Course Name: Data Engineering
Term: Fall 2023
Course Description:

Data Engineering is a crucial discipline in our increasingly data-centric world. Serving as a prerequisite for a data and machine learning career, this class equips you with the skills necessary to undertake software engineering tasks in a challenging, high-pressure technology environment. The curriculum is designed to mirror real-world job scenarios, requiring a substantial time commitment from you.

Throughout the course, you will master the principles of data engineering, which include understanding diverse data types, data storage and management, and data processing to extract valuable insights. You'll gain hands-on experience with various data engineering tools and technologies, such as cloud-based data platforms, big data processing frameworks, and data visualization tools. Moreover, you will hone your skills in building and deploying data pipelines, extracting data from diverse sources, transforming data into a desired format, and loading data into a target system.

Effective teamwork is a cornerstone in data engineering, considering the diverse skill sets and expertise within a team. Consequently, this course emphasizes developing communication, collaboration, and conflict-resolution skills. Furthermore, since the field of data engineering is continuously evolving, the ability to swiftly learn and adopt new technologies and techniques is a key learning objective of the course.

To augment the learning experience, the course will engage you in weekly demos, an industry-standard practice that enhances your metacognition abilities. By understanding what you know and identifying areas of improvement, you can fast-track your path to mastery in real-world software engineering.

One of the unique aspects of the course is the emphasis on AI Pair Programming. This approach is designed to raise the difficulty level of projects while mitigating the risk of errors and side-effects. You can leverage AI Pair Programming to develop complex projects and enhance your learning experience by applying robust DevOps automation, critical thinking skills, and effective teamwork.

Upon completing the course, you'll have an impressive portfolio, including five substantial projects and 15 mini-projects, which will attest to your readiness to prospective employers. This intensive journey is supported by mentorship from faculty and TAs at a world-class institution, ensuring you're not alone as you tackle these challenges.

Guest Lecturers (Tentative):

- Mario Rodriguez GitHub: VP Product at GitHub (In Person)
- Adi Polak: Author of Scaling Machine Learning with Spark (In Person)
- Maxim David: Software Engineering Datadog (In person)
- Anna Git Data Dialogue
- Matthew Powers: Databricks ((Remote)

Learning Objectives:

- Understand the principles of data engineering, including different data types, storage, management, and data processing for valuable insights.
- Learn to use data engineering tools and technologies, including cloud-based data platforms, big data processing frameworks, and data visualization tools.
- Develop skills to build and deploy data pipelines, including extracting data from diverse sources, transforming data into the desired format, and loading data into a target system.
- Cultivate practical teamwork skills to work in a team environment, mastering effective communication, collaboration, and conflict resolution.
- Gain proficiency in learning new things quickly to stay up-to-date in the ever-evolving field of data engineering.
- Enhance your understanding and application of AI Pair Programming, DevOps automation, and critical thinking skills in software engineering projects.

Reading Material

- How Innovation Works: https://www.amazon.com/How-Innovation-Works-Flourishes-Freedom/dp/0062916599
- Teamwork: https://www.amazon.com/Teamwork-Right-Wrong-Interpersonal-Communication/dp/0803932901/

Media and labs

- Coursera
- Azure Credits
- GitHub Codespaces
- AWS Academy Learner Labs
- *Coursera-Rust Bootcamp (Live: 9/096/2023)*

Weekly Schedule:

Section One: DevOps

Section Description:

In this section, students will learn the fundamental principles of DevOps and their significance in cloud computing environments. This includes understanding the basics of cloud infrastructure, effective technical communication, and the process of cloud onboarding. Hands-on experience will be gained through the creation of a Python GitHub template and descriptive statistics scripts using Pandas and Polars.

Section Learning Objectives:

- Understand the core concepts of DevOps and their role in cloud computing.
- Learn to communicate in a technical environment effectively.
- Understand the process of cloud onboarding and the essentials of cloud infrastructure.
- Develop skills in Python and learn to create a GitHub template.
- Understand the use of Pandas and Polars for creating descriptive statistics scripts.

- Week One: Introduction to Cloud Infrastructure and Teamwork
    - Week Description: The first week serves as an introduction to the essentials of cloud infrastructure, which will form the foundation for all your future cloud-based projects. Alongside this, we'll delve into understanding the fundamental principles of teams and teamwork.
    - Media and Readings:
        - Reading: Chapter One-Teamwork Book: Toward Understanding Teams and Teamwork
        - Reading: Chap. 6 Developing on AWS with C#-DevOps
        - Coursera: Week 5: Applying DevOps Principles
    - Assignments:
        - Weekly Mini-project 1-Topic: Create a Python GitHub template you use for the rest of class (.devcontainer, Makefile, GitHub Actions, requirements.txt, README.md)
        - Prepare for In class discussion based on media and reading and complete individual discussion spreadsheet

Week 1: Create a Python GitHub Template

Requirements

- *.devcontainer configuration for a Python environment*
- *Makefile with commands for setup, testing, and linting*
- *GitHub Actions for CI/CD*
- *requirements.txt for project dependencies*
- *README.md with setup and usage instructions*

Grading Criteria

- *Correctly configured .devcontainer*
- *Working Makefile (20 points)*

- *Successful GitHub Action runs (20 points)*
- *Clear README.md (10 points)*

Deliverables

- *GitHub repository*
- *Link to successful GitHub Action run*

- **Week Two: Goal Setting and Effective Technical Communication**
    - Week Description: This week focuses on the importance of setting clear, ambitious goals as a part of effective teamwork, and how to develop and refine communication skills specifically for technical contexts.
    - Media and Readings:
        - Reading: Chapter Two-Teamwork Book: A Clear, Elevating Goal
        - Coursera: Week 2: Developing Effective Technical Communication
    - Assignments:
        - Weekly Mini-project 2: Pandas Descriptive Statistics Script
        - Prepare for In class discussion based on media and reading and complete individual discussion spreadsheet

Week 2: Pandas Descriptive Statistics Script

Requirements

- Python script using Pandas for descriptive statistics
- Read a dataset (CSV or Excel)
- Generate summary statistics (mean, median, standard deviation)
- Create at least one data visualization

Grading Criteria

- Correctly reads dataset (20 points)
- Accurate summary statistics (20 points)
- Data visualization (10 points)

Deliverables

- Python script
- Generated summary report (PDF or markdown)

- **Week Three: Results-Driven Structure and Cloud Onboarding**
    - Week Description: In the third week, we will focus on the importance of structuring a team in a way that is driven by results, as well as an introduction to cloud onboarding processes, including its challenges and best practices.
    - Media and Readings:
        - Reading: Chapter Three-Teamwork Book: Results-Driven Structure
    - Coursera: Week 3: Exploring Cloud Onboarding
    - Assignments:
        - Individual Project #1 Due
        - Weekly Mini-project 3: Polars Descriptive Statistics Script

        - Prepare for In class discussion based on media and reading and complete individual discussion spreadsheet

Week 3: Polars Descriptive Statistics Script

Requirements

- Python script using Polars for descriptive statistics
- Read a dataset (CSV or Excel)
- Generate summary statistics (mean, median, standard deviation)
- Create at least one data visualization

Grading Criteria

- Correctly reads dataset (20 points)
- Accurate summary statistics (20 points)
- Data visualization (10 points)

Deliverables

- Python script
- Generated summary report (PDF or markdown)

- **Week Four: Competence and DevOps Principles**
    - Week Description: The fourth week focuses on the importance of having competent team members and how their skills can significantly contribute to achieving team goals. Also, we'll start exploring DevOps principles, an integral part of modern cloud computing practices.
    - Reading: Chapter Four-Teamwork Book: Competent Team Members
    - Weekly Mini-project 4: Create a GitHub Actions Matrix Build that tests more than one than one version of Python.
    - Coursera: Week 5: Applying DevOps Principles

Week 4: GitHub Actions Matrix Build for Multiple Python Versions

Requirements

- Set up a GitHub Actions workflow
- Test across at least 3 different Python versions

Grading Criteria

- Correctly configured GitHub Actions Matrix (40 points)

Deliverables

- GitHub repository
- Link to successful GitHub Actions Matrix run

Section Two: Building Tools with SQL, Rust, and Python
Co-Instructor: Alfredo Deza

Section Description:

In the second section of the course, we move towards hands-on application of the principles learned so far, delving into Python scripting, SQL, and Rust. You will gain experience building tools with these technologies, enhancing your knowledge and skill set.

Section Learning Objectives:

- Understand the application of Python scripting in conjunction with SQL.
- Learn to create and manage MySQL databases.
- Understand the principles of Python packaging and the creation of command-line tools.
- Learn the transition process from Python to Rust, particularly for MLOps and Data Engineering.
- **Week Five: Python Scripting, SQL, and Fostering Unified Commitment**
    - Week Description: This week, you'll learn about Python scripting and SQL, integral tools for building robust data applications. We'll also discuss the importance of fostering a unified commitment within a team.
    - Coursera: Week 2: Python Scripting and SQL
    - Reading: Chapter Five-Teamwork Book: Unified Commitment
    - Reading: One size database doesn't fit anyone
    - Reading: Understanding Availability
    - Reading: Learning MySQL, 2nd Edition-Chapter 6-Transactions and Locking
    - Mini-Project: Create a Python script that interacts with a SQL database.

Week 5: Python Script interacting with SQL Database
Requirements
- Connect to a SQL database
- Perform CRUD operations
- Write at least two different SQL queries

Grading Criteria
- Database connection (20 points)
- CRUD operations (20 points)

Deliverables
- Python script
- Screenshot or log of successful database operations

- **Week Six: Advanced SQL and Promoting a Collaborative Climate**
    - Week Description: This week is about mastering advanced SQL techniques and understanding how a collaborative climate can significantly impact teamwork

    - Coursera: Week 4: Working with MySQL
    - Reading: Chapter Six-Teamwork Book: Collaborative Climate
    - Reading: SQL Pocket Guide, 4th Edition-Chapter1-SQL Crash Course
    - Mini-Project: Design a complex SQL query for a MySQL database and explain the results.

Week 6: Complex SQL Query for a MySQL Database *(Can be any external database including DynamoDB, Databricks, or even Neo4)*

Requirements

- Design a complex SQL query involving joins, aggregation, and sorting
- Provide an explanation for what the query is doing and the expected results

Grading Criteria

- Query functionality (20 points)
- Explanation and documentation (20 points)

Deliverables

- SQL query
- Written or video explanation of the query
- If you choose a NoSQL database (swap instructions in a way that make sense for your databases)

- **Week Seven: Python Packaging, Command Line Tools, and Upholding Standards of Excellence**
    - Week Description: This week introduces Python packaging and the use of command-line tools. We will also discuss the importance of setting and upholding standards of excellence within a team.
    - Coursera: Week 4: Python Packaging and Command Line Tools
    - Reading: Chapter Seven-Teamwork Book: Standards of Excellence
    - Mini-Project: Package a Python script into a command-line tool and write a user guide.

Week 7: Package a Python Script into a Command-Line Tool (Students can do this in Rust as well)

Requirements

- Package a Python script with setuptools or a similar tool
- Include a user guide on how to install and use the tool
- Include communication with an external or internal database (NoSQL, SQL, etc) [If you use Rust you can skip the DB part]

Grading Criteria

- Functionality of the tool (20 points)
- User guide clarity (20 points)

Deliverables

- Python package
- User guide (PDF or markdown)

- You can delay turning this in until the following week so that Alfredo can guide through it

- **Week Eight: Transitioning from Python to Rust for MLOps and the Role of External Support and Recognition**
    - Week Description: This week delves into the practical aspects of transitioning from Python to Rust in MLOps. We will also examine the role of external support and recognition in a team's success.
    - Coursera: Week 5: Rust for MLOps: The Practical Transition from Python to Rust
    - Reading: Chapter Eight-Teamwork Book: External Support and Recognition
    - Mini-Project: Rewrite a Python script for data processing in Rust, highlighting the improvements in speed and resource usage.
    - Individual Project #2: Rust CLI Binary with SQLite

Week 8: Rewrite a Python Script in Rust

Requirements

- Take an existing Python script for data processing
- Rewrite it in Rust
- Highlight improvements in speed and resource usage

Grading Criteria

- Functionality in Rust (20 points)
- Demonstrated improvements (20 points)

Deliverables

- Rust and Python scripts
- Performance comparison report (PDF or markdown)

Section Three: Building Data Pipelines

Section Description:

This section delves into the creation and management of data pipelines using various cloud-hosted tools and platforms. The emphasis will be on PySpark, the Databricks Platform, and MLflow, each a critical tool in modern data engineering. Alongside these technical skills, you'll explore principled leadership in teamwork and delve into management dynamics within teams.

Section Learning Objectives:

- Understand the concept and application of data pipelines in data management.
- Learn to use Cloud-Hosted Notebooks for data processing.
- Get acquainted with PySpark, its benefits, and how to use it for data processing.
- Explore the Databricks platform and learn to leverage it for data management.

- Understand the basics of MLflow and its application in Machine Learning projects.

Week9

- Week Nine: Cloud-Hosted Notebooks and Principled Leadership
    - Week Description: This week introduces you to the concept of cloud-hosted notebooks and their application in data management. Concurrently, we will discuss the role of principled leadership in successful teamwork.
    - Coursera: Week 2: Cloud-Hosted Notebooks
    - Reading: Chapter Nine-Teamwork Book: Principled Leadership
    - Reading: Chapter Ten-Teamwork Book: Inside Management Teams
    - Mini-project 9: Set up a cloud-hosted notebook and demonstrate data manipulation with a sample dataset.

Week 9: Cloud-Hosted Notebook Data Manipulation

Requirements

- Set up a cloud-hosted Jupyter Notebook (e.g., Google Colab)
- Perform data manipulation tasks on a sample dataset

Grading Criteria

- Setup and configuration (20 points)
- Data manipulation tasks (20 points)

Deliverables

- Link to the cloud-hosted notebook
- Document or video demonstrating the tasks performed

Week10

- Week Ten: Introduction to PySpark and Innovation in Energy and Public Health
    - Week Description: This week serves as an introduction to PySpark, a powerful tool for large-scale data processing. Also, we will explore the innovations in the energy and public health sectors.
    - Coursera: Week 1: Overview and Introduction to PySpark
    - Reading: Intro, Chap. 1,2: How Innovation Works (Energy, Public Health)
    - Mini-project 10: Use PySpark to perform data processing on a large dataset.

Week 10: PySpark Data Processing

Requirements

- Use PySpark to perform data processing on a large dataset
- Include at least one Spark SQL query and one data transformation

Grading Criteria

- Data processing functionality (20 points)
- Use of Spark SQL and transformations (20 points)

Deliverables

- PySpark script
- Output data or summary report (PDF or markdown)

- Week Eleven: Using the Databricks Platform
    - Coursera: Azure Databricks and MLFlow
    - Week Description: This week, you'll work with the Databricks platform, designed for massive-scale data engineering and collaborative data science.
    - Mini-project 11: Create a data pipeline using the Databricks platform.
    - Reading: Chap. 3,4,5: How Innovation Works (Transport, Food, Low-technology innovation)

Week 11: Data Pipeline with Databricks

Requirements

- Create a data pipeline using Databricks
- Include at least one data source and one data sink

Grading Criteria

- Pipeline functionality (20 points)
- Data source and sink configuration (20 points)

Deliverables

- Databricks notebook or script
- Document or video demonstrating the pipeline

- Week Twelve
    - Coursera: Week 1: Introduction to MLflow
    - Reading: Chap: 6,7: How Innovation Works: (Prehistoric innovation, Innovation's essentials)
    - Mini-project 12: Use MLflow to manage a simple machine learning project.
    - Individual Project #3: Databricks ETL (Extract Transform Load) Pipeline

Week 12: Use MLflow to Manage an ML Project

Requirements

- Create a simple machine-learning model
- Use MLflow to manage the project, including tracking metrics

Grading Criteria

- Model functionality (20 points)
- MLflow tracking (20 points)

Deliverables

- MLflow project directory
- Document or video demonstrating tracking with MLflow

Section Four: Building Containerized Serverless Data Pipelines
Co-Instructor: Derek Wales

- Week Thirteen: Virtualization, Containers, and Understanding Innovation Failures
    - Week Description: This week, we dive into virtualization and containers, crucial components in building serverless data pipelines. We'll also explore various reasons why some innovations fail.
    - Coursera: Week 2: Virtualization and Containers
    - Reading: Chap: 9,10: How Innovation Works: (Fakes, frauds, fads, and failures)
    - Mini-project 13: Build and deploy a simple containerized application using Docker.

Week 13: Dockerized Application
Requirements
- Create a simple Python application
- Dockerize the application

Grading Criteria
- Application functionality (20 points)
- Docker configuration (20 points)

Deliverables
- Dockerfile
- Link to Docker image on Docker Hub or similar service

- Week Fourteen: Python Microservices, Resistance to Innovation
    - Week Description: You will learn about Python microservices, an architectural style that structures an application as a collection of loosely coupled services. We'll also understand the resistance to innovation and its impacts.
    - Coursera: Week 3: Python Microservices
    - Reading: Chap: 11,12: How Innovation Works: (Resistance to innovation, An innovation famine)
    - Present Final Project in Class
    - Mini-project 14: Develop a simple microservice using Python and deploy it to a cloud platform
    - Present Final Project in Class

Week 14: Python Microservice on Cloud Platform
Requirements
- Develop a simple Python-based microservice

- Deploy it on AWS Lambda, Google Cloud Functions, or similar

Grading Criteria

- Microservice functionality (20 points)
- Deployment (20 points)

Deliverables

- Code repository
- Link to the deployed microservice

Course Grade

- Course Grade:
- Individual Projects:
    - Project 1: Continuous Integration using GitHub Actions of Python Data Science Project (6.25%)
    - Project 2: Rust CLI Binary with SQLite (6.25%)
    - Project 3: Databricks ETL (Extract Transform Load) Pipeline (6.25%)
    - Project 4: Individual Project (6.25%)
- Team Project (25%)
- Class Discussion Grade (25%)
- Mini-projects (25%)

Letter Grades

- A+: 97-100% A: 93-96% A-: 90-92%
- B+: 87-89% B: 83-86% B-: 80-82%
- C+: 77-79% C: 73-76% C-: 70-72%
- D+: 67-69% D: 63-66% D-: 60-62%
- F: Below 60%

Notes

- The final grade will be calculated using the weighted average of the scores for each project.
- The team project will be graded on the following criteria:
- Teamwork: The team worked effectively together and communicated well.
- Communication: The team communicated effectively with the instructor and other students.
- Technical skills: The team used the appropriate technical skills to complete the project.
- Creativity: The team demonstrated creativity in their approach to the project.
- The class discussion grade will be based on the following criteria:
- Participation: The student participated actively in class discussions.

- Thoughtfulness: The student's contributions to class discussions were thoughtful and insightful.
- Respect: The student was respectful of other students' opinions and ideas.
- The course instructor reserves the right to adjust the grading scale as needed. Please note that the total weight of all assignments and class activities equals 100%.

Individual Projects

Project #1: Continuous Integration using GitHub Actions of Python Data Science Project
Lead Instructor: Noah Gift
Requirements:

- The project structure must include the following files:
    - Jupyter Notebook with:
        - Cells that perform descriptive statistics using Polars or Panda.
        - Tested by using nbval plugin for pytest
    - Python Script performing the same descriptive statistics using Polars or Panda
    - lib.py file that shares the common code between the script and notebook
    - Makefile with the following:
        - Run all tests (must test notebook and script and lib)
        - Formats code with Python black
        - Lints code with Ruff
        - Installs code via: pip install -r requirements.txt
    - test_script.py to test script
    - test_lib.py to test library
    - Pinned requirements.txt
    - GitHub Actions performs all four Makefile commands with badges for each one in the README.md

Grading Rubric for Project #1: Continuous Integration using GitHub Actions of Python Data Science Project

Project Structure (15 points): Proper organization and inclusion of all required files.

- Jupyter Notebook: 4 points
- Python Script: 4 points
- lib.py file: 4 points
- Makefile: 3 points

Content of Jupyter Notebook and Python Script (20 points): Cells/scripts that perform descriptive statistics using Polars or Panda.

- Correct and efficient use of Polars or Panda: 10 points
- Accuracy of descriptive statistics: 10 points

Testing with nbval plugin for pytest (10 points): Correct usage and implementation of the nbval plugin for pytest in the Jupyter Notebook.

Shared code in lib.py (10 points): The lib.py file correctly shares the common code between the script and notebook.

Makefile Commands (15 points): The Makefile correctly includes and executes all required commands.

- Running all tests (notebook, script, lib): 5 points
- Formatting code with Python black: 5 points
- Linting code with Ruff: 5 points

Test Scripts (10 points): The test_script.py and test_lib.py files accurately and efficiently test the Python script and library.

- test_script.py: 5 points
- test_lib.py: 5 points

Requirements.txt (5 points): The requirements.txt file is correctly pinned and installed via pip install -r requirements.txt.

GitHub Actions (10 points): GitHub Actions correctly performs all Makefile commands and displays badges for each one in the README.md.

Demo Video (15 points): A 2-5 minute video is included, explaining the project and demonstrating its functionality. The video should be of high quality (both audio and visual), not exceed the given time limit, and be linked in the README via a private or public YouTube link.

- Clarity of explanation: 5 points
- Quality demonstration of the project: 5 points
- Quality of video and audio: 5 points

Total: 100 points

Individual Project #2: Rust CLI Binary with SQLite
Lead Instructor: Alfredo Deza
Requirements:

Your project should include the following:

- Rust source code: The code should comprehensively understand Rust's syntax and unique features.
- Use of GitHub Copilot: In your README, explain how you utilized GitHub Copilot in your coding process.
- SQLite Database: Include a SQLite database and demonstrate CRUD (Create, Read, Update, Delete) operations.
- Optimized Rust Binary: Include a process that generates an optimized Rust binary as a GitHub Actions artifact that can be downloaded.
- README.md: A file that clearly explains what the project does, its dependencies, how to run the program, and how GitHub Copilot was used.
- GitHub Actions: A workflow file that tests, builds, and lints your Rust code.
- Video Demo: A YouTube link in README.md showing a clear, concise walkthrough and demonstration of your CLI binary.

Grading Rubric for Project #2: Rust CLI Binary with SQLite

Rust Source Code (25 points): Your Rust source code is well-structured and demonstrates a clear understanding of Rust's syntax and unique features.

- Proper usage of Rust syntax: 8 points
- Effective error handling in Rust: 8 points
- Implementation of Rust's unique features: 9 points

SQLite Database (25 points): Demonstrates CRUD operations on the SQLite database.

- Create Operation: 6 points
- Read Operation: 6 points
- Update Operation: 6 points
- Delete Operation: 7 points

Use of GitHub Copilot (10 points):

- Explanation of the project: 3 points
- How to run the program: 3 points
- Dependencies and how to install them: 4 points

Optimized Rust Binary (10 points): Process included that generates an optimized Rust binary as a GitHub Actions artifact that can be downloaded.

README.md (10 points): The README.md file is clear and concise and guides the user on how to run the program.

- Explanation of the project: 3 points
- How to run the program: 3 points
- Dependencies and how to install them: 4 points

GitHub Actions (10 points): Your GitHub Actions file should test, build, and lint your Rust code correctly.

- Correct testing of Rust code: 3 points
- Correct building of Rust code: 3 points
- Correct linting of Rust code: 4 points

Demo Video (10 points): A 2-5 minute video explaining the project and demonstrating its functionality is included. The video should be high-quality (both audio and visual), not exceed the given time limit, and be linked in the README via a private or public YouTube link.

- Clarity of explanation: 3 points
- Quality demonstration of the project: 3 points
- Quality of video and audio: 4 points

Total: 100 points

Individual Project #3: Databricks ETL (Extract Transform Load) Pipeline

Lead Instructor: Noah Gift

Requirements

Your project should include the following:

- A well-documented Databricks notebook that performs ETL (Extract, Transform, Load) operations, checked into the repository.
- Usage of Delta Lake for data storage (show you understand the benefits).
- Usage of Spark SQL for data transformations.
- Proper error handling and data validation.
- Visualization of the transformed data.
- An automated trigger to initiate the pipeline.
- README.md: A file that clearly explains what the project does, its dependencies, how to run the program, and concludes with actionable and data-driven recommendations to a hypothetical management team.
- Video Demo: A YouTube link in README.md showing a clear, concise walkthrough and demonstration of your ETL pipeline, including the automated trigger and recommendations to the management team.

Grading Rubric

Your project will be graded on the following criteria:

- **Databricks ETL Pipeline (20 points):** Your Databricks notebook correctly extracts data, performs transformations, and loads the data into a final output.
    - Extract operation (7 points): Your notebook correctly extracts data from the source data.
    - Transform operation (7 points): Your notebook correctly transforms the extracted data.
    - Load operation (6 points): Your notebook correctly loads the transformed data into the destination data store.
- **Usage of Delta Lake (20 points):** Your ETL pipeline correctly utilizes Delta Lake for data storage.
    - Correct setup and usage of Delta Lake (10 points): Your notebook correctly sets up Delta Lake and uses it to store the transformed data.
    - Demonstrate you understand the unique capabilities of Delta Lake, i.e. time travel, metadata, or some other feature vs plain Data Lake.
    - Data validation checks (10 points): Your notebook includes data validation checks to ensure the quality of the data.
- **Usage of Spark SQL (20 points):** Your pipeline effectively utilizes Spark SQL for data transformations.
    - Correct use of Spark SQL (10 points): Your notebook correctly uses Spark SQL to perform the data transformations.
    - Effectiveness of data transformations (10 points): The data transformations are effective in cleaning and preparing the data for analysis.
- **Visualization and Conclusion (15 points):** Your project includes a visualization of the transformed data and a data-driven, actionable recommendation for a management team, both in the notebook and README.md.
    - Quality of data visualization (7 points): The visualization is clear, concise, and effective in communicating the results of the data analysis.
    - Actionable, data-driven recommendation (8 points): The recommendation is based on the results of the data analysis and is actionable by the management team.
- **README.md (10 points):** The README.md file is clear, concise, and guides the user on how to run the program and includes the conclusion and recommendation.

    - Clarity and completeness (5 points): The README.md file is easy to read and understand and includes all the information necessary to run the program.
    - Inclusion of conclusion and recommendation (5 points): The README.md file includes the conclusion and recommendation from the project.
- **Automated Trigger (10 points):** Your project includes an automated trigger to initiate the pipeline, demonstrated in the video.
    - Correct setup of the trigger (5 points): The trigger is correctly set up to initiate the pipeline on a schedule or event.
    - Effective demonstration in video (5 points): The video demonstrates the correct setup and usage of the trigger.
- **Demo Video (5 points):** A 2-5 minute video explaining the project and demonstrating its functionality is included. The video should be high-quality (both audio and visual), not exceed the given time limit, and be linked in the README via a private or public YouTube link.
    - Clarity of explanation (2 points): The video clearly explains the project and its functionality.
    - Quality demonstration of the project (2 points): The video demonstrates the project and its functionality in a clear and concise manner.
    - Quality of video and audio (1 point): The video is high-quality and easy to understand.

Total: 100 points

Individual Project #4: Auto Scaling Flask App Using Any Platform As a Service

Lead Instructor: Derek Wales

Requirements

For this assignment, you will build a publicly accessible auto-scaling container using Azure App Services and Flask. This is an easy way to build and deploy a scaleable web-hosted app and will allow you to apply your Flask knowledge from previous lessons.

Your project will be graded on the following criteria:

- **A README.md file (15 points):** This should clearly explain what the project does, its dependencies, how to run the program, and conclude with actionable and data-driven recommendations to a hypothetical management team.
    - Clarity and completeness of README: 10 points
    - Quality of conclusion and recommendation: 5 points
- **GitHub Repo (20 points)**: A complete GitHub Repo that contains all required scripts and documentation to run your application.
- **Flask App (20 points)**:
    - Functionality within Docker/(Platform): 10 points
    - Creativity/sophistication, full credit (all 20 points for this section) will be given to students who have a functioning embedded LLM within Flask, otherwise: 10 points.
- **Use of DockerHub (Or equivalent) (10 points)**: Hosting your functioning container on DockerHub.
- **Azure Web App (Or equivalent) (15 points):** Successfully deploying your container via Azure Web App to a public endpoint. This can be done either directly from Docker or through Azure container registry.
- **Video Demo (20 points):** A YouTube link in README.md showing a clear, concise (2-5min) walkthrough and demonstration of your application. The video should be high-quality (both audio and visual), not exceed the given time limit, and be linked in the README via a private or public YouTube link.
    - Clarity of explanation: 8 points
    - Quality demonstration of the project: 7 points
    - Quality of video and audio: 5 points

Total: 100 points

Requirements

Your team project should include the following:

- Microservice
    - Build a microservice that interfaces with a data pipeline. You can choose Python or Rust for development. The microservice should include logging and be containerized using the Distroless Docker image. A Dockerfile must be included in your repository.
- Load Test

    - The microservice must be capable of handling 10,000 requests per second. A load test verifying this performance should be included.
- Data Engineering
    - Your project should involve the use of a library specializing in data engineering such as Spark, Pandas, SQL, a vector database, or any other relevant library.
- Infrastructure as Code (IaC)
    - Your project must utilize an IaC solution for infrastructure setup and management. You can choose among AWS CloudFormation, AWS SAM, AWS CDK, or the Serverless Framework.
- Continuous Integration and Continuous Delivery (CI/CD)
    - Implement a CI/CD pipeline for your project. It could be through GitHub Actions or AWS Cloud Build or any other relevant tool.
- README.md
    - A comprehensive README file that clearly explains what the project does, its dependencies, how to run the program, its limitations, potential areas for improvement, and how AI Pair Programming tools (GitHub Copilot and one more tool of your choice) were used in your development process.
- Architectural Diagram
    - A clear diagram representing the architecture of your application should be included in your project documentation.
- GitHub Configurations
    - Your GitHub repository must include GitHub Actions and a .devcontainer configuration for GitHub Codespaces. This should make the local version of your project completely reproducible. The repository should also include GitHub Action build badges for install, lint, test, and format actions.
- Teamwork Reflection
    - Each team member should submit a separate 1-2 page management report reflecting on the team's functioning according to the principles discussed in your teamwork book. This report should not be part of the GitHub README but rather a separate document. It should include a peer evaluation in which each team member is graded on their performance, stating three positive attributes and three areas for improvement as the basis for the grade. Note that each student will share the teamwork reflection with their team and discuss it in a session before turning in the report. The outcome of this feedback session must be included in the report for full credit.

- Quantitative Assessment
    - The project must include a quantitative assessment of its reliability and stability. You must use data science fundamentals to describe system performance, e.g., average latency per request at different levels of requests per second (100, 1000, etc.). Think of the software system as a data science problem that needs to be described using data science principles.
- Demo Video
    - A YouTube link in README.md showing a clear, concise walkthrough and demonstration of your application, including the load test and system performance assessment.
- Team Size and Makeup
    - The team should consist of 3-4 people, ideally composed of 1-2 strong programmers and 1-2 quantitative storytellers.

Grading Rubric

- Microservice (20%)
    - Implementation of the microservice: 10 points
    - Use of logging: 5 points
    - Proper containerization with Distroless: 5 points
- Load Test (20%)
    - Successful load test at 10,000 requests/second: 20 points
- Data Engineering (10%)
    - Effective use of a data engineering library: 10 points
- Infrastructure as Code (IaC) (10%)
    - Correct setup and management of infrastructure using IaC: 10 points
- Continuous Integration and Continuous Delivery (CI/CD) (10%)
    - Proper implementation of a CI/CD pipeline: 10 points
- README.md (10%)
    - Clarity and comprehensiveness of README.md: 5 points
    - Explanation of AI Pair Programming tool usage: 5 points
- Architectural Diagram (5%)
    - Quality and clarity of the architectural diagram: 5 points
- GitHub Configurations (5%)
    - GitHub Actions + GitHub Codespaces .devcontainer configuration: 5 points
- Final Team Presentation (15%)

    - Quality and clarity of presentation: 10 points
    - Team's ability to effectively answer questions and discuss the project: 5 points
- Teamwork Reflection (5%)
    - Quality and sincerity of reflection: 3 points
    - Reflection includes peer evaluation with three positive attributes and three areas for improvement: 2 points

Total: 100%

Graded Items:

- ***Class Discussion***
    - In-Person Attendance Mandatory
    - 25% of Grade
    - Participation is self-reported in a Google sheet:
        - Each class must either note a good point and the name of the student or mention the point made in class
        - The final self-reported Google sheet must have approximately a 50/50 mix of both. See here for example:
        - The final grade will follow Nick Eubanks Rubric and the TAs and Instructor will validate the self-grading
        - **UPDATE: Since all students cannot be called upon in class, if you were not able to speak in class, please put your comments in Microsoft Teams if not called upon in class. Do this AFTER CLASS, since your comments will reflect what actually happened in class vs only what you read.**
- ***Weekly Mini-Project***
    - 25% of Grade
    - 80%-Python/20%-Rust (Core Language)
        - Rubric:
            - Must pass lint/test/format in GitHub Actions and proven via GitHub badge for each of these three actions
            - Small Tool or service: CLI/Serverless/Microservice
            - Average time 1-2 hours max

Grading Rubric for Weekly Mini-Project

See this template which MUST be included for ALL mini-projects. See each week for a breakout of the sub-component of the mini-project, as it contains directions on achieving full credit. For example, the Project Development and Language Use are subdivided into individual

mini-project requirements, which is approximately 50% of the grade of the project. ALL projects must have linting, testing, formatting, and a GitHub README; this is the other 50% of the grade.

Project Development (25 points): The project must be a small tool or service: CLI/Serverless/Microservice.

- Appropriate project scope and complexity: 10 points
- The functionality of the tool or service: 15 points

Language Use (25 points): Over 15 weeks, 80% of the mini-projects must be developed using only Python, and 20% must be developed using only Rust.

- Correct and efficient use of Python in Python-only projects: 15 points
- Correct and efficient use of Rust in Rust-only projects: 10 points

Linting, Testing, and Formatting (25 points): The project must pass lint, test, and format in GitHub Actions, as shown by the corresponding GitHub badges.

- Passing lint check: 8 points
- Passing tests: 8 points
- Proper code formatting: 9 points

GitHub README and Submission (25 points): The project must include a GitHub README with badges proving the Makefile actions work, and the README should be submitted into Microsoft Teams.

- Clear and accurate README with badges: 15 points
- Proper submission on Microsoft Teams (including pasted version of README): 10 points

T**otal: 100 points**

- ***Four Individual Project***
    - 25% of Grade
- ***Group Project***
    - 25% of Grade
        - Rubric:
            - Group Feedback Session Required as Part of Final Writeup
                - Each Team Member is required at the end of class to add three positives and three areas to improve on during a Zoom or in-person meeting.
                - Each student then writes a reflection statement about what they learned they could improve during the final project submission.
            - The final project must include the following criteria to pass:

- Demonstrated load-test
- README explains project, and includes architectural diagram
- Must pass test/lint/format in GitHub actions and have badge providing it
- Must continuously deploy