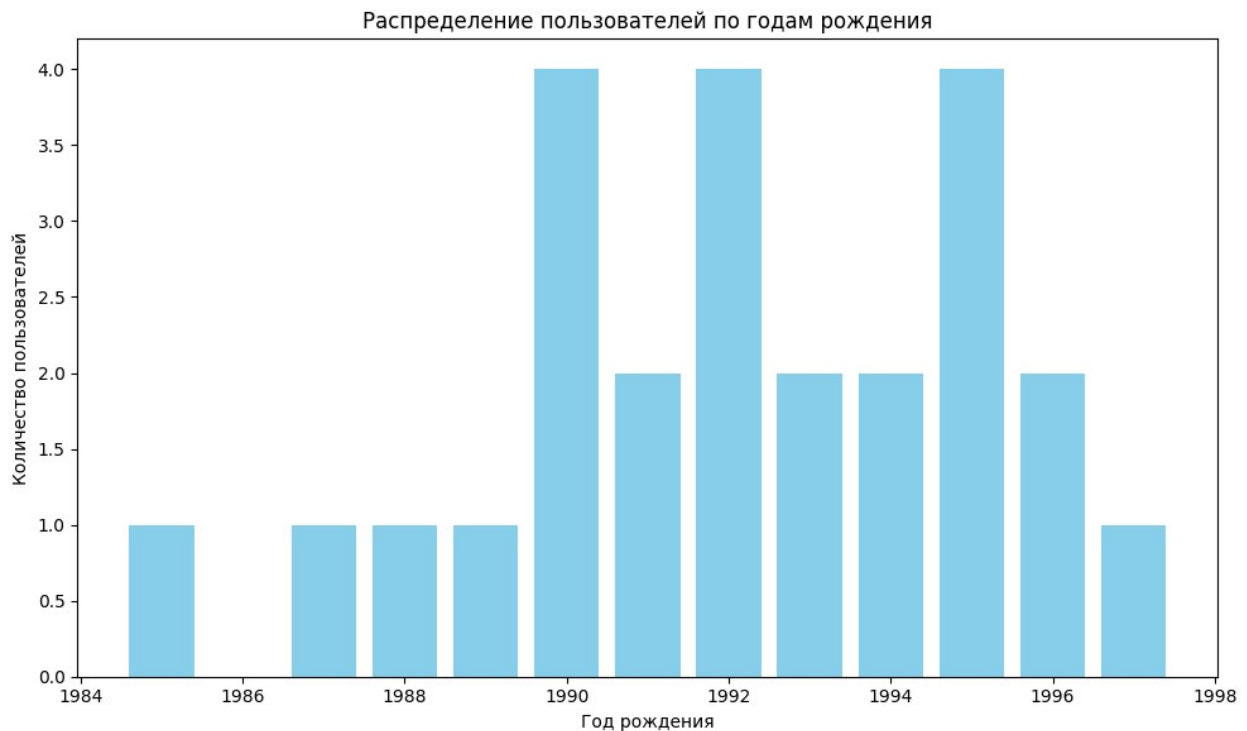
# 6. Построение графика
plt.figure(figsize=(10, 6))
plt.bar(df['birth_year'], df['num_users'], color='skyblue')

# 7. Настройка графика
plt.title('Распределение пользователей по годам рождения')
plt.xlabel('Год рождения')
plt.ylabel('Количество пользователей')

# Покажем график
plt.tight_layout()
plt.show()

```

```
# 8. Закрытие сессии
session.close()
```



Теперь можно сделать плоский график с зависимостью количества пользователей от года рождения.

```
import pandas as pd
import matplotlib.pyplot as plt
from sqlalchemy import create_engine, Column, Integer, String, Date,
func
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy.orm import sessionmaker
import psycopg2
```

```
# 1. Подключение к базе данных PostgreSQL
```

```
DATABASE_URL =
"postgresql://postgres:Arseniy228@localhost:5432/postgres"
```

```
# Попробуем установить кодировку при подключении
```

```
conn = psycopg2.connect(DATABASE_URL)
conn.set_client_encoding('LATIN1') # Используем LATIN1 или windows-
1251 для возможных проблем с кодировкой
```

```
# Создание движка подключения
```

```
engine = create_engine(DATABASE_URL)
```

```

# 2. Определение базового класса для моделей
Base = declarative_base()

# 3. Определение модели User
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    date_of_birth = Column(Date)

    def __repr__(self):
        return f"<User(id={self.id}, name={self.name}, email={self.email})>"

# 4. Создание сессии для работы с БД
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
session = SessionLocal()

try:
    # 5. Выполнение запроса для получения данных о годах рождения
    # Извлекаем год из даты рождения пользователей
    result = session.query(
        func.extract('year', User.date_of_birth).label('birth_year'),
        func.count().label('num_users')
    ).group_by('birth_year').order_by('birth_year').all()

    # Преобразуем результат в DataFrame для удобства
    df = pd.DataFrame(result, columns=['birth_year', 'num_users'])

    # 6. Построение линейного графика
    plt.figure(figsize=(10, 6))
    plt.plot(df['birth_year'], df['num_users'], marker='o',
             color='skyblue', linestyle='-', linewidth=2)

    # 7. Настройка графика
    plt.title('Распределение пользователей по годам рождения')
    plt.xlabel('Год рождения')
    plt.ylabel('Количество пользователей')

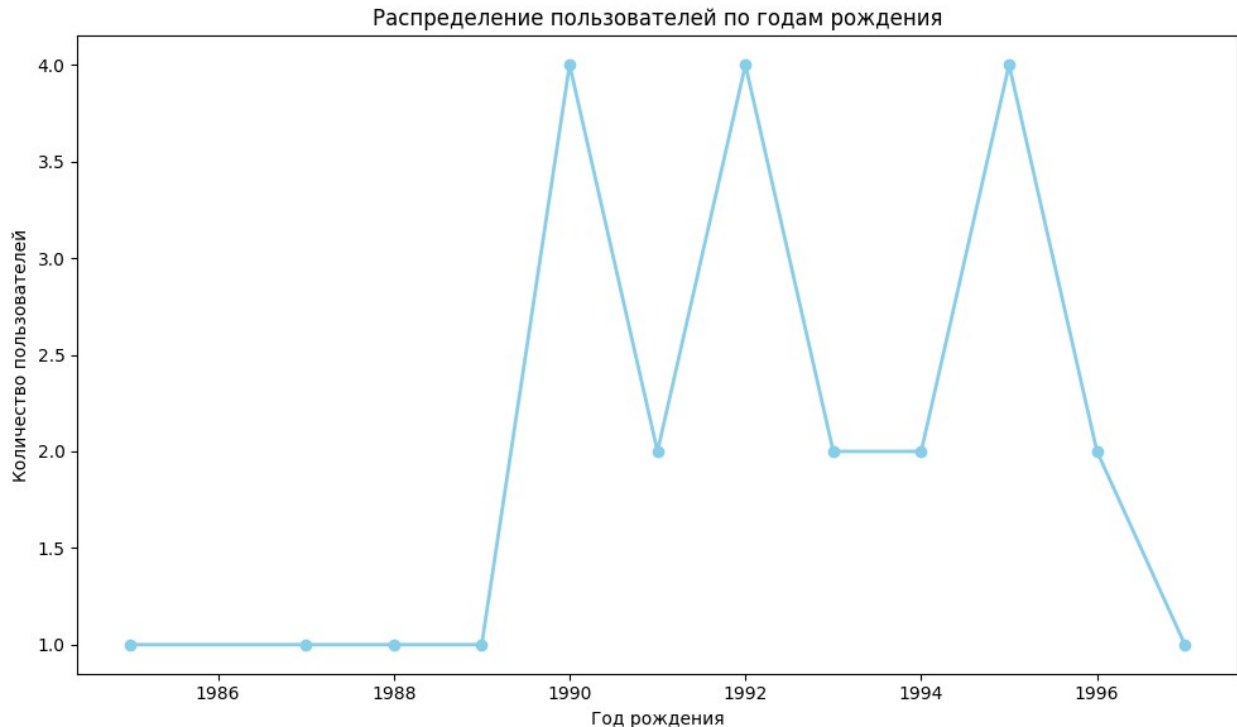
    # Покажем график
    plt.tight_layout()
    plt.show()

except UnicodeDecodeError as e:
    print(f"Ошибка декодирования: {e}")
except Exception as e:

```

```
print(f"Произошла ошибка: {e}")
```

```
# 8. Закрытие сессии  
session.close()
```



Наконец, добавим значение количества пользователей самой первой группы (самые старшие пользователи) и проведем линию среднего значения.

```
import pandas as pd  
import matplotlib.pyplot as plt  
from sqlalchemy import create_engine, Column, Integer, String, Date,  
func  
from sqlalchemy.ext.declarative import declarative_base  
from sqlalchemy.orm import sessionmaker  
import psycopg2
```

```
# 1. Подключение к базе данных PostgreSQL
```

```
DATABASE_URL =  
"postgresql://postgres:Arseuiy228@localhost:5432/postgres"
```

```
# Попробуем установить кодировку при подключении
```

```
conn = psycopg2.connect(DATABASE_URL)  
conn.set_client_encoding('LATIN1') # Используем LATIN1 или windows-  
1251 для возможных проблем с кодировкой
```

```
# Создание движка подключения
```

```
engine = create_engine(DATABASE_URL)
```



```

# 2. Определение базового класса для моделей
Base = declarative_base()

# 3. Определение модели User
class User(Base):
    __tablename__ = 'users'

    id = Column(Integer, primary_key=True)
    name = Column(String, nullable=False)
    email = Column(String, unique=True, nullable=False)
    date_of_birth = Column(Date)

    def __repr__(self):
        return f"<User(id={self.id}, name={self.name}, email={self.email})>"

# 4. Создание сессии для работы с БД
SessionLocal = sessionmaker(autocommit=False, autoflush=False, bind=engine)
session = SessionLocal()

try:
    # 5. Выполнение запроса для получения данных о годах рождения
    # Извлекаем год из даты рождения пользователей
    result = session.query(
        func.extract('year', User.date_of_birth).label('birth_year'),
        func.count().label('num_users')
    ).group_by('birth_year').order_by('birth_year').all()

    # Преобразуем результат в DataFrame для удобства
    df = pd.DataFrame(result, columns=['birth_year', 'num_users'])

    # 6. Построение линейного графика
    plt.figure(figsize=(10, 6))
    plt.plot(df['birth_year'], df['num_users'], marker='o',
             color='skyblue', linestyle='-', linewidth=2)

    # 7. Добавление линии среднего значения
    mean_value = df['num_users'].mean()
    plt.axhline(mean_value, color='lightgray', linestyle='--',
                label=f'Среднее: {mean_value:.2f}')

    # 8. Выделение первого и последнего значения
    first_value_row = df.iloc[0]
    last_value_row = df.iloc[-1]

    # Отображение первого значения
    plt.scatter(first_value_row['birth_year'],
                first_value_row['num_users'], color='green', zorder=5)

```

```

# Сдвиг текста по оси X и Y для первого значения
plt.text(first_value_row['birth_year'],
first_value_row['num_users'] + 0.05,
        f"Старший: {first_value_row['num_users']}",
color='green', fontsize=12, ha='left', va='bottom')

# 9. Настройка графика
plt.title('Распределение пользователей по годам рождения')
plt.xlabel('Год рождения')
plt.ylabel('Количество пользователей')

# 10. Легенда
plt.legend()

# Покажем график
plt.tight_layout()
plt.show()

except UnicodeDecodeError as e:
    print(f"Ошибка декодирования: {e}")
except Exception as e:
    print(f"Произошла ошибка: {e}")

# 11. Закрытие сессии
session.close()

```

