# INFO8985 - Assignment 2: Database Automation Using CI/CD

Student Name: Premchander Jebastian

Course: INFO8985 - Database Automation

---

## 1. Creating the MySQL Database on AWS RDS

A screenshot of a computer

AI-generated content may be incorrect.

1. Log in to the AWS Management Console.
2. Navigate to RDS > Databases and click Create database.
3. Choose Standard Create, then select MySQL as the engine.
4. Use Free tier settings where applicable:  
    - DB instance identifier: companydb  
    - Master username and password
5. Select instance class (e.g., db.t4g.micro).
6. Configure storage and enable automatic backups.
7. Choose or create a VPC, subnet group, and security group.
8. Set the database to be publicly accessible if needed for external tools or GitHub Actions.
9. Review and create the database.
10. Note the endpoint and port (e.g., 3306) for use in workflows.

## 2. Setting Up GitHub Actions

GitHub Repository Settings > Secrets

A screenshot of a computer

AI-generated content may be incorrect.

GitHub Workflows – Created from the ci\_cd\_pipeline.yml file:

A screenshot of a computer

AI-generated content may be incorrect.

Steps followed:

1. Created GitHub repository: INFO8985-25S-DatabaseAutomation.
2. Created secret variables under Settings > Secrets and variables > Actions:  
    - DB\_HOST  
    - DB\_NAME  
    - DB\_USER  
    - DB\_PASS
3. Added workflow file at .github/workflows/run-schema.yml:

name: Run MySQL Schema Migration  
  
on:  
 push:  
 branches: [main]  
  
jobs:  
 apply-schema:  
 runs-on: ubuntu-latest  
  
 steps:  
 - name: Checkout code  
 uses: actions/checkout@v3  
  
 - name: Set up Python  
 uses: actions/setup-python@v4  
 with:  
 python-version: '3.x'  
  
 - name: Install MySQL connector  
 run: pip install mysql-connector-python  
  
 - name: Run schema script  
 run: |  
 python3 run\_sql.py  
 env:  
 DB\_HOST: ${{ secrets.DB\_HOST }}  
 DB\_NAME: ${{ secrets.DB\_NAME }}  
 DB\_USER: ${{ secrets.DB\_USER }}  
 DB\_PASS: ${{ secrets.DB\_PASS }}

## 3. Workflow Testing and Results

GitHub Actions Successful Run Log

A screenshot of a computer

AI-generated content may be incorrect.

RDS Tables (e.g., projects and departments)]

A screenshot of a computer

AI-generated content may be incorrect.

Steps & Results:

* On pushing to main, GitHub Actions automatically ran the workflow.
* Python script run\_sql.py executed two SQL scripts:
* - projects.sql to create the projects table and alter to add budget
* - add\_departments.sql to create the departments table
* Duplicate column error (budget) was caught and logged, but did not stop execution.
* Tables were confirmed to be created on the RDS MySQL instance.

## 4. Choice of AWS for MySQL Deployment

While various cloud providers offer managed MySQL databases, this project used Amazon RDS for MySQL due to the following reasons:

* Practicality & Speed: AWS services were readily available and familiar, which helped streamline development and reduce setup time.
* CI/CD Integration: AWS RDS offered simple network configuration and public endpoints, which eased integration with GitHub-hosted CI/CD runners.
* Learning Objectives: The primary focus of the assignment—automating database schema changes and deploying via GitHub Actions—was effectively demonstrated using AWS, aligning with the course's core outcomes.
* Cost Efficiency: AWS Free Tier options helped minimize infrastructure costs during testing and validation.

## 5. References

* AWS RDS Documentation: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/
* GitHub Actions Docs: https://docs.github.com/en/actions
* MySQL Connector Python: https://pypi.org/project/mysql-connector-python/

---