

# **Assignment 2: Database Automation**

**Nidhun Murali – 8981611**

Cloud Development and Operations  
Conestoga College, Waterloo Campus  
Database Automation  
PROG8850 - Spring 2025 - Section 1

6th June 2025

**Repo Link:** <https://github.com/nmrepos/PROG8850Assignment2>

## **Question 1**

**Q1 Link:** <https://github.com/nmrepos/PROG8850Assignment2/tree/question1>

### **SQL Code**

```
-- Creating the database
CREATE DATABASE IF NOT EXISTS projectdb;
USE projectdb;

-- Creating the new table named 'projects'
CREATE TABLE IF NOT EXISTS projects (
    project_id INT AUTO_INCREMENT PRIMARY KEY,
    project_name VARCHAR(255) NOT NULL,
    start_date DATE,
    end_date DATE
);

-- Add a new column 'budget' to the existing 'projects' table
ALTER TABLE projects ADD COLUMN budget DECIMAL(10, 2);
```

### **Python Script to Run the SQL Script**

```
import mysql.connector
import os
import glob

# Connect to MySQL
connection = mysql.connector.connect(
    host=os.environ['MYSQL_HOST'],
    user=os.environ['MYSQL_USER'],
    password=os.environ['MYSQL_PASSWORD'],
    database=os.environ['MYSQL_DATABASE']
)

cursor = connection.cursor()

# Path to the single SQL file
sql_file = './create_projects_table.sql'
print(f"Executing {sql_file}")

with open(sql_file, 'r') as file:
    sql_script = file.read()

# Split script into commands
commands = sql_script.split(';')

for command in commands:
```

```
command = command.strip()
if command:
    print(f"\nRunning: {command}\n")
    cursor.execute(command)

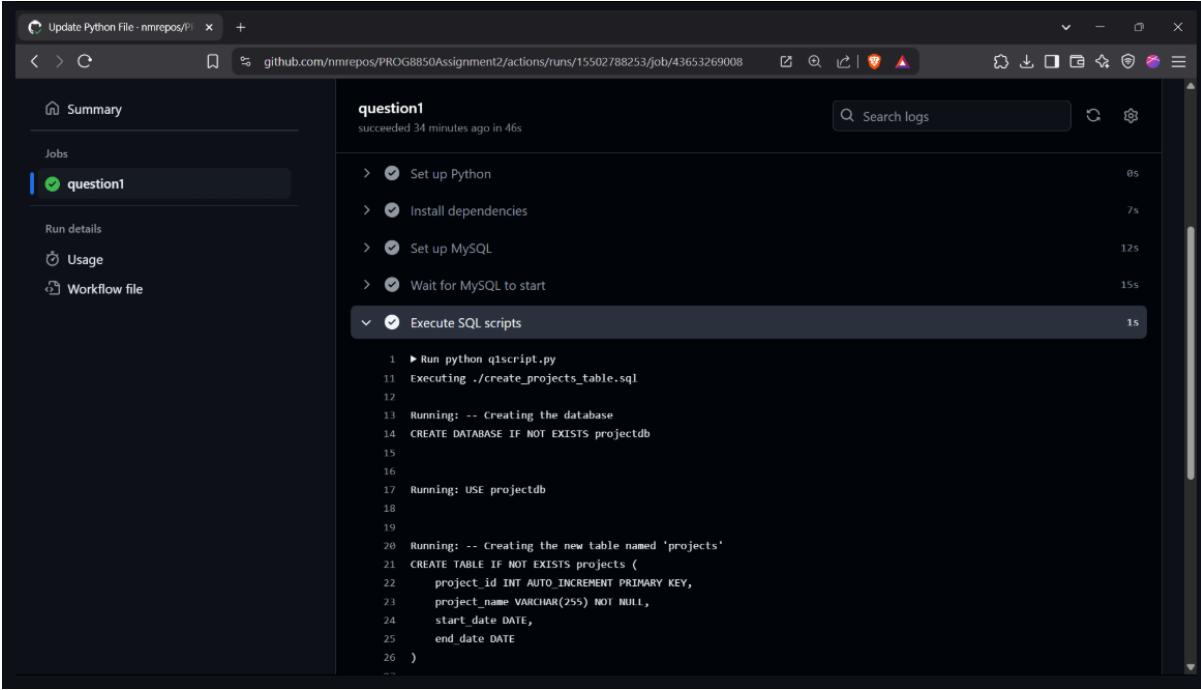
# Commit changes
connection.commit()
print("All SQL scripts executed successfully ✅ !")
```

```
print("\nVerifying `projects` table structure:\n")
cursor.execute("DESCRIBE projects;")
for row in cursor.fetchall():
    print(row)
```

```
# Close connection
cursor.close()
connection.close()
```

## Screenshots

Screenshot 1 – Running the Python Automation Script



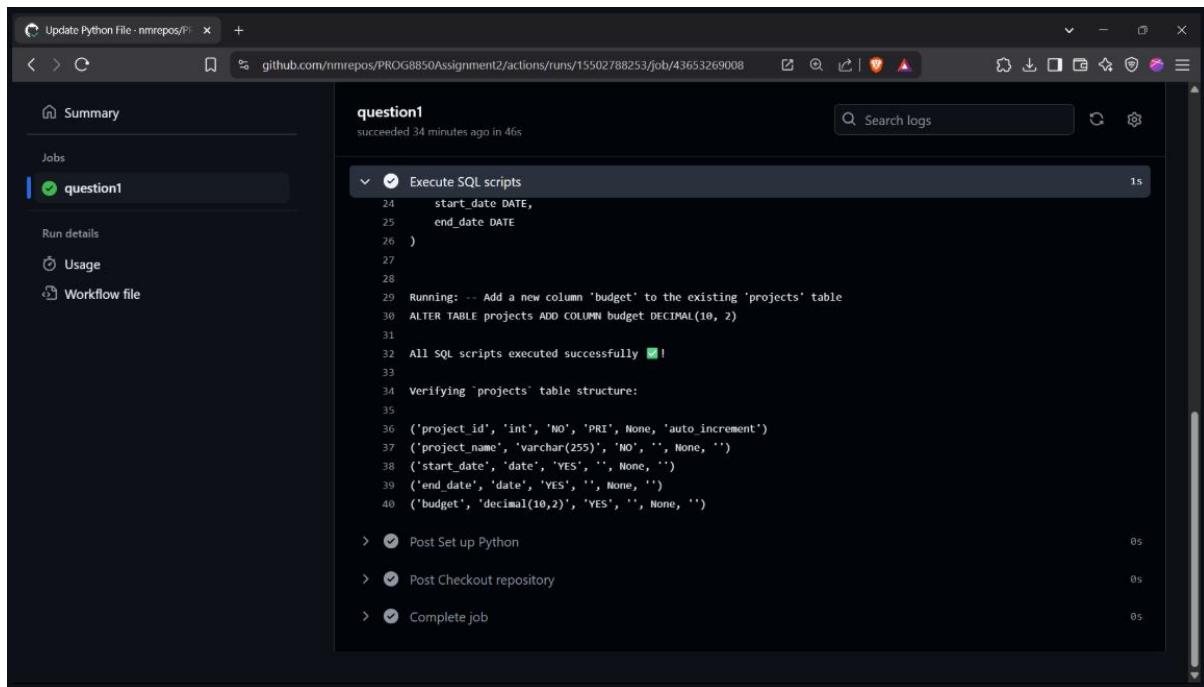
The screenshot shows a GitHub Actions log for a workflow named 'question1'. The log details the execution of several steps:

- Set up Python (0s)
- Install dependencies (7s)
- Set up MySQL (12s)
- Wait for MySQL to start (15s)
- Execute SQL scripts (15s)

The 'Execute SQL scripts' step contains the following log output:

```
1 ► Run python q1script.py
11 Executing ./create_projects_table.sql
12
13 Running: -- Creating the database
14 CREATE DATABASE IF NOT EXISTS projectdb
15
16
17 Running: USE projectdb
18
19
20 Running: -- Creating the new table named 'projects'
21 CREATE TABLE IF NOT EXISTS projects (
22     project_id INT AUTO_INCREMENT PRIMARY KEY,
23     project_name VARCHAR(255) NOT NULL,
24     start_date DATE,
25     end_date DATE
26 )
```

## Screenshot 2 – Verifying the mycompany Database



The screenshot shows a GitHub Actions job summary for a workflow named 'question1'. The job status is 'succeeded 34 minutes ago in 46s'. The main log output is as follows:

```
question1
succeeded 34 minutes ago in 46s
Search logs

Execute SQL scripts
  24    start_date DATE,
  25    end_date DATE
  26  )
  27
  28
  29 Running: -- Add a new column 'budget' to the existing 'projects' table
  30 ALTER TABLE projects ADD COLUMN budget DECIMAL(10, 2)
  31
  32 All SQL scripts executed successfully ✅!
  33
  34 Verifying 'projects' table structure:
  35
  36 ('project_id', 'int', 'NO', 'PRI', None, 'auto_increment')
  37 ('project_name', 'varchar(255)', 'NO', '', None, '')
  38 ('start_date', 'date', 'YES', '', None, '')
  39 ('end_date', 'date', 'YES', '', None, '')
  40 ('budget', 'decimal(10,2)', 'YES', '', None, '')
```

Post steps listed:

- Post Set up Python
- Post Checkout repository
- Complete job

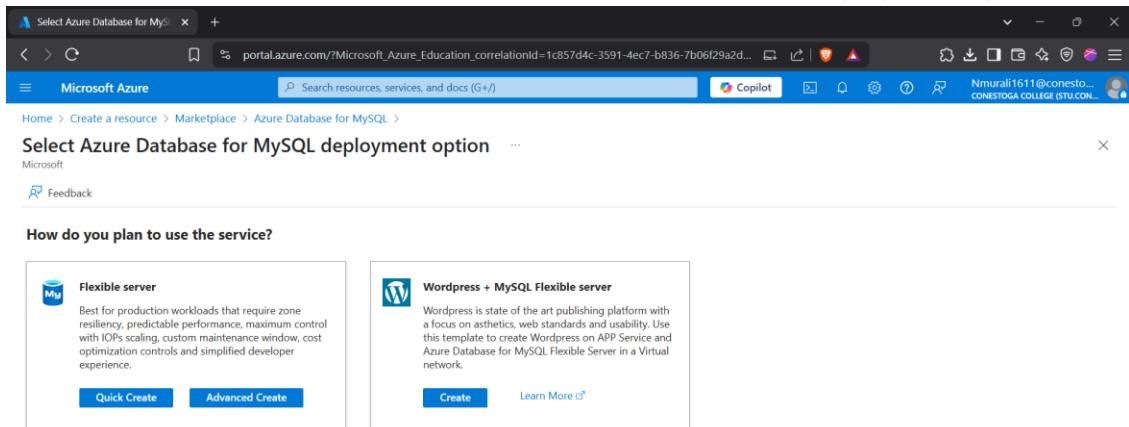
## Q2: Azure MySQL Setup and GitHub Actions Configuration

**Q2 GitHub Repo Link:** <https://github.com/nmrepos/PROG8850Assignment2/tree/question2>

### Setting Up Azure MySQL Database

#### Step 1: Create an Azure MySQL Database Instance

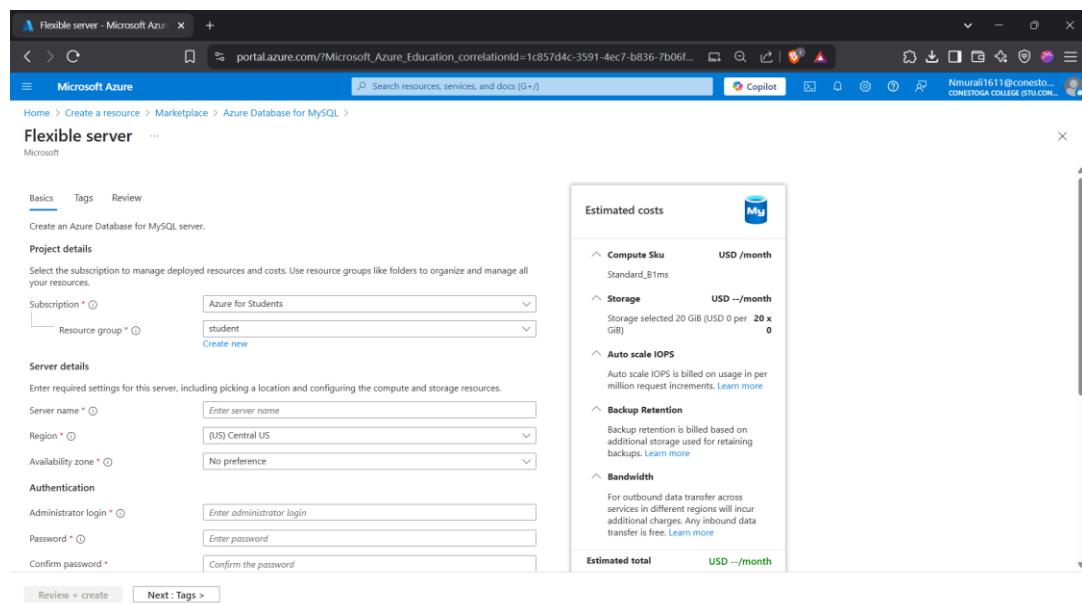
1. Log in to the Azure Portal at <https://portal.azure.com>
2. Click on “Create a resource” in the left-hand menu
3. Search for “Azure Database for MySQL” and select it
4. Click “Create” and choose “Flexible server Quick Create” as the deployment option



#### 5. Fill in the required details:

- Subscription: select your Azure subscription
- Resource group: create a new one or select an existing group
- Server name: enter a unique name (e.g., nmcompanydb-server)
- Location: choose a region (US Central)
- Admin username: create an admin username
- Password: create and note down a secure password
- Select Workload Details – Dev/Test

#### 6. Click “Review + create” and then “Create” to deploy the server



## Step 2: Configure Firewall Settings

1. Navigate to your MySQL server resource
2. Under “Settings” click “Networking”
3. Enable “Allow access to Azure services”
4. Add a firewall rule:
  - Name: AllowAll-IP (I am using github actions , if you are using the local machine you can select your IP, More Secure)
  - Start IP: 0.0.0.0
  - End IP: 255.255.255.255
5. Click “Save”

The screenshot shows the Azure portal interface for managing a MySQL flexible server named 'nmcompanydb-server'. The left sidebar navigation bar is visible, with 'Networking' selected. The main content area is titled 'nmcompanydb-server | Networking'. Under the 'Firewall rules' section, there is a table with one row: 'AllowAll\_2025-5-30\_6-56-40' (Firewall rule name), '0.0.0.0' (Start IP address), and '255.255.255.255' (End IP address). A note at the top states: 'Inbound connections from the IP addresses specified below will be allowed to port 3306 on this server.' Below the table, there is a section for 'Private endpoints' with a 'Create private endpoint' button.

## Step 3: Create the “companydb” Database

1. Under “Settings” click “Databases”
2. Click “+ Add”
3. Enter “companydb” as the name and click “OK”
4. Confirm it appears in the list

The screenshot shows the Azure portal interface for managing databases on the 'nmcompanydb-server'. The left sidebar navigation bar is visible, with 'Databases' selected. The main content area is titled 'nmcompanydb-server | Databases'. On the right, a 'Create database' dialog box is open, prompting for a 'Name' (with validation 'The value should not be empty.') and 'Character set' (set to 'utf8'). A note at the top of the dialog says: 'You can create, view and delete MySQL databases on this server. Note that you cannot connect to the databases using MySQL client tools.' Below the dialog, a table lists the existing databases: mysql, information\_schema, performance\_schema, sys, and companydb. The 'companydb' entry is highlighted in the table.

## Setting Up GitHub Actions

### Step 1: Create a GitHub Repository

### Step 2: Set Up Repository Structure

1. Open Codespace
2. Make directories:
3. `mkdir -p sql/.github/workflows`
4. Create the SQL scripts in sql folder

The screenshot shows a GitHub Codespace interface. In the center, there's a terminal window with the following content:

```
sql > add_department.sql
1 -- Create a new table named 'departments'
2 CREATE TABLE IF NOT EXISTS departments (
3     department_id INT AUTO_INCREMENT PRIMARY KEY,
4     department_name VARCHAR(255) NOT NULL,
5     location VARCHAR(255)
6 );
```

Below the terminal, a graph titled "GRAPH" displays a network of changes across multiple branches. One node is highlighted in blue: "Added SQL File, Should a... question2". Other nodes include "Update python file nmrepos", "Question 1. Screenshot...", "Update README.md nmrepos", etc.

5. Create the workflow in .github/workflows/ folder

The screenshot shows a GitHub Codespace interface with the ".github/workflows/q2.yml" file open in the editor. The file contains the following YAML configuration for a GitHub Action:

```
name: Question 2 - Database CI/CD Pipeline
on:
  push:
    branches: [ main, question2 ]
jobs:
  deploy-to-azure:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout repository
        uses: actions/checkout@v2
      - name: Set up Python
        uses: actions/setup-python@v2
        with:
          python-version: '3.9'
      - name: Install dependencies
        run:
          python -m pip install --upgrade pip
          pip install mysql-connector-python
      - name: Set up MySQL
        uses: mirromuth/mysql-action@v1.1
```

The left sidebar shows the repository structure with files like ".devcontainer", ".github/workflows/q2.yml", "add\_department.sql", "gitignore", "LICENSE", "README.md", "script.py", and "upyml".

### Step 3: Configure GitHub Secrets

1. In your repo on GitHub go to Settings → Secrets and variables → Actions
2. Add these secrets:
  - MYSQL\_ROOT\_PASSWORD (for local MySQL tests)
  - AZURE\_SQL\_HOST (e.g., nmcompanydb-server.mysql.database.azure.com)
  - AZURE\_SQL\_USER (Azure admin username)
  - AZURE\_SQL\_PASSWORD (Azure admin password)
3. Click “Add secret” for each

The screenshot shows the GitHub 'Actions secrets' configuration page for a repository named 'nmrepos/PROG8850Assignment2'. The left sidebar includes options like Branches, Tags, Rules, Actions (which is selected), Models, Webhooks, Environments, Codespaces, and Pages. Under 'Actions', there are sub-options for Advanced Security, Deploy keys, and Secrets and variables. The main content area is divided into two sections: 'Environment secrets' (which displays a message: 'This environment has no secrets.' and a 'Manage environment secrets' button) and 'Repository secrets' (which lists four secrets: AZURE\_SQL\_HOST, AZURE\_SQL\_PASSWORD, AZURE\_SQL\_USER, and MYSQL\_ROOT\_PASSWORD, all added 3 hours ago). A green 'New repository secret' button is located at the top right of the repository secrets section.

### Testing the Workflow

#### Step 1: Monitor Workflow Execution

The screenshot shows the GitHub 'Actions' page for the 'nmrepos / PROG8850Assignment2' repository. The left sidebar features sections for All workflows, Question 1, and Question 2 - Database CI/CD Pipeline (which is selected). Other sidebar options include Management, Caches, Attestations, Runners, Usage metrics, and Performance metrics. The main content area is titled 'Question 2 - Database CI/CD Pipeline' and displays '2 workflow runs'. It shows two successful runs: 'Added SQL File, Should automatically trigger the wor...' (pushed by 'nmrepos' at 7 minutes ago) and 'Update python file' (pushed by 'nmrepos' at 12 minutes ago). A 'Help us improve GitHub Actions' survey is visible at the top of the workflow runs list.

## Before Pushing SQL File

A screenshot of a GitHub Actions run page. The URL is <https://github.com/nmrepos/PROG8850Assignment2/actions/runs/15503451566>. The title is "Update python file #2". The status bar shows "Triggered via push 12 minutes ago" by "nmrepos pushed" and "Status: Success Total duration: 45s Artifacts: -". On the left, there's a sidebar with "Summary", "Jobs" (highlighted), "deploy-to-azure", "Run details", "Usage", and "Workflow file". The main area shows the workflow file "q2.yml" with a single job "deploy-to-azure" that completed successfully in 40s.

## After Pushing SQL File

A screenshot of a GitHub Actions run page. The URL is <https://github.com/nmrepos/PROG8850Assignment2/actions/runs/15503482457>. The title is "Added SQL File, Should automatically trigger the workflow. #3". The status bar shows "Triggered via push 8 minutes ago" by "nmrepos pushed" and "Status: Success Total duration: 41s Artifacts: -". The sidebar and workflow file view are identical to the previous screenshot, showing the "deploy-to-azure" job completed in 37s.

The screenshot shows a GitHub Actions workflow named "deploy-to-azure" that has completed successfully. The steps listed are:

- Set up Python
- Install dependencies
- Set up MySQL
- Wait for MySQL to start
- Deploy to Azure MySQL** (This step is expanded, showing the execution of a Python script and the execution of an SQL file to create a 'departments' table.)
- Post Set up Python
- Post Checkout repository
- Complete job

## Verifying the Azure DB for the Changes

```

nmcompanydb-server | Connect
Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
Type 'help,' or '\h' for help. Type '\c' to clear the current input statement.

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| companydb |
| information_schema |
| mysql |
| performance_schema |
| sys |
+-----+
5 rows in set (0.026 sec)

MySQL [(none)]> use companydb;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
MySQL [companydb]> describe departments;
+-----+-----+-----+-----+-----+
| Field | Type | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+
| department_id | int | NO | PRI | NULL | auto_increment |
| department_name | varchar(255) | NO | NULL | NULL | |
| location | varchar(255) | YES | | NULL | |
+-----+-----+-----+-----+-----+
3 rows in set (0.027 sec)

MySQL [companydb]> []

```

## Result

The table 'departments' with columns 'department\_id', 'department\_name', and 'location' to the 'companydb' database in AZURE.

## Final Outcome

Whenever a SQL file is pushed to the SQL folder, it should automatically trigger the workflow and apply the change to the Azure Database.

**Github Repo Link: <https://github.com/nmrepos/PROG8850Assignment2.git>**