**AUTOMATION USING AZURE DEVOPS**

**Introduction to Azure DevOps**

Azure DevOps is a cloud-based platform that provides tools for planning, developing, testing, and delivering software efficiently. It integrates version control, build automation, CI/CD pipelines, and project tracking in a single environment. In this project, Azure DevOps was used to automate the execution of Python scripts, manage workflows, and generate reports automatically.

**Need for Automation**

Manual execution of scripts can be time-consuming and prone to errors. Automation ensures consistency, reliability, and faster execution of repetitive tasks. By using Azure DevOps, the entire analysis and ETL process can be triggered automatically whenever code changes are pushed to the repository. This improves project efficiency and reduces human intervention.

**Overview of CI/CD Concepts**

CI/CD stands for Continuous Integration and Continuous Deployment. It is a DevOps practice that automates the integration and deployment of code changes. In CI, developers regularly merge code changes into a shared repository, where automated builds and tests are run. CD extends this by automatically deploying the tested code to production. In this project, Azure DevOps pipelines were used to automate these steps.

**Week 5 Tasks Overview**

During Week 5, the focus was on automating the pipeline process using Azure DevOps. The tasks included configuring the environment, writing the YAML pipeline, and setting up automation for CI/CD processes. The following key work items were created:

• Configure Environment

• Create Pipeline YAML

• CI/CD Automation

• Azure DevOps Automation

• Save Final CSV

• Status Logic

• Load Data to DB

• ETL Status Update

**Pipeline Automation Steps**

1. Set up the project repository in Azure DevOps.

2. Add the required Python scripts, dataset files, and configuration files.

3. Create a YAML pipeline defining build and execution stages.

4. Configure triggers to automatically run the pipeline upon code changes.

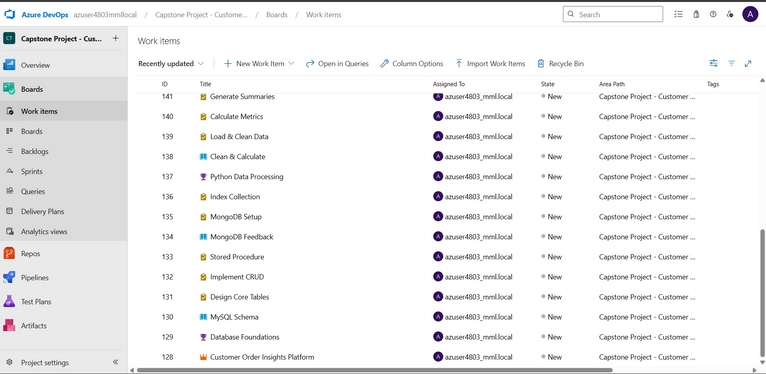
5. Use logging to record the delay summary and ETL outputs.

6. Verify pipeline run status and review generated logs.

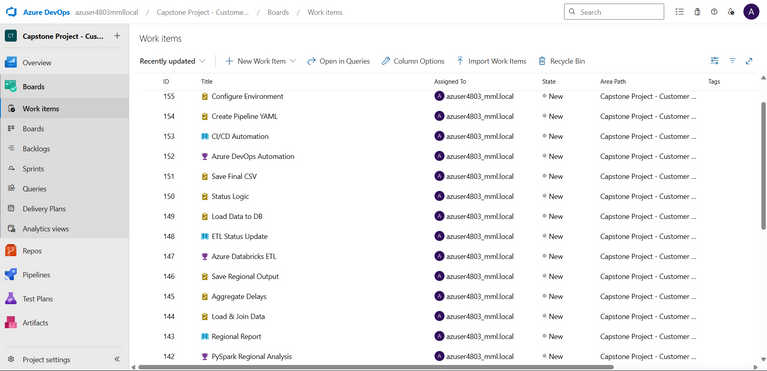
**Expected Outputs and Benefits**

The Azure DevOps pipeline successfully automated the execution of Python scripts and ETL tasks. It produced log files summarizing delay metrics and customer insights. This automation ensured consistent results, reduced manual work, and allowed continuous integration of updates from earlier weeks’ modules.

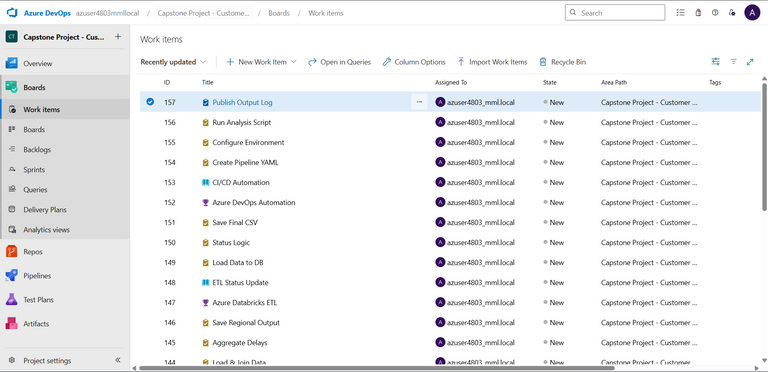
**Screenshots**



*Figure 1: Azure DevOps Board showing initial work items created for automation tasks.*



*Figure 2: Azure DevOps work items for Week 5 – Pipeline Automation phase.*



*Figure 3: Azure DevOps work items for Week 5 – Pipeline Automation phase.*

**Conclusion**

By automating the entire process using Azure DevOps, the project achieved a complete CI/CD workflow. Each phase from data collection to analysis and report generation was integrated into the pipeline, ensuring consistent results and improved delivery efficiency. This final step concluded the capstone project with a fully automated and scalable solution.