# Question 1: Analysis and Integration of Database Automation Tools (8 Points)

## Select any two database automation tools from the following list: Jenkins, GitHub Actions, Azure DevOps, GitLab CI/CD, Liquibase, Flyway, Ansible. For each tool: - Provide a brief overview and key features. - Create a comparison table evaluating the two tools based on- Ease of Use, Integration with CI/CD Pipelines, and Supported Databases

Tools : Flyway and Ansible

Tool 1: Flyway

Overview:

Flyway is an open source tool that assists an organization to automate version control and implementations of database schema changes, which is a light-weight database migration tool. To handle migrations, it utilises a row of SQL scripts (or Java code) executed in a specified order to make certain that the environments are always consistent.

Key Features:

* Migrations based on SQL are simple in terms of writing and handling.
* Database schema Version control.
* Accepts various relational databases (e.g. MySQL, PostgreSQL, Oracle, SQL server).
* Support of CLI, Java API, Maven, Gradle, and Docker.
* Idempotent since automatically only pending migrations are applied.
* Easy to integrate with CI/CD modules such as GitHub Actions, Jenkins or Azure DevOps.

Tool 2: Ansible

Overview:

Ansible Ansible is such an open-source IT automation tool that relates to configuration management, application deployment, and task automation. It writes infrastructure setup and management instructions using a simple YAML syntax (playbooks) to accomplish infrastructure construction and operation tasks.

Key Features:

* Agentless architecture- does not use agents on nodes, they work with SSH.
* Takes human-readable YAML playbooks.
* Various comprehensive modules are supported to automate databases and infrastructure.
* Are able to automate the process of providing database servers, installing and running configurations.
* CI/CD consortations and cloud compatibility.
* Simple to scale, but also reusable in the form of roles and inventories.

### **Comparison Table: Flyway vs Ansible**

| **Feature** | **Flyway** | **Ansible** |
| --- | --- | --- |
| **Ease of Use** | Very easy – SQL scripts with naming convention | Moderate – requires knowledge of YAML syntax |
| **CI/CD Integration** | Excellent – native support for most CI tools | Excellent – runs via shell/commands in CI tools |
| **Supported Databases** | MySQL, PostgreSQL, SQL Server, Oracle, etc. | All major databases supported via modules |
| **Primary Purpose** | Database schema versioning and migration | General infrastructure and DB provisioning |
| **Idempotency** | Built-in – automatically skips applied migrations | Must be manually handled using conditional tasks |
| **Learning Curve** | Low – ideal for developers | Medium – better suited for DevOps engineers |

## Integration Strategy: Propose a strategy to integrate the two selected tools into a CI/CD pipeline for a software project.

Objective:

To enable Flyway and Ansible within a CI/CD pipeline in order to address provisioning of databases, schema migration, and schema validation through automation, repeatability and version control.

Suggested CI/CD Flow

1. Trigger

* A developer commits his/her code or database migrations to the worse repository on GitHub (i.e. the addition of a new .sql file in the flyway/sql folder).
* This resulted in the activation of CI/CD (e.g.GitHub Actions or Jenkins pipeline).

1. Ansible provisioning

The reconstruction of the test environment will work on an Ansible playbook ( up.yaml ), run by the CI/CD tool:

* Initializes the MySQL container or service (with Docker or with MySQL in a localhost).
* Promotes the setup of the environment (e.g. configs, paths, directories, etc).

1. Migration using Flyway
2. After the MySQL server has been started, Flyway can be called either:

* By executing a shell command directly within the CI/CD, or
* Through an Ansible with a shell or command module.

1. Flyway:

* Uses fresh migration scripts of flyway/sql directory.
* Causes that only pending migrations should be run (idempotency).
* The database has a flyway\_schema\_history table where log migration history is put.

1. Schema verification

* To validate the following Python script (dbtests.py) is executed:
* There are table structure and mandatory columns (e.g. subscription\_date).

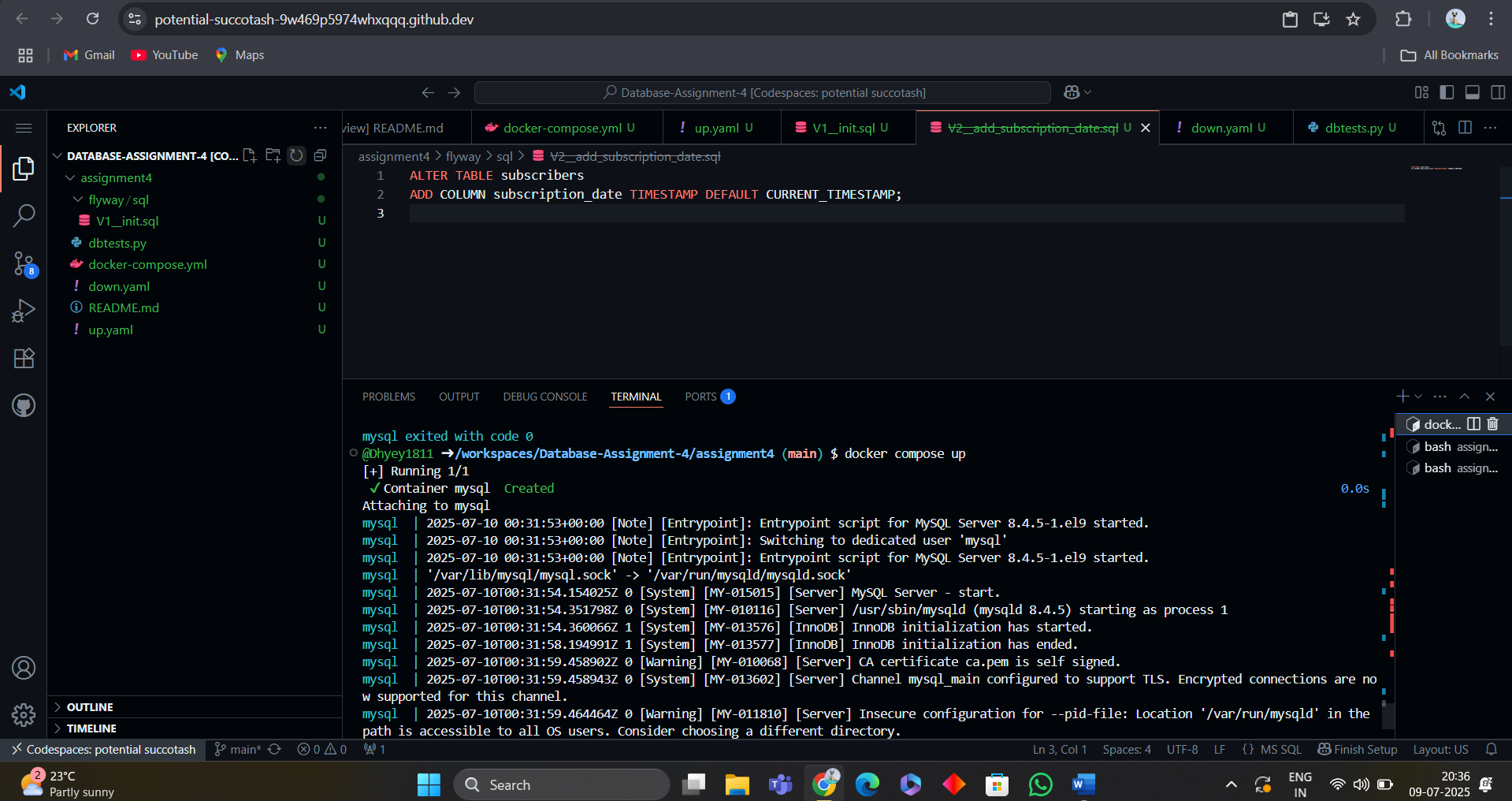
Advantages of Such Integration

* Automated: Provides automation of database change thereby reducing manual errors.
* Repeatable: Migrations are versioned, and they are installed consistently in environments.
* Scalable: It is possible to extend it to a multi-environment (dev, staging, prod) deployment.
* CI/CD Compatible: Both tools can be easily relied in the popular CI/CD solutions.
* Validation-Driven: This makes sure that the database is in the intended condition prior to proceeding with deployment.

# Question 2: Hands-on Exercise Using Ansible (12 Points) Task: Implement an up.yaml and down.yaml ansible-playbook that deploys a Mysql database, includes a schema update step, validates the update and creates a migration when down.yaml is run for the next time up.yaml is run.

Git Link : https://github.com/Dhyey1811/Database-Assignment-4.git

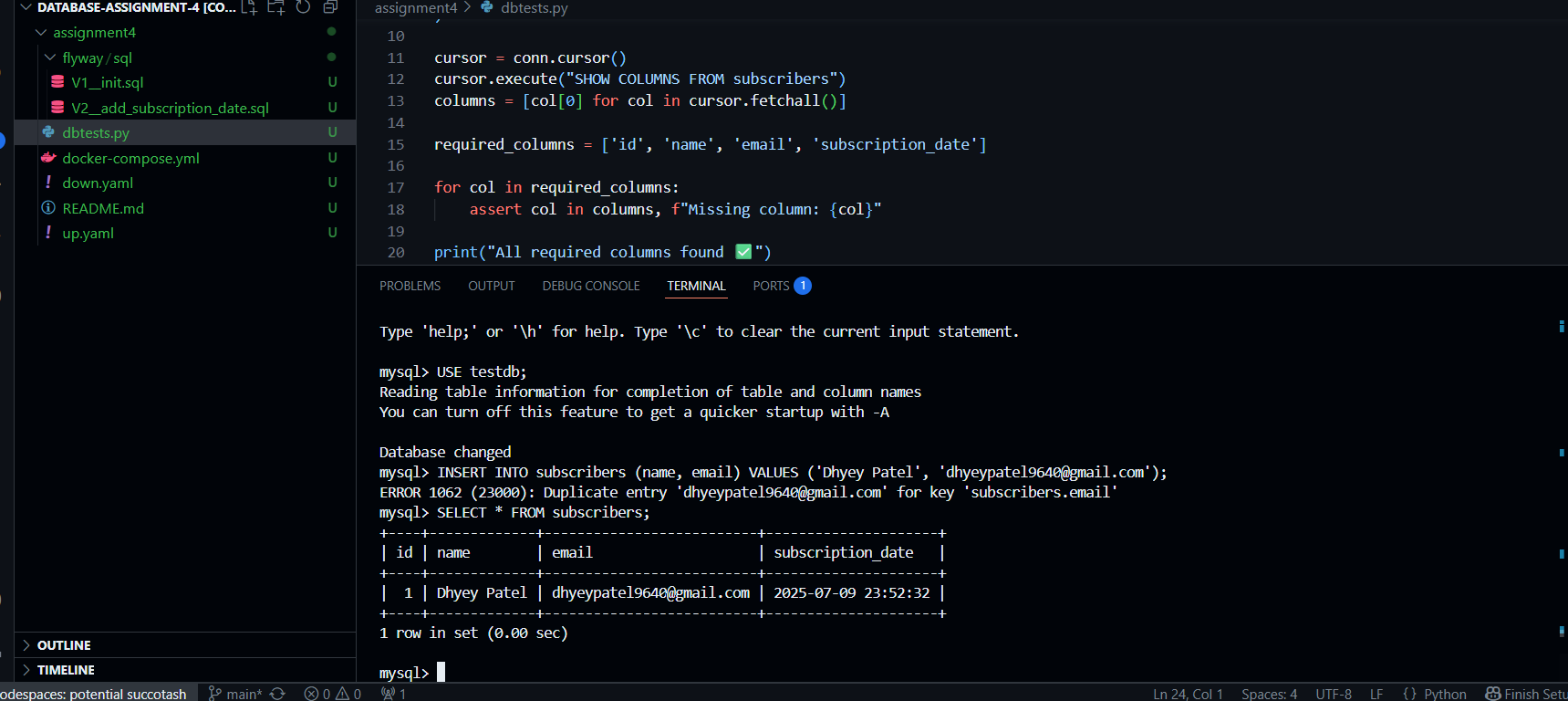
1. Run Docker



1. Database table



1. Add details



1. Migration successfully

