Week 5 – Automation via Azure DevOps (Theoretical)

Objective

Automate the **Expense Analysis** process using Azure DevOps Pipelines. The pipeline should run on a scheduled basis (weekly or monthly), fetch the latest data, execute the analysis script, and generate a summary report.

Steps to Achieve This

Step 1: Create a Project in Azure DevOps

1. Go to dev.azure.com and sign in with your account.
2. Click on **New Project**.
3. Enter a project name (e.g., *ExpenseAnalysisProject*).
4. Choose visibility as Public/Private depending on your preference.
5. Click **Create** to set up the project.

Step 2: Push Code to Azure Repos

1. Inside the new project, go to the **Repos** section.
2. Copy the Git repository URL provided by Azure DevOps.
3. Clone the repository to your local system using:
4. git clone <repo_url>
5. Add the following files to the repo:
    - Python or PySpark expense analysis script.
    - requirements.txt file with dependencies.
    - Sample cleaned data (CSV/Delta).
    - azure-pipelines.yml file (pipeline definition).
6. Commit and push the files back to Azure Repos.

Step 3: Create a New Pipeline

1. In Azure DevOps, go to the **Pipelines** section.
2. Click **New Pipeline**.

3. Select the code repository (Azure Repos Git or GitHub if connected).
4. Choose the repository where your code is stored.
5. Select **Starter Pipeline** and then replace it with your YAML pipeline file.

Step 4: Define Pipeline Stages in YAML

The pipeline should include these main stages:

1. Checkout code from the repository.
2. Set up Python environment (install Python 3.x on the build agent).
3. Install dependencies from requirements.txt.
4. Pull cleaned data (CSV/Delta files from repo or storage).
5. Run the analysis script (Python/PySpark).
6. Save results (generate a CSV summary report).
7. Publish artifact (make the CSV available for download).
8. (Optional) Upload to Azure Storage for long-term storage.

Step 5: Configure Scheduling

1. Open the azure-pipelines.yml file.
2. Add **cron expressions** to schedule pipeline runs.
    - Weekly: Every Sunday night.
    - Monthly: On the 1st day of every month.
3. Example (YAML snippet):

```
schedules:
# Weekly Run - Every Sunday at 18:00 UTC (Monday 11:30 PM IST)
- cron: "0 18 * * 0"
  displayName: "Weekly Expense Analysis"
  always: true
  branches:
   include:
```

```
    - main
  batch: true

# Monthly Run - 1st day of every month at 18:00 UTC (1st @ 11:30 PM IST)
  - cron: "0 18 1 * *"
    displayName: "Monthly Expense Analysis"
    always: true
    branches:
      include:
        - main
    batch: true
```

Step 6: Output Report

1. The analysis script generates a CSV file with:
    - Monthly spend.
    - Savings.
    - Alerts if expenses exceed 80% of income.
2. The pipeline publishes this file as an **artifact**.
3. Team members can download the report from the pipeline run summary.

Step 7: Alerts and Notifications

1. If the script detects overspending, it prints a warning message in the logs.
2. Azure DevOps captures these logs.
3. (Optional) Configure **Notifications** or use **Extensions** to send an email alert when the warning is triggered.