НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

**Кафедра системного програмування та спеціалізованих комп’ютерних систем**

**Лабораторна робота №2**

з дисципліни

**«Бази даних та засоби управління»**

**ТЕМА: «СТВОРЕННЯ ДОДАТКУ БАЗИ ДАНИХ, ОРІЄНТОВАНОГО НА ВЗАЄМОДІЮ З СУБД POSTGRESQL»**

Виконав: студент ІІІ курсу

ФПМ групи КВ-04

Оніщук А.О.

Київ – 2022

*Метою роботи* є здобуття вмінь програмування прикладних додатків баз даних PostgreSQL.

*Загальне завдання* роботи полягає у наступному:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.

2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.

3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.

4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Для програмування була використана мова Python i PyCharm Community Edition. В програмі були використані бібліотеки:

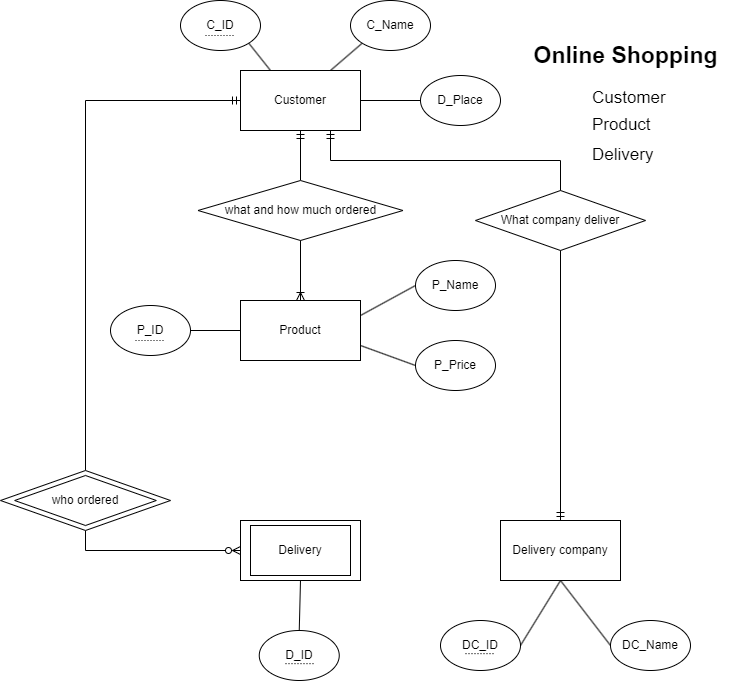
psycopg2 – для управління базою даних

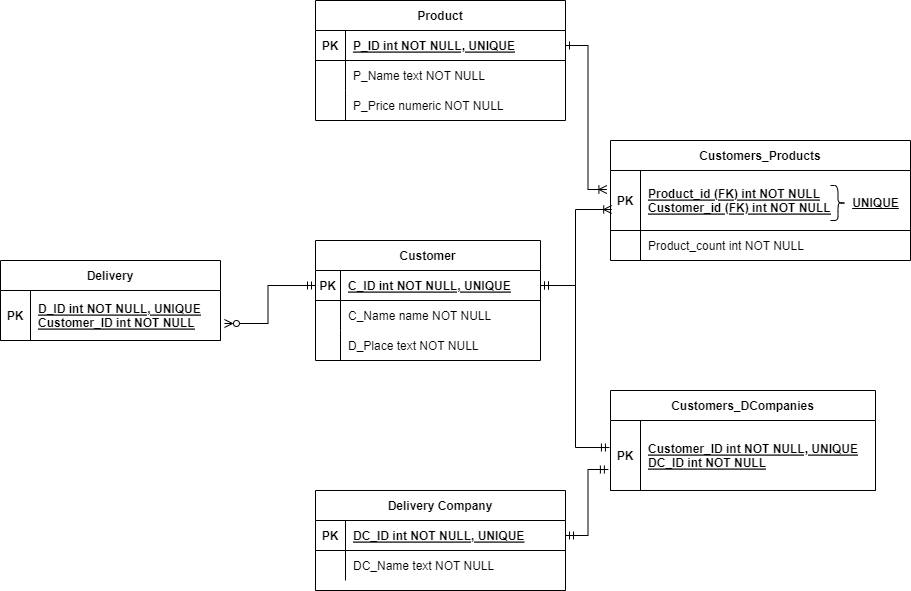
time – для заміру часу

Git-репозиторій: https://github.com/ViDjet-Git/Data-Base-University

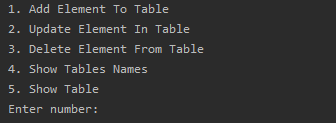
**Діаграма сутність-зв’язок та структура бази даних з лабораторної роботи №1**

*Варіант завдання:* Для завдання була обрана галузь “Інтернет-магазин (Покупець, продукція, доставка, компанія доставки)”. Покупець обирає, що замовити, через яку компанію виконати доставку, і далі звертається до служби доставки, щоб вона виконала його замовлення.



****

**Меню користувача**



**1 пункт – додає один елемент до таблиці**

**2 пункт – обновляє елемент з таблиці**

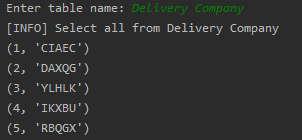
**3 пункт – видаляє елемент з таблиці**

**4 пункт – виводить назви таблиць бази даних**

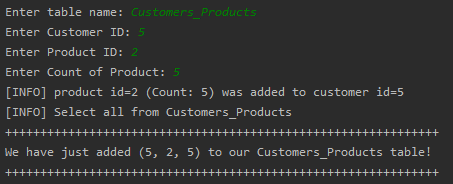
**5 пункт – виводить на екран таблицю з її елементами**

**Пункт 1. Реалізація функцій**

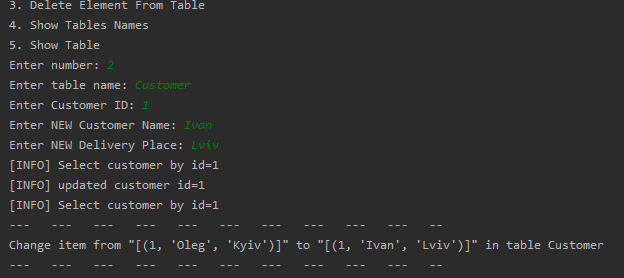
**Операція виведення таблиці**



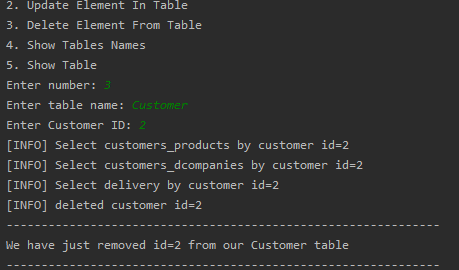
**Операція додавання елементу**



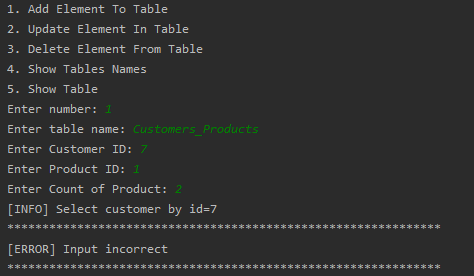
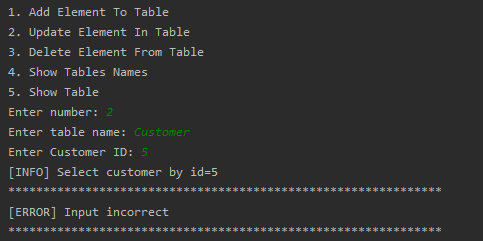
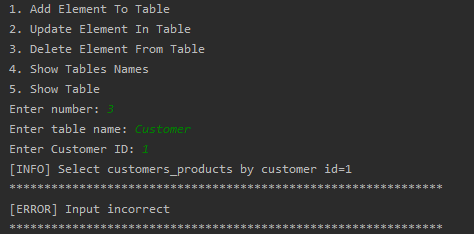
**Редагування елементу**



**Видалення елементу**

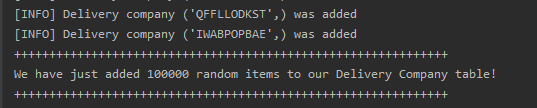


**Помилка при невірному використанні операцій додавання видалення та оновлення**

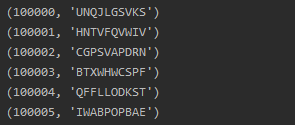


**Пункт 2. Генерація випадкових даних**

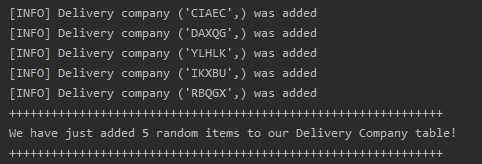
**Додавання 100 000 випадкових елементів**



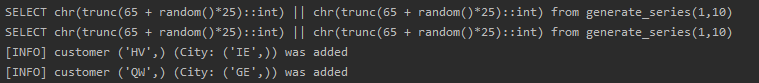
**Вибірка цих елементів(в таблиці до додавання ще було 5 елементів)**



**Генерування 5 випадкових елементів**

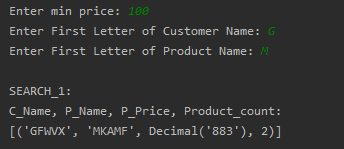


**SQL запити для генерування даних (в нашому випадку два рядки довжиною два символи)**

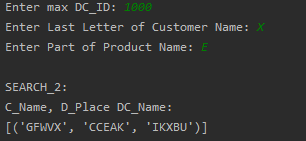


**Пункт 3. Пошукові запити**

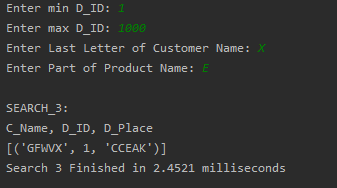
**Пошуковий запит 1 (За мінімальною ціною товару, за першою літерою імені покупця і за першою літерою продукту)**



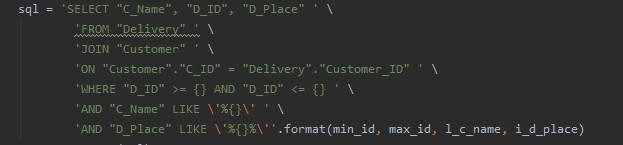
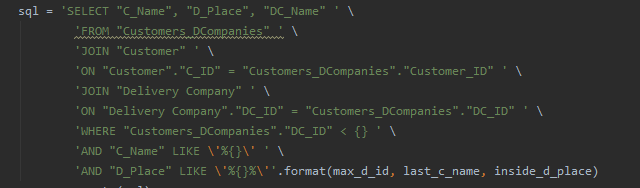
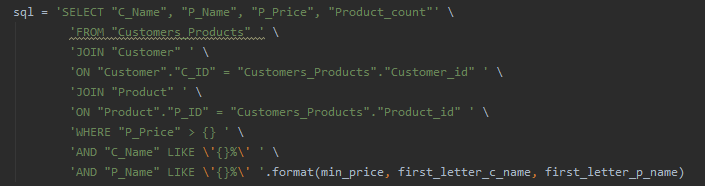
**Пошуковий запит 2 (За максимальним ID компанії доставки, за останньою літерою імені покупця і за частиною назви продукту)**



**Пошуковий запит 3 (В діапазоні ID доставок, за останньою літерою імені покупця і за частиною назви продукту)**



**Генерування SQL запитів**



**Пункт 4. Шаблон програмного коду**

**model.py з описом функцій**

import backend as b  
import time  
  
  
class ModelBasic(object):  
  
 #додавання покупця  
 def add\_new\_customer(self, connection, customerName, deliveryPlace):  
 b.add\_customer(connection, customerName, deliveryPlace)  
  
 #редагування покупця  
 def update\_customer(self, connection, \_id, customerName, deliveryPlace):  
 b.update\_customer(connection, \_id, customerName, deliveryPlace)  
  
 #видалення покупця  
 def delete\_customer(self, connection, \_id):  
 b.delete\_customer(connection, \_id)  
  
 #додання продукту  
 def add\_new\_product(self, connection, productName, productPrice):  
 b.add\_product(connection, productName, productPrice)  
  
 #редагування продукту  
 def update\_product(self, connection, \_id, productName, productPrice):  
 b.update\_product(connection, \_id, productName, productPrice)  
  
 #видалення продукту  
 def delete\_product(self, connection, \_id):  
 b.delete\_product(connection, \_id)  
  
 #додавання компанії  
 def add\_new\_delivery\_company(self, connection, companyName):  
 b.add\_delivery\_company(connection, companyName)  
  
 #редагування компанії  
 def update\_delivery\_company(self, connection, \_id, companyName):  
 b.update\_delivery\_company(connection, \_id, companyName)  
  
 #видалення компанії  
 def delete\_delivery\_company(self, connection, \_id):  
 b.delete\_delivery\_company(connection, \_id)  
  
 #доданя елементу в таблицю Покупець/Продукти  
 def add\_product\_to\_customer(self, connection, customerID, productID, count):  
 b.add\_product\_to\_customer(connection, customerID, productID, count)  
  
 #видалення елементу з таблиці Покупець/Продукти  
 def delete\_product\_from\_customer(self, connection, customer\_id, product\_id):  
 b.delete\_product\_from\_customer(connection, customer\_id, product\_id)  
  
 #доданя елементу в таблицю Покупець/Компанія доставки  
 def add\_delivery\_company\_to\_customer(self, connection, customerID, companyID):  
 b.add\_delivery\_company\_to\_customer(connection, customerID, companyID)  
  
 #редагування елементу в таблиці Покупець/Компанія доставки  
 def update\_delivery\_company\_to\_customer\_by\_customer\_id(self, connection, customerID, companyID):  
 b.update\_delivery\_company\_to\_customer\_by\_customer\_id(connection, customerID, companyID)  
  
 #видалення елементу з таблиці Покупець/Компанія доставки  
 def delete\_delivery\_company\_from\_customer(self, connection, \_id):  
 b.delete\_delivery\_company\_from\_customer(connection, \_id)  
  
 #доданя елементу в таблицю Доставка  
 def add\_delivery(self, connection, customerID):  
 b.add\_delivery(connection, customerID)  
  
 #видалення елементу з таблиці Доставка  
 def delete\_delivery(self, connection, \_id):  
 b.delete\_delivery(connection, \_id)  
  
 #вибірка всіх елементів таблиці  
 def select\_all(self, connection, tableName):  
 return b.select\_all(connection, tableName)  
  
 #вибірка елемента покупця за його id  
 def select\_customer\_by\_id(self, connection, \_id):  
 return b.select\_customer\_by\_id(connection, \_id)  
  
 # вибірка елемента продукту за його id  
 def select\_product\_by\_id(self, connection, \_id):  
 return b.select\_product\_by\_id(connection, \_id)  
  
 # вибірка елемента компанії доставки за його id  
 def select\_delivery\_company\_by\_id(self, connection, \_id):  
 return b.select\_delivery\_company\_by\_id(connection, \_id)  
  
 # вибірка елемента Покупець/Продукт за id покупця  
 def select\_product\_to\_customer\_by\_customer\_id(self, connection, \_id):  
 return b.select\_product\_to\_customer\_by\_customer\_id(connection, \_id)  
  
 # вибірка елемента Покупець/Продукт за id продукту  
 def select\_product\_to\_customer\_by\_product\_id(self, connection, \_id):  
 return b.select\_product\_to\_customer\_by\_product\_id(connection, \_id)  
  
 # вибірка елемента Покупець/Компанія Доставки за id покупця  
 def select\_company\_to\_customer\_by\_customer\_id(self, connection, \_id):  
 return b.select\_company\_to\_customer\_by\_customer\_id(connection, \_id)  
  
 # вибірка елементів Покупець/Компанія Доставки за id компанії доставки  
 def select\_company\_to\_customer\_by\_company\_id(self, connection, \_id):  
 return b.select\_company\_to\_customer\_by\_company\_id(connection, \_id)  
  
 # вибірка елемента Доставки за id  
 def select\_delivery\_by\_id(self, connection, \_id):  
 return b.select\_delivery\_by\_id(connection, \_id)  
  
 # вибірка елемента Доставки за id покупця  
 def select\_delivery\_by\_customer\_id(self, connection, \_id):  
 return b.select\_delivery\_by\_customer\_id(connection, \_id)  
  
 # видалення всієї таблиці  
 def delete\_all\_from\_table(self, connection, tableName):  
 b.delete\_all\_from\_table(connection, tableName)  
  
 # генерування чисел  
 def auto\_gen\_int(self, connection, max\_val, rows\_number):  
 return b.auto\_gen\_int(connection, max\_val, rows\_number)  
  
 # генерування рядків  
 def auto\_gen\_char(self, connection, str\_len, rows\_number):  
 return b.auto\_gen\_char(connection, str\_len, rows\_number)  
  
 # пошук 1 і замір часу  
 def search\_1(self, connection, min\_price, f\_c\_name, f\_p\_name):  
 start\_time = time.perf\_counter()  
 b.search1(connection, min\_price, f\_c\_name, f\_p\_name)  
 end\_time = time.perf\_counter()  
 run\_time = (end\_time - start\_time) \* 1000  
 print(f"Search 1 Finished in {run\_time:.4f} milliseconds")  
  
 # пошук 2 і замір часу  
 def search\_2(self, connection, max\_id, l\_c\_name, i\_d\_place):  
 start\_time = time.perf\_counter()  
 b.search2(connection, max\_id, l\_c\_name, i\_d\_place)  
 end\_time = time.perf\_counter()  
 run\_time = (end\_time - start\_time) \* 1000  
 print(f"Search 1 Finished in {run\_time:.4f} milliseconds")  
  
 # пошук 3 і замір часу  
 def search\_3(self, connection, min\_id, max\_id, l\_c\_name, i\_d\_place):  
 start\_time = time.perf\_counter()  
 b.search3(connection, min\_id, max\_id, l\_c\_name, i\_d\_place)  
 end\_time = time.perf\_counter()  
 run\_time = (end\_time - start\_time) \* 1000  
 print(f"Search 3 Finished in {run\_time:.4f} milliseconds")