# ДОДАТОК А
## Скрипт бази даних

```
USE [master]
GO
/****** Object:  Database [DB]    Script Date: 03.06.2024 17:54:05 ******/
CREATE DATABASE [DB]
 CONTAINMENT = NONE
 ON  PRIMARY
( NAME = N'DB', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\DB.mdf' , SIZE = 8192KB , MAXSIZE =
UNLIMITED, FILEGROWTH = 65536KB )
 LOG ON
( NAME = N'DB_log', FILENAME = N'C:\Program Files\Microsoft SQL
Server\MSSQL16.MSSQLSERVER\MSSQL\DATA\DB_log.ldf' , SIZE = 8192KB , MAXSIZE =
2048GB , FILEGROWTH = 65536KB )
 WITH CATALOG_COLLATION = DATABASE_DEFAULT, LEDGER = OFF
GO
GO
ALTER DATABASE [DB] SET QUERY_STORE (OPERATION_MODE = READ_WRITE,
CLEANUP_POLICY = (STALE_QUERY_THRESHOLD_DAYS = 30),
DATA_FLUSH_INTERVAL_SECONDS = 900, INTERVAL_LENGTH_MINUTES = 60,
MAX_STORAGE_SIZE_MB = 1000, QUERY_CAPTURE_MODE = AUTO,
SIZE_BASED_CLEANUP_MODE = AUTO, MAX_PLANS_PER_QUERY = 200,
WAIT_STATS_CAPTURE_MODE = ON)
GO
USE [DB]
GO
/****** Object:  Table [dbo].[Category]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Category](
	[CategoryId] [int] IDENTITY(1,1) NOT NULL,
	[CategoryName] [nvarchar](100) NULL,
	[Description] [nvarchar](max) NULL,
PRIMARY KEY CLUSTERED
(
	[CategoryId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Employee]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Employee](
```

```
	[EmployeeId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](50) NULL,
	[LastName] [nvarchar](50) NULL,
	[Position] [nvarchar](100) NULL,
	[Email] [nvarchar](150) NULL,
	[Phone] [nvarchar](20) NULL,
PRIMARY KEY CLUSTERED
(
	[EmployeeId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Logs]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Logs](
	[LogsId] [int] IDENTITY(1,1) NOT NULL,
	[UsersId] [int] NULL,
	[EventNameShow] [nvarchar](max) NULL,
	[EventDate] [datetime] NULL,
	[UsersName] [nvarchar](250) NULL,
PRIMARY KEY CLUSTERED
(
	[LogsId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Tool]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Tool](
	[ToolId] [int] IDENTITY(1,1) NOT NULL,
	[ToolName] [nvarchar](100) NULL,
	[Description] [nvarchar](max) NULL,
	[ToolNumber] [nvarchar](20) NULL,
	[ToolPrice] [float] NULL,
	[CategoryId] [int] NULL,
	[ToolStatusId] [int] NULL,
PRIMARY KEY CLUSTERED
(
	[ToolId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
```

```
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[ToolMaintenance]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ToolMaintenance](
	[MaintenanceId] [int] IDENTITY(1,1) NOT NULL,
	[ToolId] [int] NULL,
	[MaintenanceDate] [datetime] NULL,
	[Description] [nvarchar](max) NULL,
	[Cost] [float] NULL,
	[Status] [bit] NULL,
PRIMARY KEY CLUSTERED
(
	[MaintenanceId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[ToolTransaction]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[ToolTransaction](
	[ToolTransactionId] [int] IDENTITY(1,1) NOT NULL,
	[ToolId] [int] NULL,
	[EmployeeId] [int] NULL,
	[TransactionDate] [datetime] NULL,
	[ReturnDate] [datetime] NULL,
	[Description] [nvarchar](max) NULL,
	[Status] [bit] NULL,
PRIMARY KEY CLUSTERED
(
	[ToolTransactionId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
/****** Object:  Table [dbo].[Users]    Script Date: 03.06.2024 17:54:05 ******/
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO
CREATE TABLE [dbo].[Users](
	[UsersId] [int] IDENTITY(1,1) NOT NULL,
	[FirstName] [nvarchar](100) NULL,
	[LastName] [nvarchar](100) NULL,
```

```
	[UsersName] [nvarchar](100) NULL,
	[UsersPassword] [nvarchar](200) NULL,
	[RoleId] [int] NULL,
	[Description] [nvarchar](max) NULL,
	[Email] [nvarchar](200) NULL,
PRIMARY KEY CLUSTERED
(
	[UsersId] ASC
)WITH (PAD_INDEX = OFF, STATISTICS_NORECOMPUTE = OFF, IGNORE_DUP_KEY =
OFF, ALLOW_ROW_LOCKS = ON, ALLOW_PAGE_LOCKS = ON,
OPTIMIZE_FOR_SEQUENTIAL_KEY = OFF) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO
USE [master]
GO
ALTER DATABASE [DB] SET  READ_WRITE
GO
```

# ДОДАТОК Б
## Лістинги програми

Лістинг 1. Код класу «ActiveTransactionsByCategoryForm»

```
using IssuanceElToolsApp.AppCode;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace IssuanceElToolsApp.Forms.Raports {
  public partial class ActiveTransactionsByCategoryForm : Form {
    RaportBLL _RaportBLL = new RaportBLL();
    List<CategoryActiveTransactions> _CategoryActiveTransactionsList = new
List<CategoryActiveTransactions>();

    public ActiveTransactionsByCategoryForm() {
      InitializeComponent();
      _CategoryActiveTransactionsList = _RaportBLL.GetActiveTransactionsByCategory();
      GetRaport(_CategoryActiveTransactionsList);
    }

    public void GetRaport(List<CategoryActiveTransactions> CategoryActiveTransactionsList) {
      int num = 0;
      if (CategoryActiveTransactionsList.Count > 0) {
        RaportTBox.Text += "-----------------------------------------------------------------------\r\n";
        RaportTBox.Text += String.Format("{0,3}|{1, -50}|{2, 12}|\r\n", "№", "Назва категорії",
"Використання");
        for (int i = 0; i < CategoryActiveTransactionsList.Count(); i++) {
          string raportString = String.Format("{0,3}|{1, -50}|{2, 12}|\r\n",
          ++num,
          CategoryActiveTransactionsList[i].CategoryName,
          CategoryActiveTransactionsList[i].ActiveTransactionsCount);
          RaportTBox.Text += raportString;
        }
        RaportTBox.Text += "-----------------------------------------------------------------------\r\n";
      } else {
        RaportTBox.Text = "Даних не знайдено";
      }
    }


  }
}
```

Лістинг 2. Код класу «IssuedToolsByPeriodForm»

```
using IssuanceElToolsApp.AppCode;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace IssuanceElToolsApp.Forms.Search {
  public partial class IssuedToolsByPeriodForm : Form {
    RaportBLL _RaportBLL = new RaportBLL();
    List<IssuedToolDetails> _IssuedToolDetailsList = new List<IssuedToolDetails>();

    public IssuedToolsByPeriodForm() {
      InitializeComponent();
    }

    private void FormBtn_Click(object sender, EventArgs e) {
      DateTime start = new DateTime(StartDTP.Value.Year, StartDTP.Value.Month,
StartDTP.Value.Day, 0, 0, 0);
      DateTime end = new DateTime(EndDTP.Value.Year, EndDTP.Value.Month, EndDTP.Value.Day,
23, 59, 59);
      _IssuedToolDetailsList = _RaportBLL.GetIssuedToolsByPeriod(start, end);
      GetRaport(_IssuedToolDetailsList);
    }

    public void GetRaport(List<IssuedToolDetails> IssuedToolDetailsList) {
      int num = 0;
      if (IssuedToolDetailsList.Count > 0) {
        RaportTBox.Text = "Звіт за вибраний перід з " + StartDTP.Value.ToShortDateString() + " до "
+ EndDTP.Value.ToShortDateString() + ":\r\n";
        RaportTBox.Text += "----------------------------------------------------------------------------------------------------
----------\r\n";

        RaportTBox.Text += String.Format("{0,3}|{1, -50}|{2, -12}|{3, -50}|\r\n", "№", "Інструмент",
"Дата видачі", "Працівник");
        for (int i = 0; i < IssuedToolDetailsList.Count(); i++) {
          string raportString = String.Format("{0,3}|{1, -50}|{2, -12}|{3, -50}|\r\n",
          ++num,
          IssuedToolDetailsList[i].ToolName,
          IssuedToolDetailsList[i].TransactionDate.ToShortDateString(),
          IssuedToolDetailsList[i].EmployeeName);
          RaportTBox.Text += raportString;
        }
        RaportTBox.Text += "----------------------------------------------------------------------------------------------------
----------\r\n";
      } else {
        RaportTBox.Text = "За вибраний період даних не знайдено!";
      }
```

```
    }

  }
}
```

Лістинг 3. Код класу «ToolMaintenanceCostForm»

```
using IssuanceElToolsApp.AppCode;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace IssuanceElToolsApp.Forms.Raports {
  public partial class ToolMaintenanceCostForm : Form {
    RaportBLL _RaportBLL = new RaportBLL();
    List<ToolMaintenanceCost> _ToolMaintenanceCostList = new List<ToolMaintenanceCost>();

    public ToolMaintenanceCostForm() {
      InitializeComponent();
      _ToolMaintenanceCostList = _RaportBLL.GetTotalMaintenanceCostByTool();
      GetRaport(_ToolMaintenanceCostList);
    }

    public void GetRaport(List<ToolMaintenanceCost> ToolMaintenanceCostList) {
      int num = 0;
      double allSum = 0.0;
      if (ToolMaintenanceCostList.Count > 0) {
        RaportTBox.Text += "--------------------------------------------------------------------------------\r\n";
        RaportTBox.Text += String.Format("{0,3}|{1, -50}|{2, -20}|\r\n", "№", "Назва", "Загальна
сума");
        for (int i = 0; i < ToolMaintenanceCostList.Count(); i++) {
          string raportString = String.Format("{0,3}|{1, -50}|{2, 20}|\r\n",
          ++num,
          ToolMaintenanceCostList[i].ToolName,
          ToolMaintenanceCostList[i].TotalMaintenanceCost);
          allSum += ToolMaintenanceCostList[i].TotalMaintenanceCost;
          RaportTBox.Text += raportString;
        }
        RaportTBox.Text += "--------------------------------------------------------------------------------\r\n";
        RaportTBox.Text += String.Format("{0,55}{1, 20}\r\n", "Загальна сума: ", allSum);
      } else {
        RaportTBox.Text = "Немає інформації";
      }
    }

  }
}
```

Лістинг 4. Код класу «RaportBLL»

```
using IssuanceElToolsApp.Providers;
using System;
using System.Collections.Generic;
using System.Data;
using System.Data.SqlClient;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace IssuanceElToolsApp.AppCode {
  internal class RaportBLL {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];
    ToolProvider _ToolProvider = new ToolProvider();

    public List<Tool> GetRaportForClientFIO(string SearchFIO) {
      int num = 0;
      List<Tool> ToolList = new List<Tool>();
      ToolList = _ToolProvider.GetAllTool();
      List<Tool> searchToolList = new List<Tool>();

      for (int i = 0; i < ToolList.Count(); i++) {
        if (ToolList[i].ToolName.ToLower().Contains(SearchFIO.ToLower()) ||
ToolList[i].ToolNumber.ToLower().Contains(SearchFIO.ToLower())) {
          num++;
          ToolList[i].Number = num;
          searchToolList.Add(ToolList[i]);
        }
      }
      return searchToolList;
    }


    public List<ToolMaintenanceCost> GetTotalMaintenanceCostByTool() {
      string sqlString = @"
      SELECT
        t.ToolId, t.ToolName,
        SUM(tm.Cost) AS TotalMaintenanceCost
      FROM
        Tool t
      JOIN
        ToolMaintenance tm ON t.ToolId = tm.ToolId
      GROUP BY
        t.ToolId, t.ToolName
      ORDER BY
        TotalMaintenanceCost DESC";

      List<ToolMaintenanceCost> listToolMaintenanceCost = new List<ToolMaintenanceCost>();

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
```

```
        using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              ToolMaintenanceCost toolMaintenanceCost = new ToolMaintenanceCost {
                ToolId = Convert.ToInt32(reader["ToolId"]),
                ToolName = reader["ToolName"].ToString(),
                TotalMaintenanceCost = Convert.ToDouble(reader["TotalMaintenanceCost"])
              };
              listToolMaintenanceCost.Add(toolMaintenanceCost);
            }
          }
          conn.Close();
        }
      }

      return listToolMaintenanceCost;
    }

    public List<CategoryActiveTransactions> GetActiveTransactionsByCategory() {
      string sqlString = @"
      SELECT
        c.CategoryId, c.CategoryName,
        COUNT(tt.ToolTransactionId) AS ActiveTransactionsCount
      FROM
        Category c
      JOIN
        Tool t ON c.CategoryId = t.CategoryId
      JOIN
        ToolTransaction tt ON t.ToolId = tt.ToolId
      WHERE
        tt.Status = 0
      GROUP BY
        c.CategoryId, c.CategoryName
      ORDER BY
        ActiveTransactionsCount DESC";

      List<CategoryActiveTransactions> listActiveTransactions = new
List<CategoryActiveTransactions>();

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              CategoryActiveTransactions activeTransactions = new CategoryActiveTransactions {
                CategoryId = Convert.ToInt32(reader["CategoryId"]),
                CategoryName = reader["CategoryName"].ToString(),
                ActiveTransactionsCount = Convert.ToInt32(reader["ActiveTransactionsCount"])
              };
              listActiveTransactions.Add(activeTransactions);
            }
```

```
        }
        conn.Close();
      }
    }

    return listActiveTransactions;
  }

  public List<IssuedToolDetails> GetIssuedToolsByPeriod(DateTime startDate, DateTime endDate)
{
    string sqlString = @"
    SELECT
      t.ToolId, t.ToolName, tt.ToolTransactionId, tt.TransactionDate, e.EmployeeId,
      e.FirstName + ' ' + e.LastName AS EmployeeName,
      tt.Description
    FROM
      Tool t
    JOIN
      ToolTransaction tt ON t.ToolId = tt.ToolId
    JOIN
      Employee e ON tt.EmployeeId = e.EmployeeId
    WHERE
      tt.TransactionDate BETWEEN @StartDate AND @EndDate
    ORDER BY
      tt.TransactionDate";

    List<IssuedToolDetails> listIssuedTools = new List<IssuedToolDetails>();

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@StartDate", startDate);
        cmd.Parameters.AddWithValue("@EndDate", endDate);
        conn.Open();
        using (SqlDataReader reader = cmd.ExecuteReader()) {
          while (reader.Read()) {
            IssuedToolDetails issuedTool = new IssuedToolDetails {
              ToolId = Convert.ToInt32(reader["ToolId"]),
              ToolName = reader["ToolName"].ToString(),
              ToolTransactionId = Convert.ToInt32(reader["ToolTransactionId"]),
              TransactionDate = Convert.ToDateTime(reader["TransactionDate"]),
              EmployeeId = Convert.ToInt32(reader["EmployeeId"]),
              EmployeeName = reader["EmployeeName"].ToString(),
              Description = reader["Description"].ToString()
            };
            listIssuedTools.Add(issuedTool);
          }
        }
        conn.Close();
      }
    }
```

```
      return listIssuedTools;
    }

  }
}



public class ToolMaintenanceCost {
  public int ToolId { get; set; }          // Ідентифікатор інструменту
  public string ToolName { get; set; }      // Назва інструменту
  public double TotalMaintenanceCost { get; set; } // Загальна вартість обслуговування
}


public class CategoryActiveTransactions {
  public int CategoryId { get; set; }              // Ідентифікатор категорії
  public string CategoryName { get; set; }          // Назва категорії
  public int ActiveTransactionsCount { get; set; }  // Кількість активних операцій
}


public class IssuedToolDetails {
  public int ToolId { get; set; }          // Ідентифікатор інструменту
  public string ToolName { get; set; }      // Назва інструменту
  public int ToolTransactionId { get; set; } // Ідентифікатор операції з інструментом
  public DateTime TransactionDate { get; set; } // Дата операції
  public int EmployeeId { get; set; }       // Ідентифікатор працівника
  public string EmployeeName { get; set; }  // Ім'я працівника
  public string Description { get; set; }   // Опис операції
}
```

Лістинг 5. Код класу «GenData»

```
using System;
using System.Collections.Generic;
using System.Data.SqlClient;
using System.Data;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Bogus;

namespace IssuanceElToolsApp.Providers {
  internal class GenData {
    private string _ConnString =
System.Configuration.ConfigurationSettings.AppSettings["CONNECT"];

    public void SeedEmployees(int numberOfRecords) {
      var employeeFaker = new Faker<Employee>("uk")
        .RuleFor(e => e.LastName, f => f.Name.LastName())
        .RuleFor(e => e.FirstName, f => f.Name.FirstName())
        .RuleFor(e => e.Position, f => f.Name.JobTitle())
```

```
        .RuleFor(e => e.Email, f => f.Internet.Email())
        .RuleFor(e => e.Phone, f => f.Phone.PhoneNumber());

      var employees = employeeFaker.Generate(numberOfRecords);

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        conn.Open();
        using (SqlTransaction transaction = conn.BeginTransaction()) {
          try {
            foreach (var employee in employees) {
              InsertEmployee(conn, transaction, employee.LastName, employee.FirstName,
employee.Position, employee.Email, employee.Phone);
            }
            transaction.Commit();
          } catch (Exception) {
            transaction.Rollback();
            throw;
          }
        }
        conn.Close();
      }
    }

    private void InsertEmployee(SqlConnection conn, SqlTransaction transaction, string lastName,
string firstName, string position, string email, string phone) {
      string sqlString = "INSERT INTO Employee (LastName, FirstName, Position, Email, Phone) " +
                         "VALUES (@LastName, @FirstName, @Position, @Email, @Phone)";

      using (SqlCommand cmd = new SqlCommand(sqlString, conn, transaction)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@LastName", lastName);
        cmd.Parameters.AddWithValue("@FirstName", firstName);
        cmd.Parameters.AddWithValue("@Position", position);
        cmd.Parameters.AddWithValue("@Email", email);
        cmd.Parameters.AddWithValue("@Phone", phone);
        cmd.ExecuteNonQuery();
      }
    }

    public void SeedCategories() {
      var categories = new List<(string CategoryName, string Description)> {
      ("Дрилі", "Електричні дрилі для різних видів свердління."),
      ("Пили", "Електричні пили, включаючи циркулярні, шабельні та лобзикові пили."),
      ("Шліфувальні машини", "Електричні шліфувальні машини для різання, шліфування та
полірування."),
      ("Шліфувальні машини", "Електричні шліфувальні машини для згладжування поверхонь."),
      ("Викрутки", "Електричні викрутки для закручування гвинтів."),
      ("Ударні гайковерти", "Електричні ударні гайковерти для важких кріплень."),
      ("Ротарні інструменти", "Електричні ротарні інструменти для детальних робіт та
рукоділля."),
      ("Термофени", "Електричні термофени для зняття фарби та інших застосувань."),
      ("Міксери", "Електричні міксери для змішування фарби, цементу та інших матеріалів."),
```

```
    ("Рубанки", "Електричні рубанки для згладжування дерев'яних поверхонь.")
   };

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
     conn.Open();
     using (SqlTransaction transaction = conn.BeginTransaction()) {
      try {
       foreach (var category in categories) {
        InsertCategory(conn, transaction, category.CategoryName, category.Description);
       }
       transaction.Commit();
      } catch (Exception) {
       transaction.Rollback();
       throw;
      }
     }
     conn.Close();
    }
   }

   private void InsertCategory(SqlConnection conn, SqlTransaction transaction, string categoryName,
string description) {
    string sqlString = "INSERT INTO Category (CategoryName, Description) " +
                "VALUES (@CategoryName, @Description)";

    using (SqlCommand cmd = new SqlCommand(sqlString, conn, transaction)) {
     cmd.CommandType = CommandType.Text;
     cmd.Parameters.AddWithValue("@CategoryName", categoryName);
     cmd.Parameters.AddWithValue("@Description", description);
     cmd.ExecuteNonQuery();
    }
   }



   public void SeedTools() {
    var toolsByCategory = new Dictionary<string, List<(string ToolName, string Description, string
ToolNumber, double ToolPrice)>> {
     {"Дрилі", new List<(string, string, string, double)> {
      ("Дриль ударний", "Ударний дриль для важких робіт", "DR001", 1500.0),
      ("Дриль безударний", "Безударний дриль для легких робіт", "DR002", 1200.0),
      ("Дриль акумуляторний", "Акумуляторний дриль для мобільних робіт", "DR003", 2000.0)
     }},
     {"Пили", new List<(string, string, string, double)> {
      ("Циркулярна пила", "Циркулярна пила для різання деревини", "SA001", 2500.0),
      ("Шабельна пила", "Шабельна пила для важкодоступних місць", "SA002", 2300.0),
      ("Лобзикова пила", "Лобзик для точного різання", "SA003", 1800.0)
     }},
     {"Шліфувальні машини", new List<(string, string, string, double)> {
      ("Кутова шліфувальна машина", "Кутова шліфувальна машина для різання та шліфування",
"GR001", 2200.0),
```

```
      ("Шліфувальна машина по дереву", "Шліфувальна машина для згладжування деревини",
"GR002", 2000.0),
      ("Шліфувальна машина для бетону", "Шліфувальна машина для бетонних поверхонь",
"GR003", 2800.0)
    }},
    {"Викрутки", new List<(string, string, string, double)> {
      ("Електрична викрутка", "Електрична викрутка для закручування гвинтів", "SC001",
1000.0),
      ("Акумуляторна викрутка", "Акумуляторна викрутка для мобільних робіт", "SC002",
1500.0),
      ("Викрутка з змінними насадками", "Викрутка з набором змінних насадок", "SC003",
1300.0)
    }},
    {"Ударні гайковерти", new List<(string, string, string, double)> {
      ("Гайковерт ударний", "Ударний гайковерт для важких кріплень", "IW001", 3000.0),
      ("Акумуляторний гайковерт", "Акумуляторний ударний гайковерт для мобільних робіт",
"IW002", 3500.0)
    }},
    {"Ротарні інструменти", new List<(string, string, string, double)> {
      ("Ротарний інструмент для гравірування", "Інструмент для гравірування та різьби",
"RT001", 2000.0),
      ("Ротарний інструмент для шліфування", "Інструмент для дрібного шліфування", "RT002",
1800.0),
      ("Ротарний інструмент з насадками", "Набір ротарних інструментів з різними насадками",
"RT003", 2500.0)
    }},
    {"Термофени", new List<(string, string, string, double)> {
      ("Термофен будівельний", "Термофен для зняття фарби", "HG001", 1200.0),
      ("Термофен промисловий", "Промисловий термофен для важких робіт", "HG002", 1800.0)
    }},
    {"Міксери", new List<(string, string, string, double)> {
      ("Міксер для фарби", "Міксер для змішування фарби", "MX001", 1400.0),
      ("Міксер для цементу", "Міксер для змішування цементу", "MX002", 2200.0)
    }},
    {"Рубанки", new List<(string, string, string, double)> {
      ("Рубанок електричний", "Електричний рубанок для згладжування деревини", "PL001",
2000.0),
      ("Рубанок з регулюванням глибини", "Електричний рубанок з регулюванням глибини",
"PL002", 2400.0)
    }}
  };

    using (SqlConnection conn = new SqlConnection(_ConnString)) {
      conn.Open();
      using (SqlTransaction transaction = conn.BeginTransaction()) {
        try {
          foreach (var category in toolsByCategory) {
            var categoryId = GetCategoryId(conn, transaction, category.Key);
            foreach (var tool in category.Value) {
              InsertTool(conn, transaction, tool.ToolName, tool.Description, tool.ToolNumber,
tool.ToolPrice, categoryId, 1);
            }
```

```
            }
            transaction.Commit();
          } catch (Exception) {
            transaction.Rollback();
            throw;
          }
        }
        conn.Close();
      }
    }

    private int GetCategoryId(SqlConnection conn, SqlTransaction transaction, string categoryName) {
      string sqlString = "SELECT CategoryId FROM Category WHERE CategoryName =
@CategoryName";
      using (SqlCommand cmd = new SqlCommand(sqlString, conn, transaction)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@CategoryName", categoryName);
        return Convert.ToInt32(cmd.ExecuteScalar());
      }
    }

    private void InsertTool(SqlConnection conn, SqlTransaction transaction, string toolName, string
description, string toolNumber, double toolPrice, int categoryId, int toolStatusId) {
      string sqlString = "INSERT INTO Tool (ToolName, Description, ToolNumber, ToolPrice,
CategoryId, ToolStatusId) " +
                         "VALUES (@ToolName, @Description, @ToolNumber, @ToolPrice, @CategoryId,
@ToolStatusId)";

      using (SqlCommand cmd = new SqlCommand(sqlString, conn, transaction)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@ToolName", toolName);
        cmd.Parameters.AddWithValue("@Description", description);
        cmd.Parameters.AddWithValue("@ToolNumber", toolNumber);
        cmd.Parameters.AddWithValue("@ToolPrice", toolPrice);
        cmd.Parameters.AddWithValue("@CategoryId", categoryId);
        cmd.Parameters.AddWithValue("@ToolStatusId", toolStatusId);
        cmd.ExecuteNonQuery();
      }
    }


    public void SeedToolMaintenance(int numberOfRecords) {
      var toolIds = GetToolIds(); // Отримати список доступних ToolId
      var faker = new Faker<ToolMaintenance>("uk")
        .RuleFor(tm => tm.ToolId, f => f.PickRandom(toolIds))
        .RuleFor(tm => tm.MaintenanceDate, f => f.Date.Past(1, DateTime.Now.AddDays(-1)))
        .RuleFor(tm => tm.Description, f => f.Lorem.Sentence())
        .RuleFor(tm => tm.Cost, f => Math.Round(f.Random.Double(100, 1000), 2));

      var toolMaintenances = faker.Generate(numberOfRecords);

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
```

```
        conn.Open();
        using (SqlTransaction transaction = conn.BeginTransaction()) {
          try {
            foreach (var toolMaintenance in toolMaintenances) {
              InsertToolMaintenance(conn, transaction, toolMaintenance.ToolId,
toolMaintenance.MaintenanceDate, toolMaintenance.Description, toolMaintenance.Cost);
            }
            transaction.Commit();
          } catch (Exception) {
            transaction.Rollback();
            throw;
          }
        }
        conn.Close();
      }
    }

    private List<int> GetToolIds() {
      var toolIds = new List<int>();
      string sqlString = "SELECT ToolId FROM Tool";

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              toolIds.Add(Convert.ToInt32(reader["ToolId"]));
            }
          }
          conn.Close();
        }
      }

      return toolIds;
    }

    private void InsertToolMaintenance(SqlConnection conn, SqlTransaction transaction, int toolId,
DateTime maintenanceDate, string description, double cost) {
      string sqlString = "INSERT INTO ToolMaintenance (ToolId, MaintenanceDate, Description, Cost,
Status) " +
                         "VALUES (@ToolId, @MaintenanceDate, @Description, @Cost, @Status)";

      using (SqlCommand cmd = new SqlCommand(sqlString, conn, transaction)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@ToolId", toolId);
        cmd.Parameters.AddWithValue("@MaintenanceDate", maintenanceDate);
        cmd.Parameters.AddWithValue("@Description", description);
        cmd.Parameters.AddWithValue("@Cost", cost);
        cmd.Parameters.AddWithValue("@Status", 1);
        cmd.ExecuteNonQuery();
      }
    }
```

```
    public void SeedToolTransactions(int numberOfRecords) {
      var toolIds = GetToolIds(); // Отримати список доступних ToolId
      var employeeIds = GetEmployeeIds(); // Отримати список доступних EmployeeId
      var faker = new Faker<ToolTransaction>("uk")
        .RuleFor(tt => tt.ToolId, f => f.PickRandom(toolIds))
        .RuleFor(tt => tt.EmployeeId, f => f.PickRandom(employeeIds))
        .RuleFor(tt => tt.TransactionDate, f => f.Date.Past(1, DateTime.Now.AddDays(-10))) // Дата транзакції до 10 днів назад
        .RuleFor(tt => tt.ReturnDate, (f, tt) => f.Date.Between(tt.TransactionDate, DateTime.Now.AddDays(-1))) // Дата повернення після дати транзакції до вчорашнього дня
        .RuleFor(tt => tt.Description, f => f.Lorem.Sentence());

      var toolTransactions = faker.Generate(numberOfRecords);

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        conn.Open();
        using (SqlTransaction transaction = conn.BeginTransaction()) {
          try {
            foreach (var toolTransaction in toolTransactions) {
              InsertToolTransaction(conn, transaction, toolTransaction.ToolId, toolTransaction.EmployeeId, toolTransaction.TransactionDate, toolTransaction.ReturnDate, toolTransaction.Description);
            }
            transaction.Commit();
          } catch (Exception) {
            transaction.Rollback();
            throw;
          }
        }
        conn.Close();
      }
    }


    private List<int> GetEmployeeIds() {
      var employeeIds = new List<int>();
      string sqlString = "SELECT EmployeeId FROM Employee";

      using (SqlConnection conn = new SqlConnection(_ConnString)) {
        using (SqlCommand cmd = new SqlCommand(sqlString, conn)) {
          conn.Open();
          using (SqlDataReader reader = cmd.ExecuteReader()) {
            while (reader.Read()) {
              employeeIds.Add(Convert.ToInt32(reader["EmployeeId"]));
            }
          }
          conn.Close();
        }
      }
```

```
      return employeeIds;
    }

    private void InsertToolTransaction(SqlConnection conn, SqlTransaction transaction, int toolId, int
employeeId, DateTime transactionDate, DateTime returnDate, string description) {
      string sqlString = "INSERT INTO ToolTransaction (ToolId, EmployeeId, TransactionDate,
ReturnDate, Description, Status) " +
                         "VALUES (@ToolId, @EmployeeId, @TransactionDate, @ReturnDate,
@Description, @Status)";

      using (SqlCommand cmd = new SqlCommand(sqlString, conn, transaction)) {
        cmd.CommandType = CommandType.Text;
        cmd.Parameters.AddWithValue("@ToolId", toolId);
        cmd.Parameters.AddWithValue("@EmployeeId", employeeId);
        cmd.Parameters.AddWithValue("@TransactionDate", transactionDate);
        cmd.Parameters.AddWithValue("@ReturnDate", returnDate);
        cmd.Parameters.AddWithValue("@Description", description);
        cmd.Parameters.AddWithValue("@Status", 1);
        cmd.ExecuteNonQuery();
      }
    }

  }
}
```

ДОДАТОК В
Демонстраційний матеріал у виді презентації

# ДОДАТОК Г
## Відомість кваліфікаційної роботи

| № | Позначення | Найменування | Дод. відомості |
|---|---|---|---|
| | | Текстові документи | |
| | | | |
| 1 | ГЮІК.416616.007ПЗ | Пояснювальна записка | 86 с. |
| | | | |
| | | Додаткові матеріали | |
| | | | |
| | | Скрипт бази даних | |
| | | | |
| | | Лістинги програми | |
| | | | |
| | | Демонстраційний матеріал | У форматі |
| | | | *.pptx |
| | | | 15 слайдів |
| | | | |
| | | | |
| | | | |

| | | | | | ГЮІК.416616.007ПЗ | | | |
|---|---|---|---|---|---|---|---|---|
| Зм. | Арк. | № докум. | Підпис | Дата | | | | |
| Розробив | | Масіч М.В. | | | Розробка системи автоматизації процесу зберігання та видачі електричних інструментів | Літера | Аркуш | Аркушів |
| Перевір. | | Замірець О.М. | | | | | | 1 |
| | | | | | | ХНУРЕ Кафедра КІТАР | | |
| Н. контр. | | Стародубцев М. Г. | | | | | | |
| Затв. | | Невлюдов І.Ш. | | | | | | |