# PROG8850 Assignment 2 - Database Automation and CI/CD Deployment Report

Student Name: Twinkle Akhilesh Mishra

Student Id: 8894858

Date: June 8, 2025

Github link: <https://github.com/TwinkleM97/DBAutomation_A2>

## Overview

This report documents the steps taken to automate database schema changes and implement a CI/CD pipeline using GitHub Actions and MySQL. The implementation meets the requirements of Assignment 2, which includes:

* Automating schema updates via SQL and Python
* Creating a GitHub Actions workflow that deploys changes to a MySQL instance simulated using Docker and Scaffold in GitHub Codespaces

## Question 1: Automating Database Schema Changes

### **SQL Script (schema\_changes.sql)**

* **Creates a table named projects:**

It uses CREATE TABLE IF NOT EXISTS to ensure that table is created only once. If it already exists, it skips creation, preventing errors during re-runs.

* **Adds budget column conditionally:**

The script checks the INFORMATION\_SCHEMA.COLUMNS table to check if the budget column exists already in projects table. If it doesn’t, it will executes an ALTER TABLE statement to add the column. This ensures the schema can be safely reapplied multiple times without causing duplication or errors.

* **Inserts sample records:**

Three sample projects are inserted into the table with realistic names, dates, and budget values to test the database structure and verify that the pipeline works as intended.

**Key SQL logic for conditional column addition:**

****

### **Python Script (automate\_db.py)**

* **Connects to MySQL**:  
  Establishes a connection to the **companydb** database using the mysql-connector-python library, which is a reliable way to interact with MySQL from Python.
* **Executes both SQL scripts**:  
  Reads and runs the SQL commands from **schema\_changes.sql** and **add\_departments.sql** sequentially. It processes each statement and ensures they’re executed correctly.
* **Confirms successful execution and commits changes**:  
  After running each script, it commits the transactions to save the changes permanently. If any statement fails, it logs the error without stopping the rest of the script, ensuring fault-tolerant automation.

## Question 2: Implementing CI/CD Pipeline

### **MySQL Environment Setup**

* Simulated local MySQL instance using Docker
* Used **mysql-adminer.yml** and **up.yml** Ansible-playbook to bootstrap MySQL and Adminer
* Adminer available at [**http://localhost:8080**](http://localhost:8080)
* **GitHub Actions Workflow (ci\_cd\_pipeline.yml)**
* Triggered on push to main or manual trigger
* Executes steps:
  + Set up MySQL client & Python
  + Waits for MySQL to be ready
  + Runs ***schema\_changes.sql*** and ***add\_departments.sql***
  + Executes **automate\_db.py**
  + Verifies table structure and inserts
  + Generates deployment report

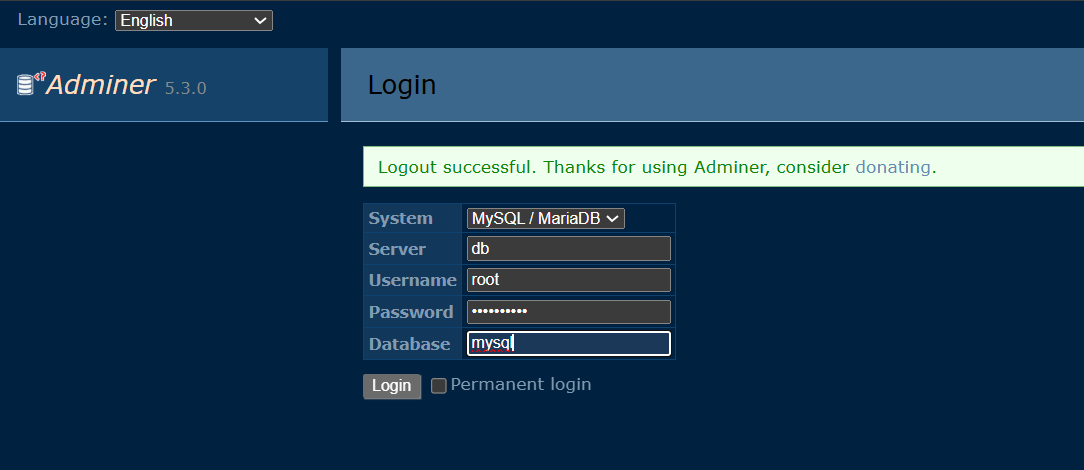
### **Final Deployment Report (from CI/CD Pipeline)**

**A screenshot of a computer

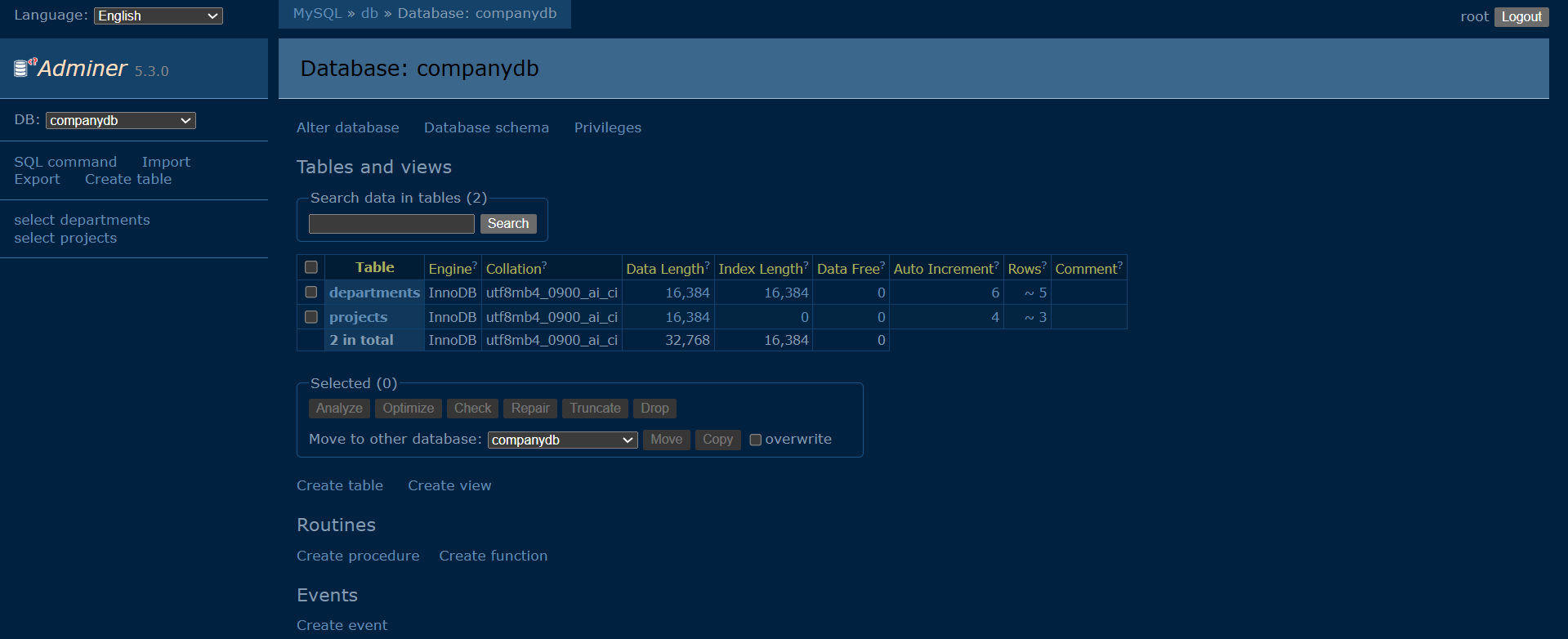
AI-generated content may be incorrect.**

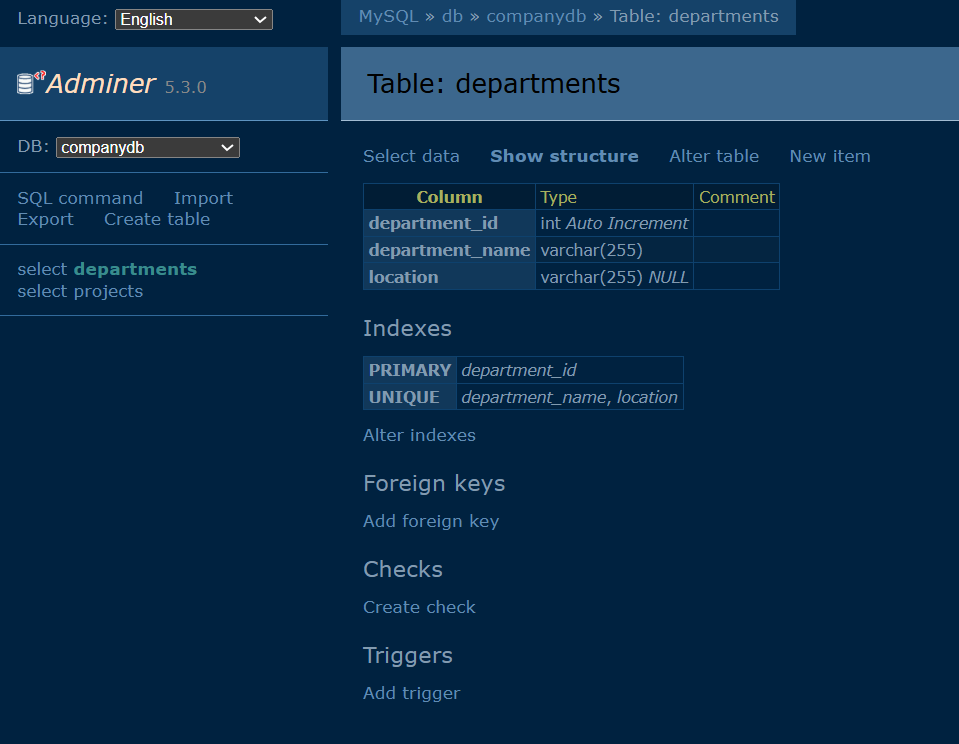
### **Screenshots**

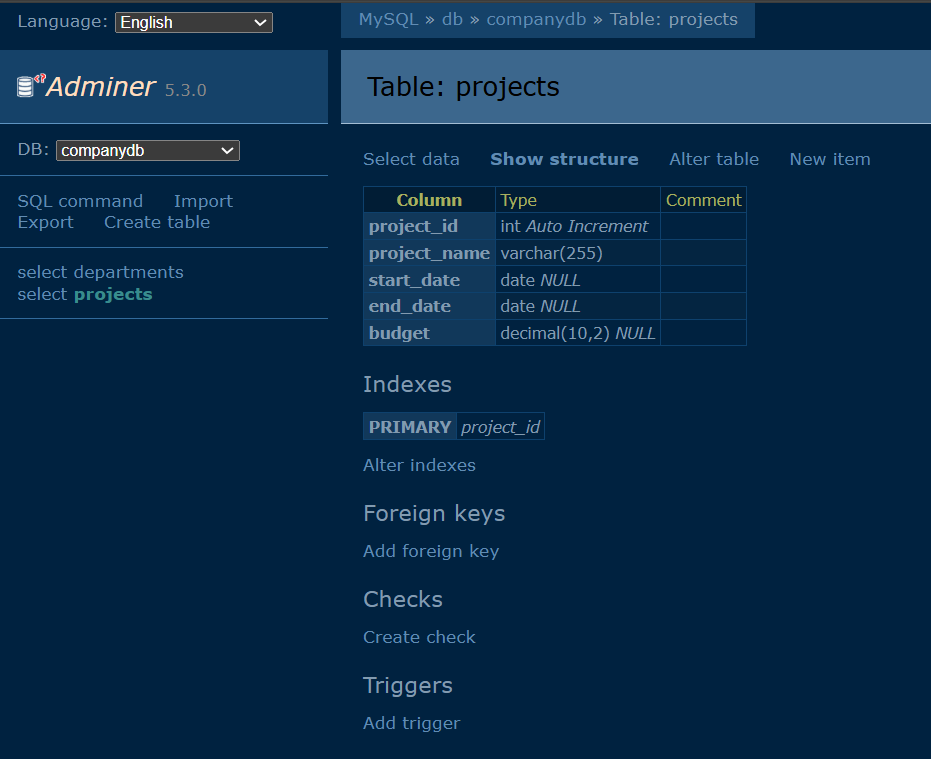
1. Adminer Login Page: Showing login to MySQL with DB companydb



1. SQL Command Tab in Adminer: Local Database Verification



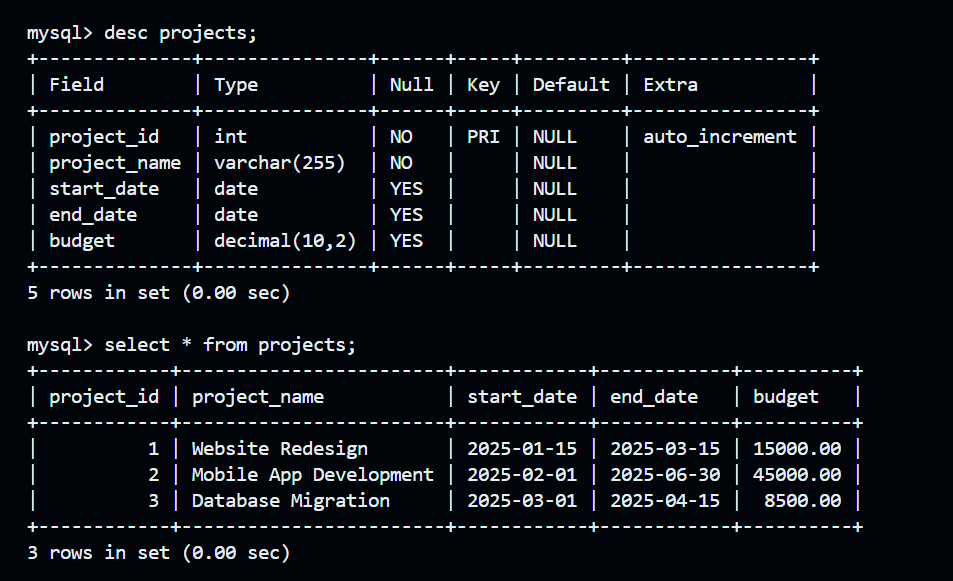




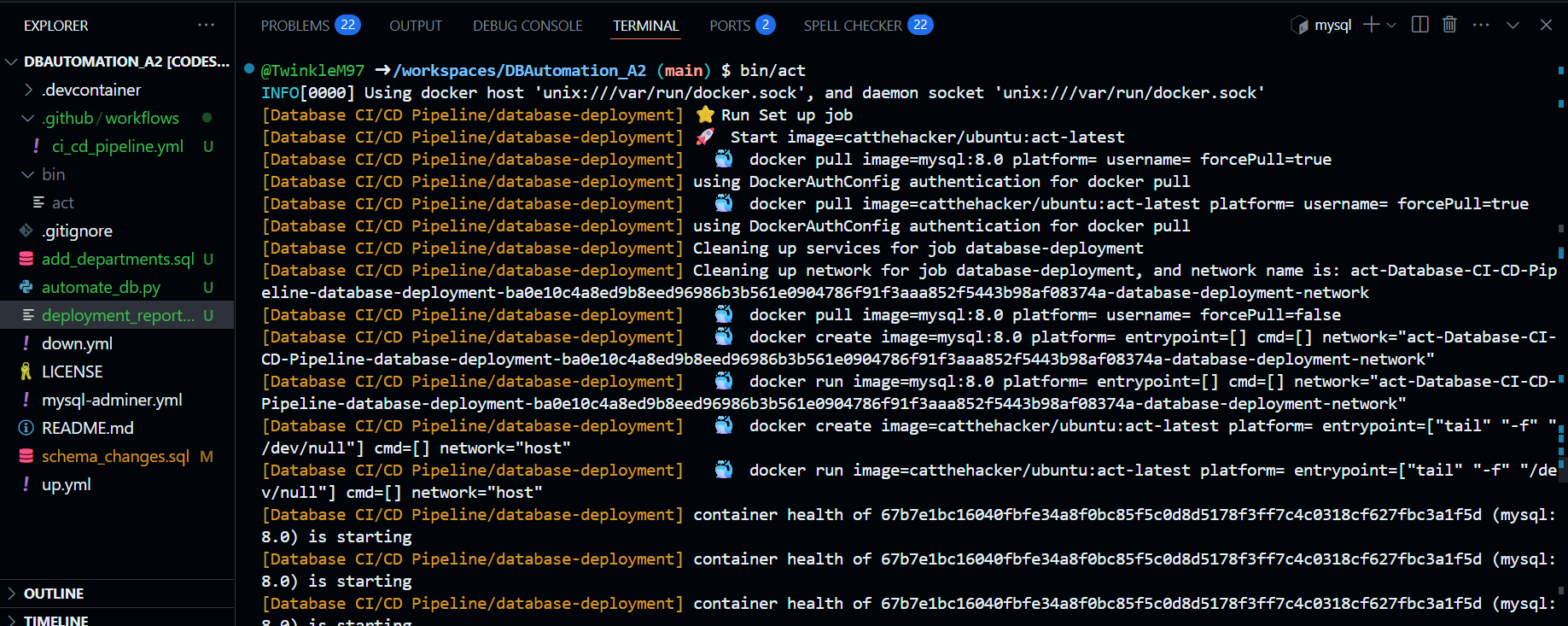
1. Terminal output: Final show tables;, desc projects;, and select \* from projects;

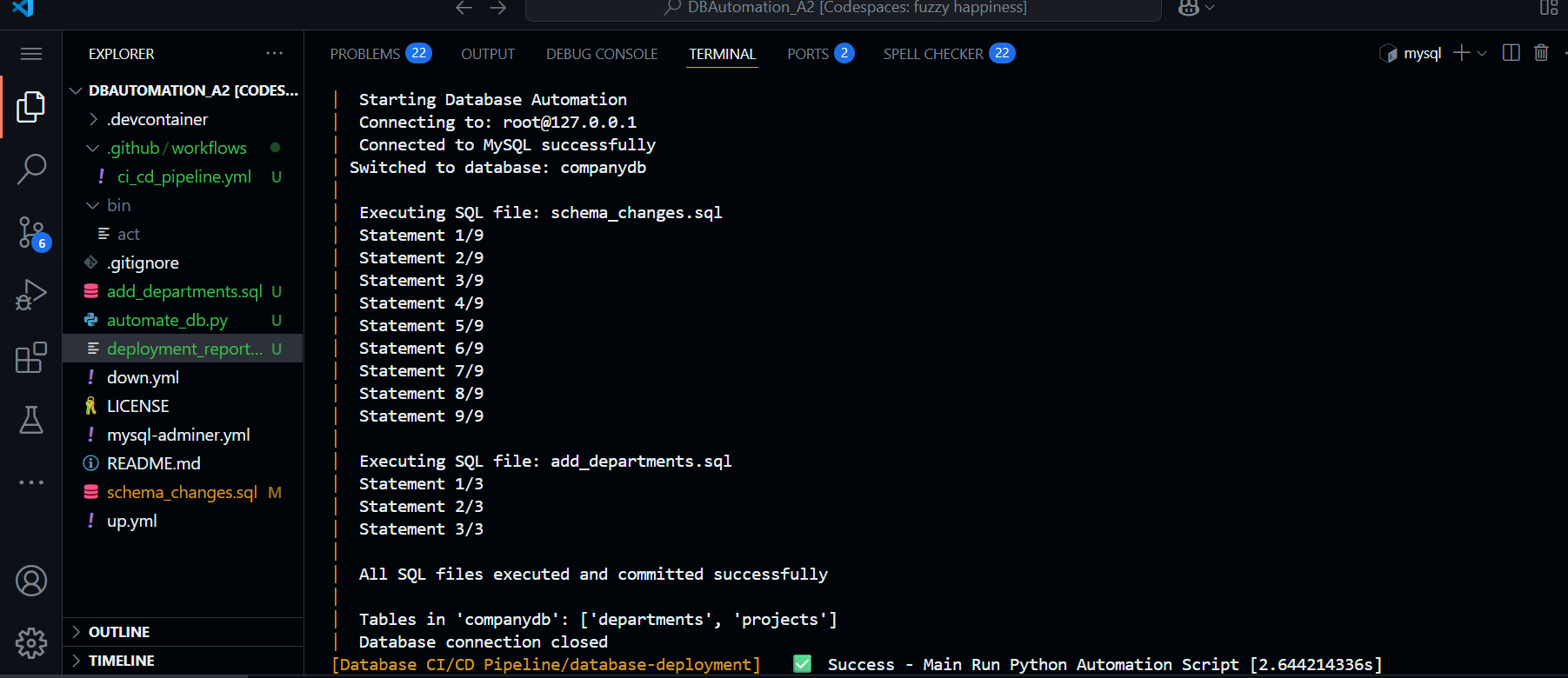
A screenshot of a computer program

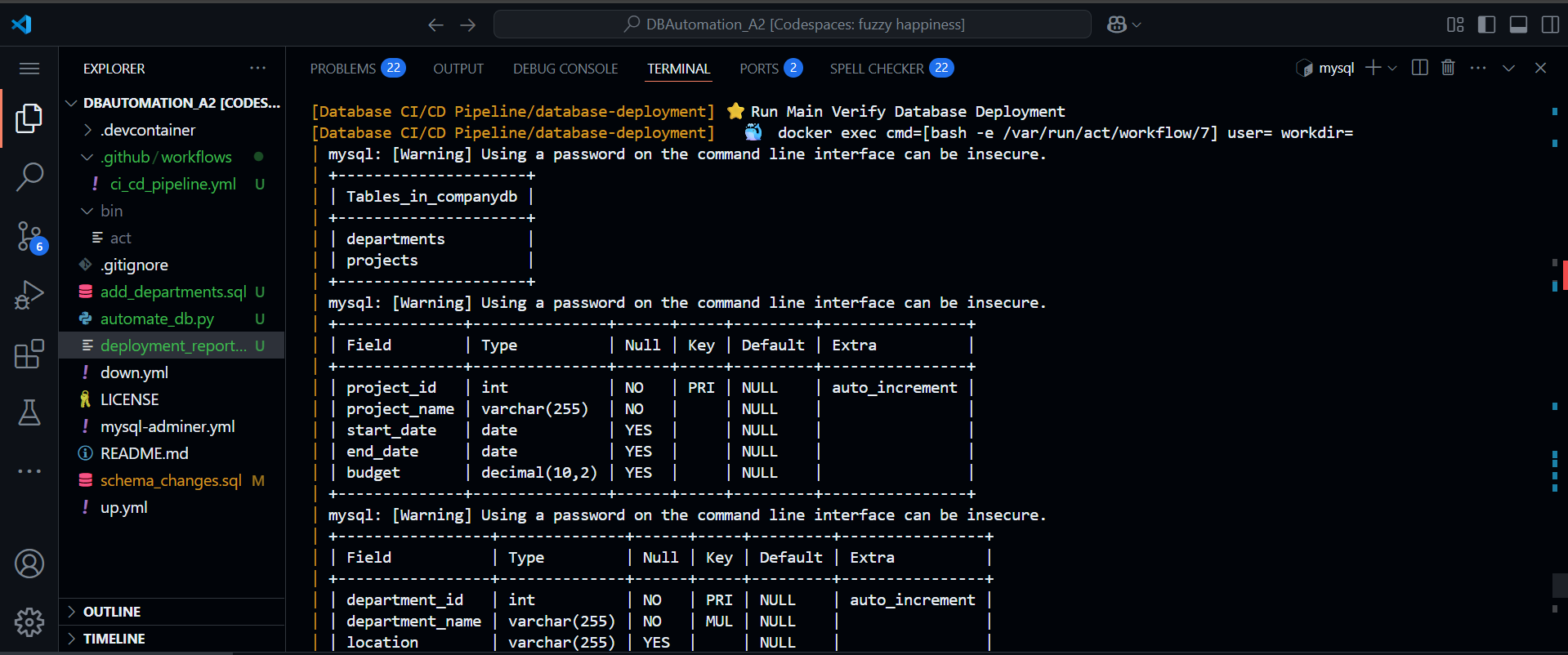
AI-generated content may be incorrect.



1. GitHub Actions output in terminal (bin/act): Success from Run Execute SQL, Run Python Automation Script, and Generate Deployment Report







A screenshot of a computer screen

AI-generated content may be incorrect.

1. Directory in Codespace: folder structure with all the files, .yml, .sql, .py, and deployment\_report.txt

A screenshot of a computer

AI-generated content may be incorrect.

1. Succesful CI-CD pipeline triggered on push:

