BÁO CÁO CÔNG VIỆC

**Công việc số:** 56

**Mô tả công việc:** Đẩy dữ liệu file từ file Excel vào bảng Transaction trong cơ sở dữ liệu và sử đổi mã nguồn lấy dữ liệu giao dịch từ bảng trên.

**Người thực hiện:** Hạ Quang Dũng

**Ngày bắt đầu:** 16/10/2024

**Ngày kết thúc:** 16/10/2024

Mục lục

[**1. Hướng dẫn nhập dữ liệu giao dịch từ file Excel vào cơ sở dữ liệu 2**](#_3gqe6mjuohzo)

[**2. Tạo lớp DatabaseHelper trong Common 9**](#_lo4wo830tjt3)

[**3. Chỉnh sửa file Program 10**](#_8rrwkxlvw8wk)

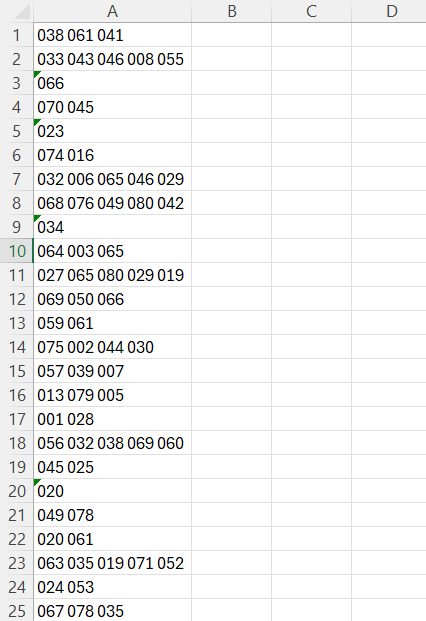
[**4. Kết quả thực hiện 13**](#_cldfzfyekkhr)

# 1. Hướng dẫn nhập dữ liệu giao dịch từ file Excel vào cơ sở dữ liệu

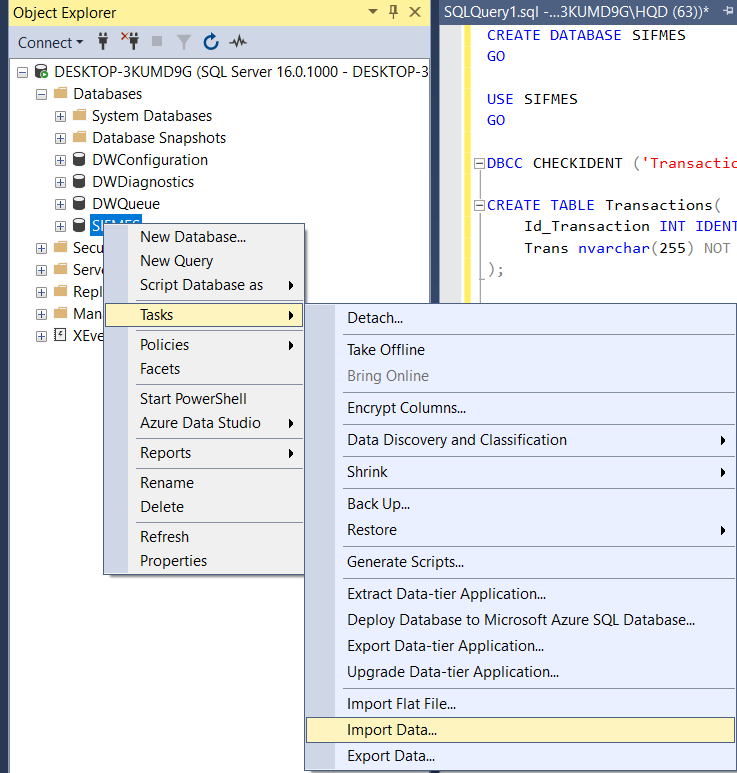
Hiện tại bảng **Transaction** trong CSDL đang có các thuộc tính như sau:

| CREATE TABLE Transactions(  Id\_Transaction INT IDENTITY(1, 1) PRIMARY KEY,  Trans nvarchar(255) NOT NULL, ); |
| --- |

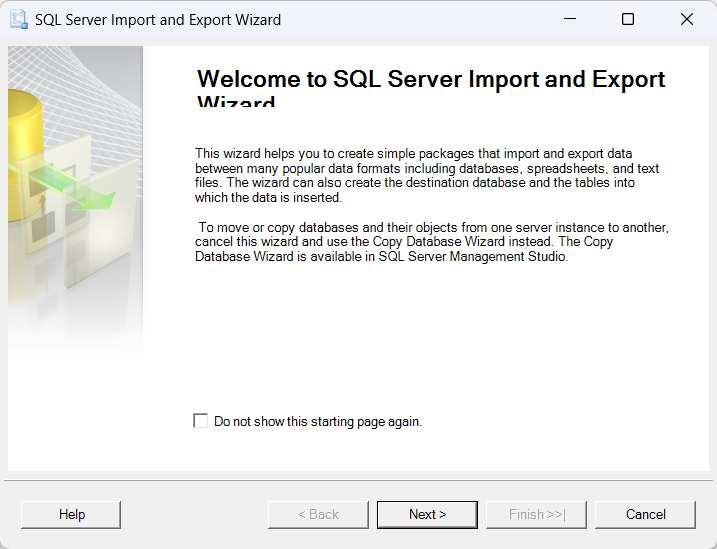
Do cột Id đang để tự tăng, vậy nên dữ liệu trong file Excel chỉ cần 1 cột là cột giao dịch như sau:



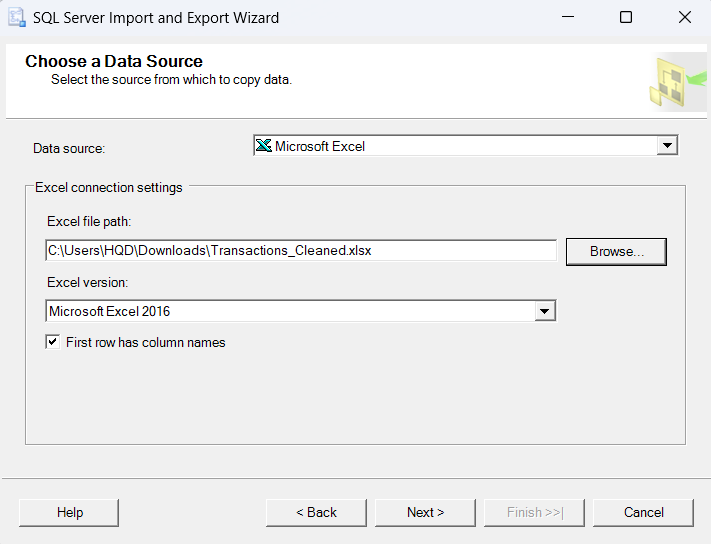
**Bước 1:** Nhấp chuột phải vào **CSDL > Task > Import Data**



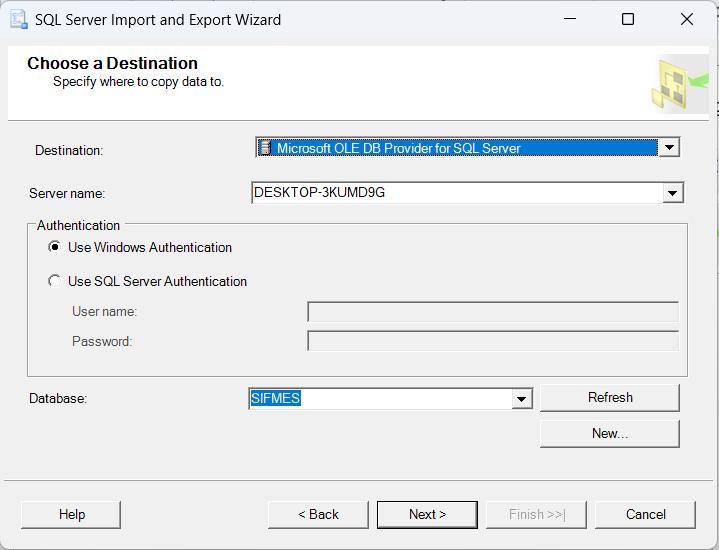
Bước 2: Màn hình **SQL Server Import and Export Wizard** sẽ được bật lên, bấm **Next**



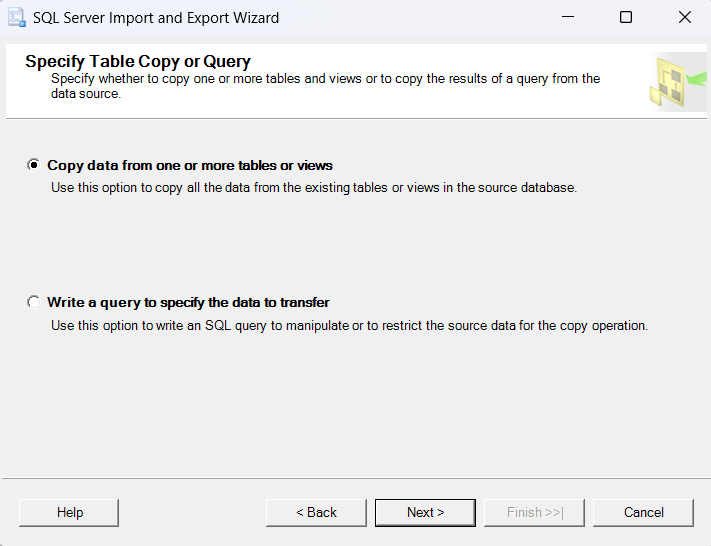
**Bước 3:** Chọn **Data Source**, **Đường dẫn file** cần import, và **phiên bản Excel**, bỏ tích nếu hàng đầu tiên trong file excel là tiêu đề cột.



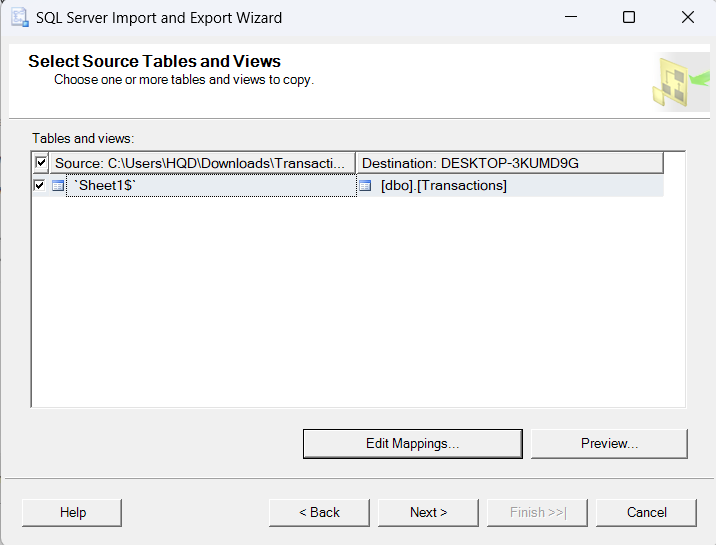
**Bước 4:** Chọn đích đến của dữ liệu như hình bên dưới



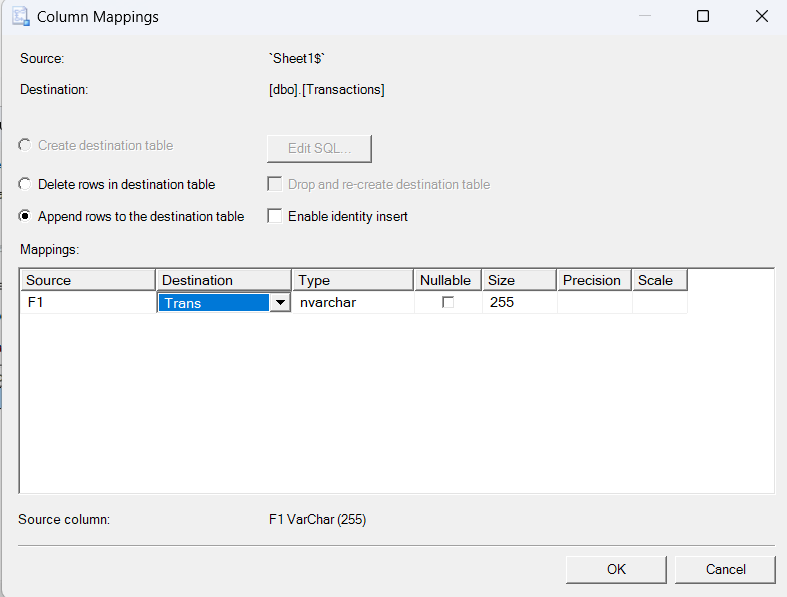
**Bước 5:** Chỉ định sao chép bảng hoặc truy vấn bằng câu lệnh, chọn **Copy data from one or more tables or views**



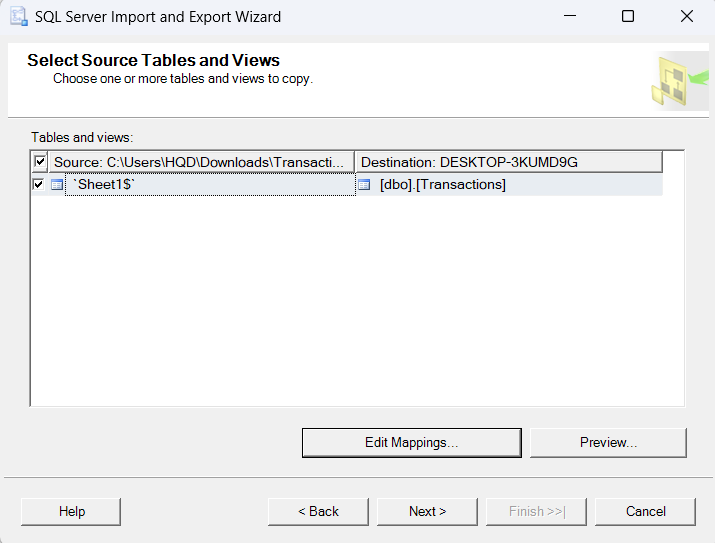
**Bước 6:** Bấm chọn bảng đích và bấm chọn **Edit Mappings…**



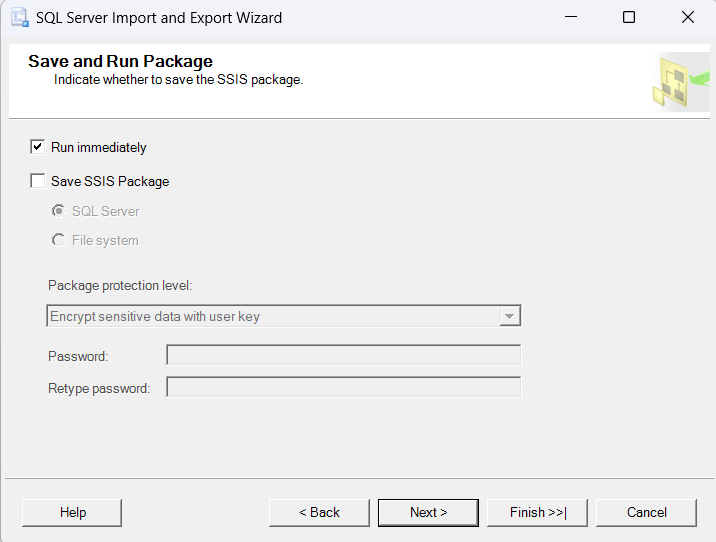
**Bước 7:** Chọn Cột đích cần chèn và bấm **OK**



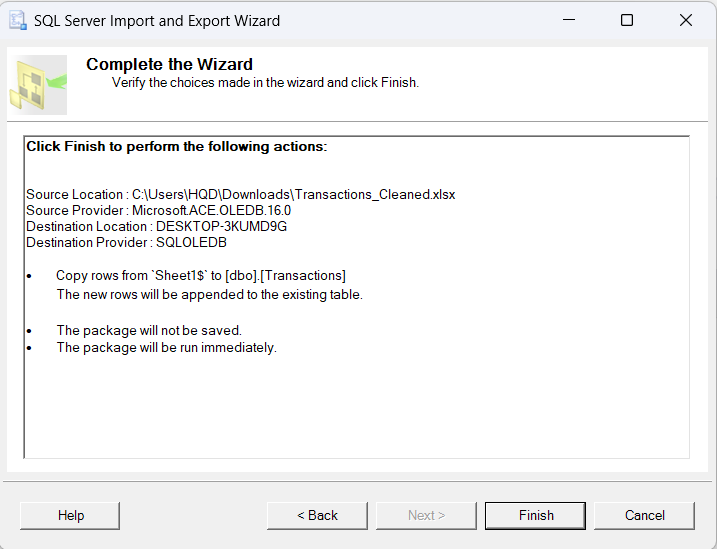
Bước 8: Bấm **Next**



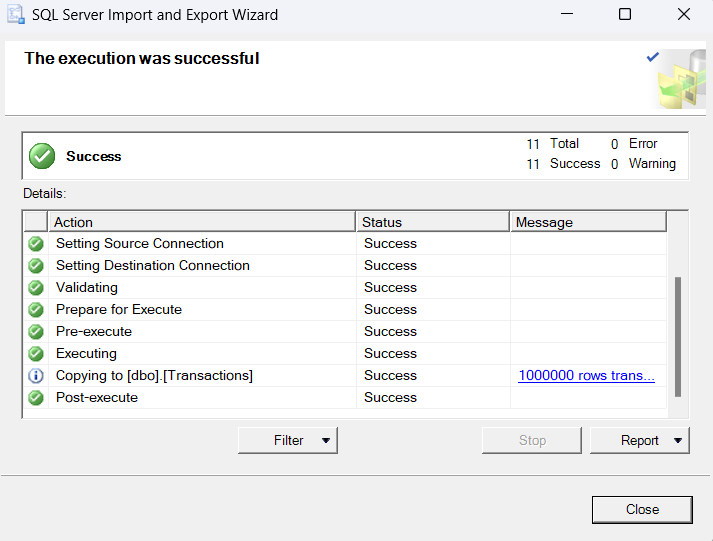
**Bước 9:** Bấm **Next**



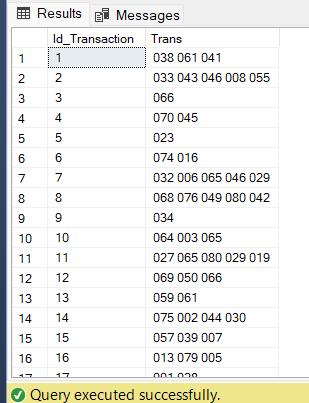
**Bước 10:** Bấm **Finish**



**Bước 11:** Dữ liệu đã được chèn thành công, bấm **Close**



Như vậy, dữ liệu trong CSDL đã được nhập vào thành công



# 2. Tạo lớp DatabaseHelper trong Common

| sing System.Data.SqlClient; using System.Data;  namespace Common {  public static class DatabaseHelper  {  private static string connectionString = $"Data Source={System.Environment.MachineName};Initial Catalog=SIFMES;Integrated Security=True;";   // Phương thức để mở kết nối đến cơ sở dữ liệu  public static SqlConnection GetConnection()  {  return new SqlConnection(connectionString);  }   // Phương thức để thực thi một câu lệnh SQL và trả về một DataTable  public static DataTable ExecuteQuery(string query)  {  using (var connection = GetConnection())  {  connection.Open();  using (var command = new SqlCommand(query, connection))  {  using (var adapter = new SqlDataAdapter(command))  {  var dataTable = new DataTable();  adapter.Fill(dataTable);  return dataTable;  }  }  }  }   // Phương thức để thực thi câu lệnh không trả về dữ liệu (INSERT, UPDATE, DELETE)  public static void ExecuteNonQuery(string query)  {  using (var connection = GetConnection())  {  connection.Open();  using (var command = new SqlCommand(query, connection))  {  command.ExecuteNonQuery();  }  }  }   } } |
| --- |

# 3. Chỉnh sửa file Program

Tiến hành thay đổi các phần như sau:

* Dữ liệu sẽ truy vấn từ CSDL
* Đọc tham số từ dòng lệnh để truy vấn số bản ghi (giao dịch), ngưỡng hỗ trợ tối thiểu

| using Algorithms.Apriori; using Algorithms.FPGrowth; using Common; using System; using System.Collections.Generic; using System.Data; using System.Diagnostics; using System.Text;  namespace Main {  internal class Program  {  static void Main(string[] args)  {  Console.OutputEncoding = Encoding.UTF8;   Console.Write("Nhập số lượng giao dịch muốn lấy từ CSDL: ");  int transactionCount = int.Parse(Console.ReadLine());   List<string> transactions = FetchTransactionsFromDatabase(transactionCount);    if (transactions.Count == 0)  {  Console.WriteLine("Sử dụng giao dịch mẫu");  transactions = new List<string>  {  "f a c d g i m p",  "a b c f l m o",  "b f h j o",  "b c k s p",  "a f c e l p m n",  };  }   Console.Write($"Ngưỡng hỗ trợ tối thiểu (min = 2): ");  int supportThreshold = int.Parse(Console.ReadLine());  var transactionsPreprocessed = Helper.PreprocessDatabase(transactions);   string choice = GetUserChoice();  if (choice == "3") return; // Thoát chương trình nếu chọn 3   RunAlgorithm(choice, transactionsPreprocessed, supportThreshold);  }   static List<string> FetchTransactionsFromDatabase(int transactionCount)  {  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();   List<string> transactions = new List<string>();   try  {  string query = $"SELECT TOP {transactionCount} Trans FROM Transactions"; // Điều chỉnh theo cấu trúc bảng  DataTable result = DatabaseHelper.ExecuteQuery(query);   foreach (DataRow row in result.Rows)  {  string transaction = row["Trans"].ToString();  transactions.Add(transaction);  }   Console.WriteLine("Danh sách giao dịch lấy từ cơ sở dữ liệu:");  transactions.ForEach(Console.WriteLine);  }  catch (Exception ex)  {  Console.WriteLine("Có lỗi xảy ra khi kết nối CSDL: " + ex.Message);  }  stopwatch.Stop();  Console.WriteLine($"\n\nThời gian lấy dữ liệu: {stopwatch.Elapsed}");  return transactions;  }   static string GetUserChoice()  {  //Console.Clear();  Console.WriteLine("Chọn thuật toán để chạy:");  Console.WriteLine("1. FP-Growth");  Console.WriteLine("2. Apriori");  Console.WriteLine("3. Thoát chương trình");  Console.Write("Nhập lựa chọn của bạn (1, 2, hoặc 3): ");  return Console.ReadLine();  }   static void RunAlgorithm(string choice, List<string> transactions, int supportThreshold)  {  List<List<string>> frequentItemSets = null;  Stopwatch stopwatch = new Stopwatch();  stopwatch.Start();   switch (choice)  {  case "1":  Console.WriteLine("\nChạy thuật toán FP-Growth...");  frequentItemSets = new FPGrowth(transactions, supportThreshold).GetFrequentItemsets();  break;   case "2":  Console.WriteLine("\nChạy thuật toán Apriori...");  frequentItemSets = new Apriori(transactions, supportThreshold).GetFrequentItemSet();  break;   default:  Console.WriteLine("Lựa chọn không hợp lệ. Kết thúc chương trình.");  return;  }   Helper.PrintFrequentItemsets(frequentItemSets, transactions, supportThreshold);   var rules = Helper.GetAssociationRules(frequentItemSets, transactions, supportThreshold);  Helper.PrintAssociationRule(rules);   stopwatch.Stop();  Console.WriteLine($"\n\nThời gian thực thi: {stopwatch.Elapsed}");   Console.WriteLine("\nNhấn phím bất kỳ để tiếp tục...");  Console.ReadKey();  }  } } |
| --- |

# 4. Kết quả thực hiện

| **Số lượng bộ dữ liệu** | **Thời gian thực hiện** |
| --- | --- |
| 500 giao dịch | 00:00:00.6615284 |
| 1000 giao dịch | 00:00:00.6652973 |
| 1500 giao dịch | 00:00:00.6641220 |
| 2000 giao dịch | 00:00:00.6731115 |
| 3000 giao dịch | 00:00:00.6784834 |
| 5000 giao dịch | 00:00:00.7504407 |
| 7000 giao dịch | 00:00:00.7411897 |
| 10000 giao dịch | 00:00:00.7940717 |
| 15000 giao dịch | 00:00:00.7709709 |
| 20000 giao dịch | 00:00:00.7292192 |
| 30000 giao dịch | 00:00:00.7085400 |
| 50000 giao dịch | 00:00:00.7714136 |
| 100000 giao dịch | 00:00:00.8254298 |
| 500000 giao dịch | 00:00:01.8275751 |
| 1000000 giao dịch | 00:00:02.4909611 |

Nhận xét:

1. **Thời gian tăng theo số lượng giao dịch**, nhưng không tăng tuyến tính và ổn định.
2. **Hiệu suất ổn định** với các tập dữ liệu nhỏ và trung bình (500 - 50.000 giao dịch), thời gian phản hồi không chênh lệch nhiều.
3. **Giảm hiệu suất với tập dữ liệu lớn**: Thời gian tăng đáng kể khi xử lý 500.000 - 1.000.000 giao dịch.
4. **Có hiện tượng tăng thời gian không đều**, có thể do cơ chế tối ưu hóa hoặc điều kiện hệ thống.