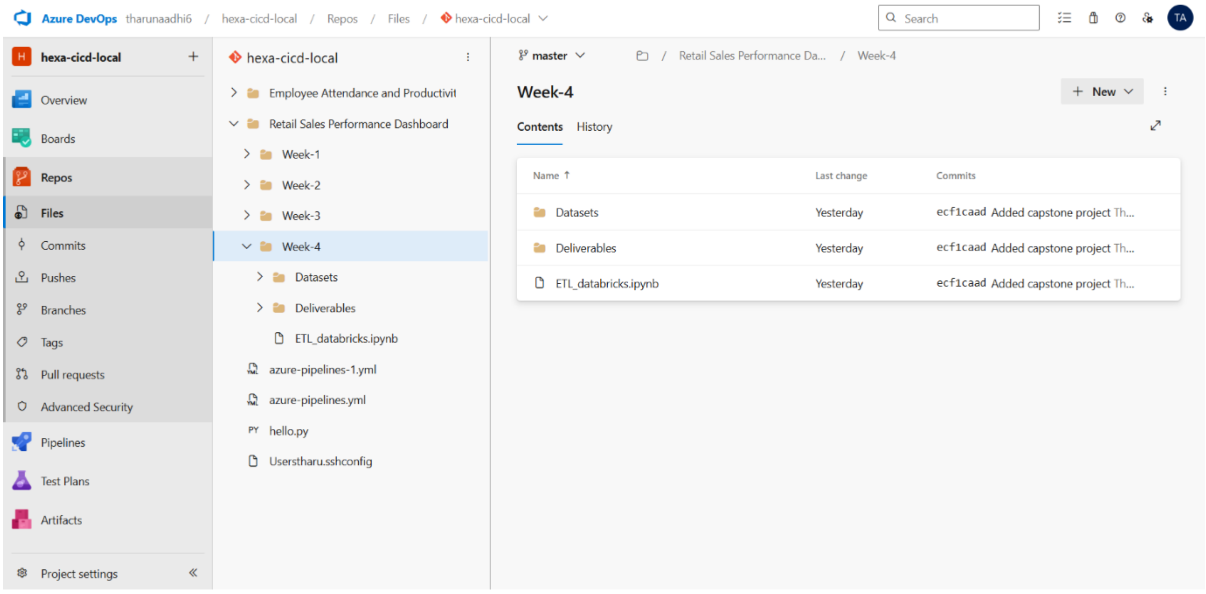
Automation via Azure DevOps

Workflow

Step-1:

Load the folder or databricks script into repo. Here I have loaded week-4 script which runs the full analysis



Azure DevOps tharunaadhi6 / hexa-cicd-local / Repos / Files / hexa-cicd-local

hexa-cicd-local

- Overview
- Boards
- Repos
- Files
- Commits
- Pushes
- Branches
- Tags
- Pull requests
- Advanced Security
- Pipelines
- Test Plans
- Artifacts
- Project settings

hexa-cicd-local

- Employee Attendance and Productivit
- Retail Sales Performance Dashboard
  - Week-1
  - Week-2
  - Week-3
  - Week-4
    - Datasets
    - Deliverables
    - ETL_databricks.ipynb
  - azure-pipelines-1.yml
  - azure-pipelines.yml
  - hello.py
  - Userstharu.sshconfig

master / Retail Sales Performance Da... / Week-4

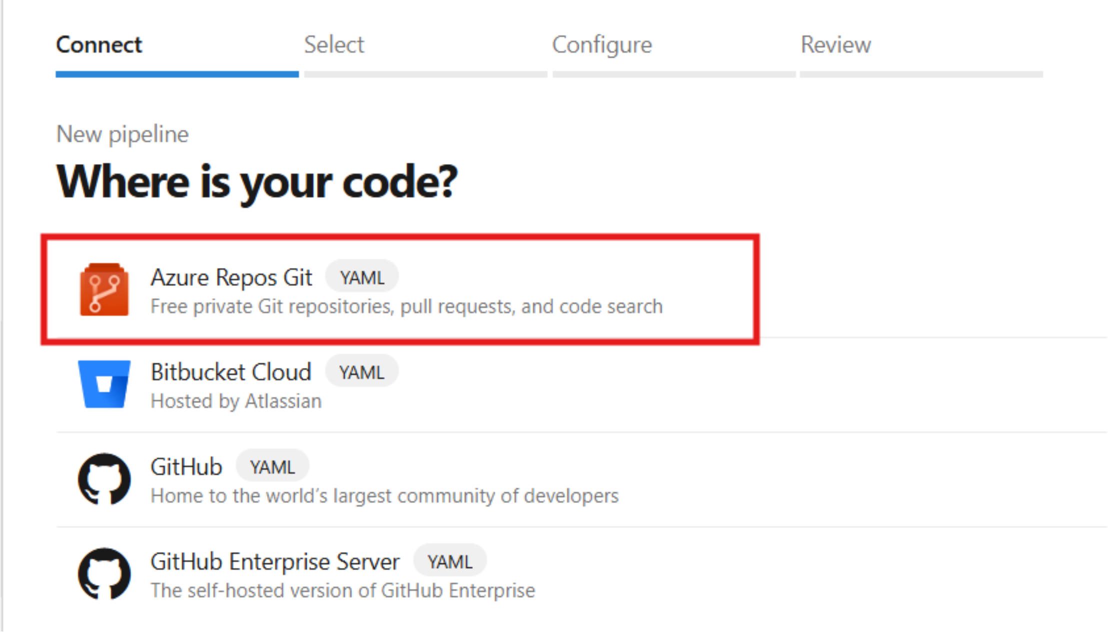
Week-4

Contents History

Name ↑	Last change	Commits
Datasets	Yesterday	ecf1caad Added capstone project Th...
Deliverables	Yesterday	ecf1caad Added capstone project Th...
ETL_databricks.ipynb	Yesterday	ecf1caad Added capstone project Th...

Step-2:

Select the Version control system(VCS) to continue. In my case it is Azure Repo Git.



Connect Select Configure Review

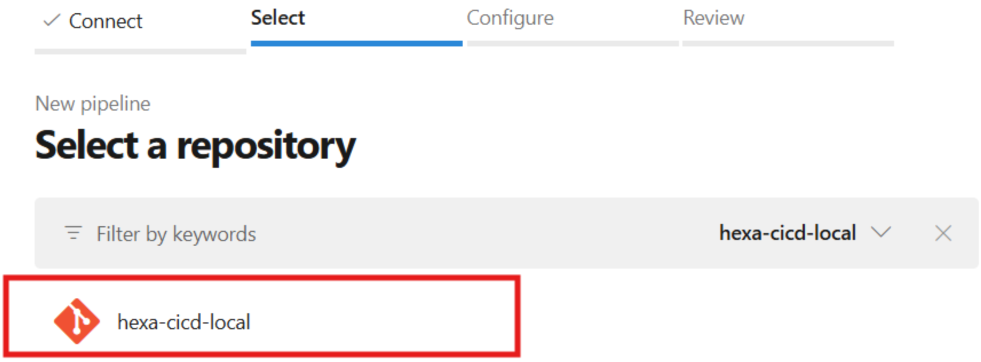
New pipeline

Where is your code?

- Azure Repos Git YAML
  Free private Git repositories, pull requests, and code search
- Bitbucket Cloud YAML
  Hosted by Atlassian
- GitHub YAML
  Home to the world's largest community of developers
- GitHub Enterprise Server YAML
  The self-hosted version of GitHub Enterprise

Step-3:

Select the respective repo at which the code, datasets are present. In my case it is present in hexa-cicd-local.



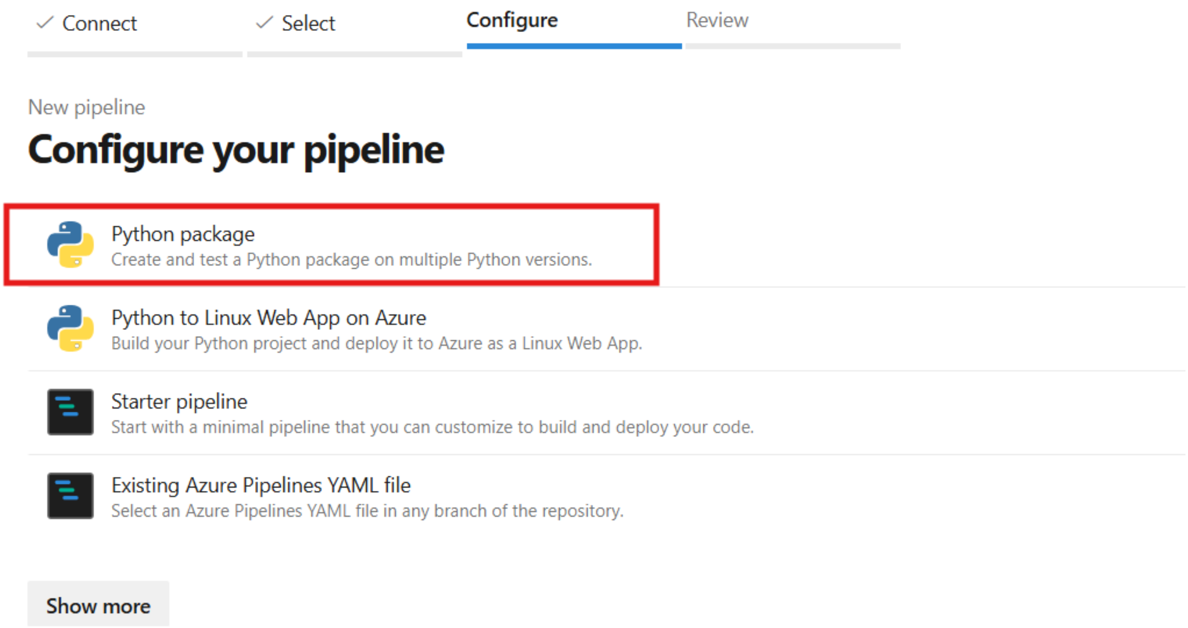
New pipeline

Select a repository

hexa-cicd-local

Step-4:

Then select python package to configure the pipeline.



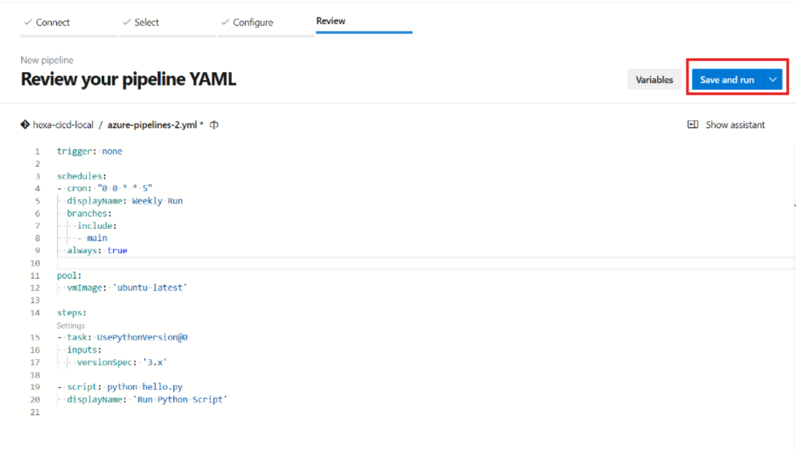
New pipeline

Configure your pipeline

- Python package — Create and test a Python package on multiple Python versions.
- Python to Linux Web App on Azure — Build your Python project and deploy it to Azure as a Linux Web App.
- Starter pipeline — Start with a minimal pipeline that you can customize to build and deploy your code.
- Existing Azure Pipelines YAML file — Select an Azure Pipelines YAML file in any branch of the repository.

Step-5:

Ensure the YAML file configured properly



New pipeline

Review your pipeline YAML

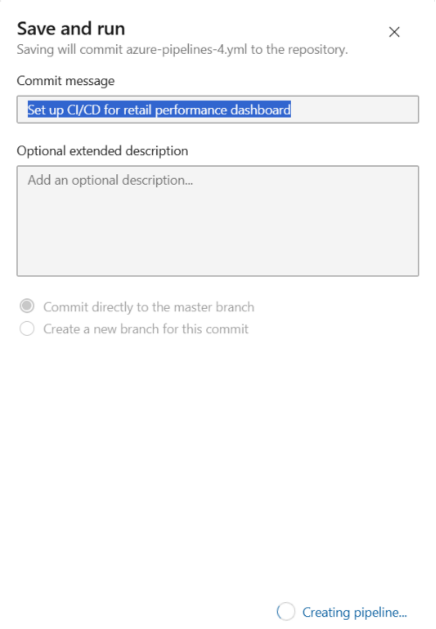
hexa-cicd-local / azure-pipelines-2.yml *

```yaml
trigger: none

schedules:
- cron: "0 0 * * 5"
  displayName: Weekly Run
  branches:
    include:
    - main
  always: true

pool:
  vmImage: 'ubuntu-latest'

steps:
- task: UsePythonVersion@0
  inputs:
    versionSpec: '3.x'

- script: python hello.py
  displayName: 'Run Python Script'

```

trigger	branch to run
pool	agent to run the tasks
vmImage	virtual machine image
steps	sequence of tasks to perform
task	a predefined tasks from azure

Step-6:

Once all configuration are set then click save and run



Save and run

Saving will commit azure-pipelines-4.yml to the repository.

Commit message: Set up CI/CD for retail performance dashboard

Optional extended description: Add an optional description...

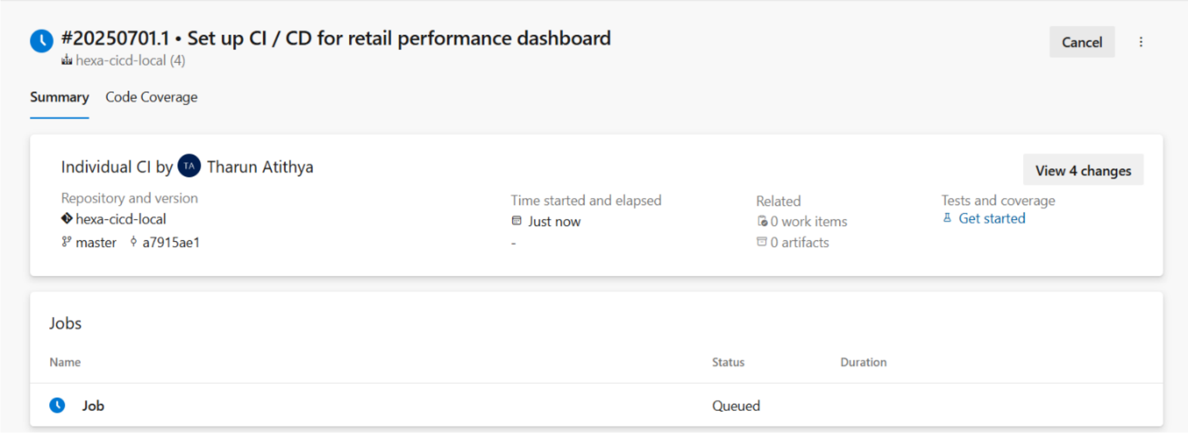
● Commit directly to the master branch

○ Create a new branch for this commit

Creating pipeline...

Step-7:

And then we can see the summary of the pipeline and it is scheduled to run the job with configured agent.



#20250701.1 • Set up CI / CD for retail performance dashboard

hexa-cicd-local (4)

Individual CI by TA Tharun Atithya

Repository and version: hexa-cicd-local, master, a7915ae1

Time started and elapsed: Just now, -

Related: 0 work items, 0 artifacts

Tests and coverage: Get started

Jobs

Name	Status	Duration
Job	Queued	

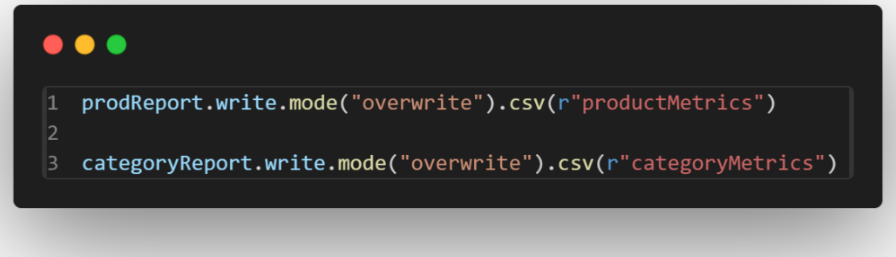
Capstone Tasks

1. Create a pipeline that runs the full analysis weekly

Entire workflow defines how to setup a devops pipeline that runs on each week Friday to generate weekly analysis report.

2. Output results to a CSV or log file

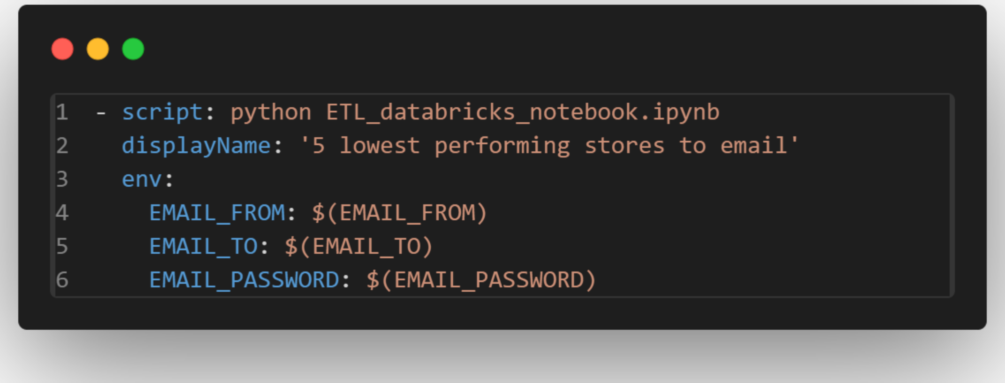
The code block shown below from the databricks script can save the both product and category report to CSV format.



```
prodReport.write.mode("overwrite").csv(r"productMetrics")

categoryReport.write.mode("overwrite").csv(r"categoryMetrics")
```

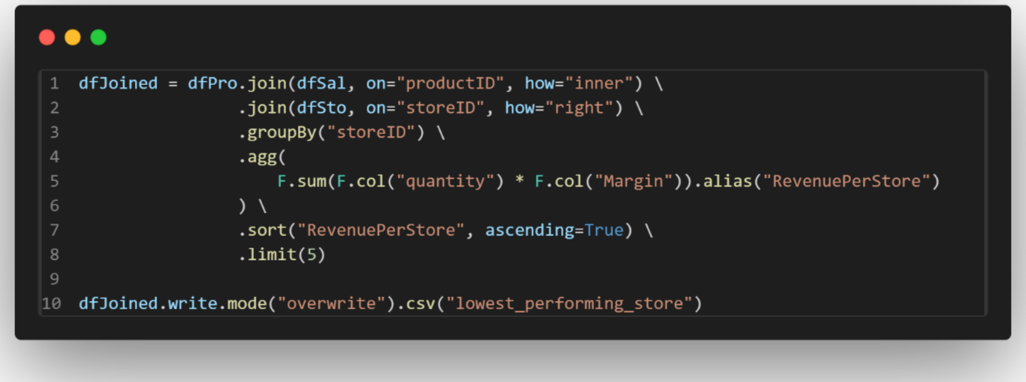
3. Add a step to email or log top 5 lowest performing stores

1. YAML configuration



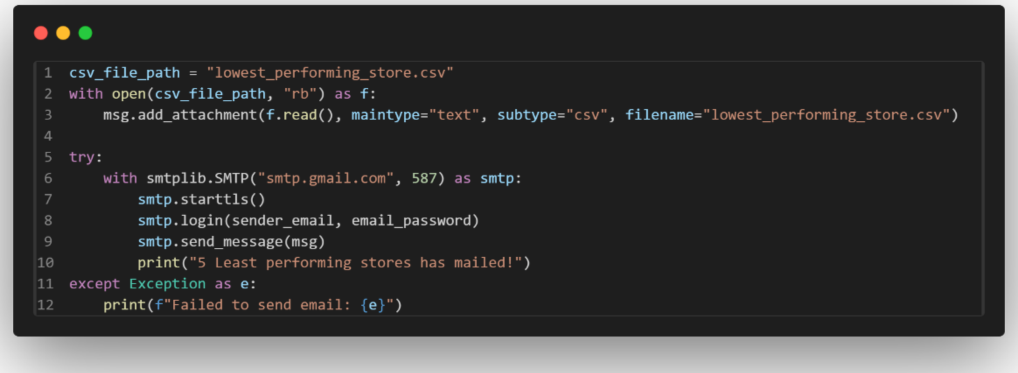
```
- script: python ETL_databricks_notebook.ipynb
  displayName: '5 lowest performing stores to email'
  env:
    EMAIL_FROM: $(EMAIL_FROM)
    EMAIL_TO: $(EMAIL_TO)
    EMAIL_PASSWORD: $(EMAIL_PASSWORD)
```

2. Python CSV report generator for 5 least performing stores



```
dfJoined = dfPro.join(dfSal, on="productID", how="inner") \
                .join(dfSto, on="storeID", how="right") \
                .groupBy("storeID") \
                .agg(
                    F.sum(F.col("quantity") * F.col("Margin")).alias("RevenuePerStore")
                ) \
                .sort("RevenuePerStore", ascending=True) \
                .limit(5)

dfJoined.write.mode("overwrite").csv("lowest_performing_store")
```

3. Python E-mail script



```
csv_file_path = "lowest_performing_store.csv"
with open(csv_file_path, "rb") as f:
    msg.add_attachment(f.read(), maintype="text", subtype="csv", filename="lowest_performing_store.csv")

try:
    with smtplib.SMTP("smtp.gmail.com", 587) as smtp:
        smtp.starttls()
        smtp.login(sender_email, email_password)
        smtp.send_message(msg)
        print("5 Least performing stores has mailed!")
except Exception as e:
    print(f"Failed to send email: {e}")
```

Deliverables

YAML pipeline file and output file showing key sales insights are present in /Deliverables folder uploaded to github.