Министерство образования и науки Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский Томский политехнический Университет»

****

самостоятельная работа № 6

курс:

# «Базы данных»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Исполнитель:** |  | | | | |
| студент группы | 0В21ДзебанА.А |  | ФИО студента |  | 26.12.24 |
|  |  |  |  |  |  |
| **Руководитель:** |  | | | | |
| преподаватель |  |  | Полищук В.Ю. |  |  |
|  |  |  |  |  |  |

Томск – 2024

Цель работы: Обрести навыки работы с базами данных.

Задание:

Создать программу для работы с базой данных. База данных должна содержать не менее 5 связанных таблиц. Реализовать вывод данных из созданной базы данных, запросы должны выбирать данные из нескольких таблиц. Допускается данные для разрабатываемой базы данных сгенерировать с помощью модуля «random», либо использовать из сторонних источников. Применить ООП стиль программирования.

Ход работы:

1. Импорт необходимых библиотек:

import sqlite3

from datetime import datetime, timedelta

import random

1. Создадим класс Database, содержащий несколько методов:
2. Создание таблиц
3. Вставка данных
4. Вывод сводной таблицы продавцов, производителей и продуктов

import sqlite3

from datetime import datetime, timedelta

import random

class Database:

    def \_\_init\_\_(self, db\_name):

        self.conn = sqlite3.connect(db\_name)

        self.cursor = self.conn.cursor()

    def create\_tables(self):

        self.cursor.execute('''

            CREATE TABLE IF NOT EXISTS Manufacturers (

                manufacturer\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                name TEXT NOT NULL,

                country TEXT

            )

        ''')

        self.cursor.execute('''

            CREATE TABLE IF NOT EXISTS Suppliers (

                supplier\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                name TEXT NOT NULL,

                address TEXT

            )

        ''')

        self.cursor.execute('''

            CREATE TABLE IF NOT EXISTS Products (

                product\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                name TEXT NOT NULL,

                manufacturer\_id INTEGER,

                supplier\_id INTEGER,

                release\_year INTEGER,

                FOREIGN KEY (manufacturer\_id) REFERENCES Manufacturers (manufacturer\_id),

                FOREIGN KEY (supplier\_id) REFERENCES Suppliers (supplier\_id)

            )

        ''')

        self.cursor.execute('''

            CREATE TABLE IF NOT EXISTS Customers (

                customer\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                name TEXT NOT NULL,

                address TEXT,

                phone\_number TEXT

            )

        ''')

        self.cursor.execute('''

            CREATE TABLE IF NOT EXISTS Purchases (

                purchase\_id INTEGER PRIMARY KEY AUTOINCREMENT,

                product\_id INTEGER,

                customer\_id INTEGER,

                purchase\_date DATE,

                warranty\_end\_date DATE,

                FOREIGN KEY (product\_id) REFERENCES Products (product\_id),

                FOREIGN KEY (customer\_id) REFERENCES Customers (customer\_id)

            )

        ''')

        self.conn.commit()

    def insert\_sample\_data(self):

        # Вставка данных в таблицу Manufacturers

        manufacturers = [('Samsung', 'Корея'), ('Apple', 'США'), ('Xiaomi', 'Китай')]

        self.cursor.executemany('INSERT INTO Manufacturers (name, country) VALUES (?, ?)', manufacturers)

        # Вставка данных в таблицу Suppliers

        suppliers = [('ElectroWorld', 'New York'), ('TechLand', 'Los Angeles'), ('GadgetHub', 'Chicago')]

        self.cursor.executemany('INSERT INTO Suppliers (name, address) VALUES (?, ?)', suppliers)

        # Вставка данных в таблицу Products

        products = [('Galaxy S21', 1, 1, 2021), ('iPhone 13', 2, 2, 2021), ('Mi 11', 3, 3, 2021)]

        self.cursor.executemany('INSERT INTO Products (name, manufacturer\_id, supplier\_id, release\_year) VALUES (?, ?, ?, ?)', products)

        # Вставка данных в таблицу Customers

        customers = [('Илья Зуев', 'New York', '7234567890'), ('Дзебан Арсений', 'Los Angeles', '7987654321')]

        self.cursor.executemany('INSERT INTO Customers (name, address, phone\_number) VALUES (?, ?, ?)', customers)

        # Вставка данных в таблицу Purchases

        for i in range(1, 4):

            purchase\_date = datetime.now() - timedelta(days=random.randint(1, 30))

            warranty\_end\_date = purchase\_date + timedelta(days=random.randint(365, 730))

            self.cursor.execute('INSERT INTO Purchases (product\_id, customer\_id, purchase\_date, warranty\_end\_date) VALUES (?, ?, ?, ?)',

                                (i, random.randint(1, 2), purchase\_date.strftime('%Y-%m-%d'), warranty\_end\_date.strftime('%Y-%m-%d')))

        self.conn.commit()

    def get\_products\_with\_manufacturers\_and\_suppliers(self):

        self.cursor.execute('''

            SELECT Products.name, Manufacturers.name, Suppliers.name

            FROM Products

            JOIN Manufacturers ON Products.manufacturer\_id = Manufacturers.manufacturer\_id

            JOIN Suppliers ON Products.supplier\_id = Suppliers.supplier\_id

        ''')

        return self.cursor.fetchall()

    def get\_purchases\_with\_products\_and\_customers(self):

        self.cursor.execute('''

            SELECT Products.name, Customers.name, Purchases.purchase\_date, Purchases.warranty\_end\_date

            FROM Purchases

            JOIN Products ON Purchases.product\_id = Products.product\_id

            JOIN Customers ON Purchases.customer\_id = Customers.customer\_id

        ''')

        return self.cursor.fetchall()

    def close(self):

        self.conn.close()

1. Создадим базу и выведем результат некоторых запросов:

db = Database('electronics\_store.db')

db.create\_tables()

db.insert\_sample\_data()

print("Товары с производителями и поставщиками:")

for product in db.get\_products\_with\_manufacturers\_and\_suppliers():

    print(product)

print("\nПокупки с товарами и клиентами:")

for purchase in db.get\_purchases\_with\_products\_and\_customers():

    print(purchase)

db.close()

Вывод:

Товары с производителями и поставщиками:

('Galaxy S21', 'Samsung', 'ElectroWorld')

('iPhone 13', 'Apple', 'TechLand')

('Mi 11', 'Xiaomi', 'GadgetHub')

('Galaxy S21', 'Samsung', 'ElectroWorld')

('iPhone 13', 'Apple', 'TechLand')

('Mi 11', 'Xiaomi', 'GadgetHub')

Покупки с товарами и клиентами:

('Galaxy S21', 'Jane Smith', '2024-12-10', '2026-05-23')

('iPhone 13', 'Jane Smith', '2024-12-19', '2026-01-20')

('Mi 11', 'Jane Smith', '2024-12-24', '2026-06-06')

('Galaxy S21', 'John Doe', '2024-12-10', '2026-09-06')

('iPhone 13', 'Jane Smith', '2024-12-23', '2026-02-08')

('Mi 11', 'John Doe', '2024-12-23', '2026-01-20')

**Вывод:**

Были исследованы возможности работы с реляционными базами данных в python.