**TASK 2**

STEPS TO CREATE A CICD PIPELINE:

1. Create a new project.
2. Clone the project folder using command git clone.
3. Move to that specified folder and open the same in code editor.
4. Now create the required files in the code editor (VS Code)
5. Raw\_sales.py

import csv

def create\_raw\_sales():

data = [

["Date", "Customer", "Amount"],

["2025-08-01", "Alice", 200],

["2025-08-02", "Bob", 150],

[None, "Charlie", None],

["2025-08-04", None, 300],

["2025-08-05", "Esha", 400],

]

with open("raw\_sales\_data.csv", "w", newline="") as f:

writer = csv.writer(f)

writer.writerows(data)

print("Raw sales data CSV created: raw\_sales\_data.csv")

if \_\_name\_\_ == "\_\_main\_\_":

create\_raw\_sales()

This code is executed with the help of Python’s built-in **csv** library. It uses the sample data given in the list data. Upon execution, it opens a new file named raw\_sales\_data.csv and writes the sales records into it.

1. Data\_processing.py

import pandas as pd

def process\_sales():

df\_raw = pd.read\_csv("raw\_sales\_data.csv")

df\_clean = df\_raw.dropna()

df\_clean.columns = [col.lower() for col in df\_clean.columns]

df\_clean["date"] = pd.to\_datetime(df\_clean["date"]).dt.strftime("%Y-%m-%d")

df\_clean.to\_csv("clean\_sales\_data.csv", index=False)

print("Cleaned sales data CSV created: clean\_sales\_data.csv")

if \_\_name\_\_ == "\_\_main\_\_":

process\_sales()

This code is also executed with the help of pandas library. Upon execution it read the raw\_sales\_data.csv, converts it into data frame and follows the following operations:

* Removes rows with missing values.
* Converts date columns to YYYY-MM-DD format.

- Normalizes the column names to lowercase.

1. Azure\_pipelines.yml:

trigger:

- main # Triggers when code is pushed to the main branch

pool:

vmImage: ubuntu-latest

steps:

# **Step 1**: Checkout the repository

- task: Checkout@1

# **Step 2**: Set up Python

- task: UsePythonVersion@0

inputs:

versionSpec: '3.10'

addToPath: true

# **Step 3**: Install dependencies

- script: |

python -m pip install --upgrade pip

pip install -r requirements.txt

displayName: 'Install dependencies'

# **Step 4**: Generate raw sales data

- script: |

cd Task 2

python raw\_sales.py

displayName: 'Generate raw sales data'

# **Step 5**: Process the sales data

- script: |

cd Task 2

python data\_processing.py

displayName: 'Process sales data'

# **Step 6**: Publish the cleaned file as an artifact

- task: PublishBuildArtifacts@1

inputs:

PathtoPublish: 'clean\_sales\_data.csv'

ArtifactName: 'CleanedData'

publishLocation: 'Container'

**trigger: - main**

* This tells Azure DevOps to **run the pipeline automatically** every time someone pushes commits to the **main** branch.
* If you commit to any other branch, this pipeline won’t start (unless you run it manually).

**pool: vmImage: ubuntu-latest**

* The pipeline will run on a **Microsoft-hosted Ubuntu Linux VM**.
* Think of it as a fresh, temporary machine that Azure gives you just for this run.

**steps:**

This is the list of actions the VM will perform, in order.

**STEP 1 - Checkout the repository**

* pull the latest code from your Azure DevOps repo into the pipeline’s build agent (VM)"requirements.txt.

**STEP 2 – Set up python**

* Installs Python 3.10 on the VM and adds it to the PATH.
* This makes the python command available for the next steps.

**STEP 3 - Install dependencies**

* Upgrades pip (Python’s package manager).
* Installs all libraries listed in requirements.txt (e.g., pandas).
* After this, the environment is ready to run your Python scripts.

**STEP 4 - Run the data fetcher**

* Runs raw\_sales.py, which creates raw\_sales\_data.csv containing sample sales data.
* You’ll see a console message like “Raw sales data CSV created: raw\_sales\_data.csv”

**STEP 5 - Process the data**

* Runs data\_processing.py, which reads raw\_sales\_data.csv and performs the operations mentioned in it.
* You’ll see “Cleaned sales data CSV created: clean\_sales\_data.csv

”.

**STEP 6 - Publish the processed file as an artifact**

* Takes the file task2/clean\_sales\_data.csv and uploads it to the pipeline run as an Artifact named CleanedData.
* After the run, in Azure DevOps → Pipelines → select the specific pipeline run → Artifacts tab, you can download clean\_sales\_data.csv from the CleanedData artifact.

**BONUS QUESTIONS:**

1. Why is data cleaning important in real-time data processing?

* **Ensures accuracy** – Real-time decisions rely on incoming data. If the raw data has duplicates, missing values, or incorrect formats, the insights or ML predictions will be wrong.
* **Prevents downstream errors** – Dirty data can break transformations, cause wrong aggregations, or trigger false alerts.
* **Improves performance** – Clean, consistent data reduces unnecessary retries or exception handling in streaming pipelines.
* **Compliance & reporting** – For industries like healthcare/finance, clean data ensures compliance with regulations.

1. What are pipeline artifacts and how are they used in DevOps workflows?

**Definition**:  
In Azure DevOps, *artifacts* are the **output files or packages generated by pipeline**.

**How they’re used**:

* Share outputs between different pipeline stages (e.g., build → test → deploy).
* Download artifacts from the **Artifacts tab** of a pipeline run.
* Store processed files for future runs, auditing, or deployments.

1. How would you modify the pipeline to store the cleaned data into Azure Blob Storage?

# Step 6: Upload cleaned file to Azure Blob Storage

- task: AzureCLI@2

inputs:

azureSubscription: 'YourServiceConnectionName' # Create a service connection in DevOps

scriptType: 'bash'

scriptLocation: 'inlineScript'

inlineScript: |

az storage blob upload \

--account-name yourstorageaccount \

--container-name yourcontainer \

--file <path-to-your-cleaned-file> \

--name <blob-name-in-container>

**AzureCLI@2 task** → allows running Azure CLI commands in the pipeline.

**Service connection** → provides authentication to your Azure account.

**Blob upload command** → pushes your cleaned file into the storage container.

After this, cleaned data will be available in **Azure Blob Storage** instead of (or in addition to) pipeline artifacts.