This document will give you comprehensive about the files uploaded in github as a part of submission. It also covers proposed solution.

Step 1:

- IDE Used: Visual studio and Jupyter notebook inside VS Code
- Create new venv and activate it using python or conda
- Using pip, you can install requirements.txt file (using command: pip install –r requirements.txt)

Step 2: Data Sample files and Data Generation Approach

- Inside your workspace – create two folders:
    - csv_files
    - json_files
- Paste all csv_files and json_files inside their respective folders.
- Note: Change the path as per your need in order to open/analyze files
- Json data files:
    - sample_diverse_dataset.json
        - Contains diverse herd information (yak name, age, health and behavior)
    - sample_json.json
        - Contains herd information (yak name, sex, age)
    - sample_order_data.json
        - Contains order information for 100 random customers
        - It shows customer name, order, date
    - sample_stock_data.json
        - Contains 100 stock samples (milk, skins)
    - customer_order_fullfillment_results.json
        - Contains order fulfillment status for 100 customers from sample_order_data.json with corresponding orders in sample_stock_data.json
- **CSVs data files:**
    - sample_diverse_dataset.csv – for querying using NLP agent
    - sample_json.csv – for querying using NLP agent
    - sample_order_data.csv – for querying using NLP agent
    - sample_stock_data.csv – for quering using NLP agent
- **Data Creation Approach:**
    - <u>REFERENCE FILE: data_generator.ipynb</u>
    - Based on the sample data provided in the tasks, 100 samples are created randomly for each json.
    - These json files are further utilized for every other tasks – core_functionality, AI models, Behavior analysis
    - CSV files are just for NLP Query Agents

Other Python Files

- **core_functionality_solution.ipynb**
  - this file solves core functionality needed for the tasks which includes:
    - Data Preprocessing
    - Stock and Herd Management Functions
    - Order Fulfillment Logic
  - Note: code logics can be referred via comments
- **anomaly_detection.ipynb**
  - ML model for anomaly detection
    - Model Used – Unsupervised Learning ML Model: IsolationForest
    - Reason to choose this model:
      - Effectiveness in Handling Outliers
      - Robustness to Noise and Irregularities
      - Efficient Computation
      - Parameter-Free Approach
      - Handling High-Dimensional Data
      - No Assumptions about Data Distribution
      - Effective in Unsupervised Learning Scenarios
  - **Note:** Testing has been done using inference data
  - **Note:** Model has been evaluated on accuracy, false positive rates
- behavior_analysis.ipynb
  - ML model for behavior analysis
  - **Note:**
    - The provided behavior analysis model attempts to predict yak behavior based on 'Age' and 'Health' attributes. While it's a step toward understanding yak behavior, fulfilling the statement to predict and comprehend their behavior over time requires a more comprehensive approach and more features. These features were not present as a part of the sample data
    - Additional relevant features, such as environment, diet, social interactions, or seasonal changes, might provide more comprehensive insights.
  - Model used: Binary Classification model where yak behavior is analyzed with age and health attributes
  - **Note:** Testing has been done on selecting random data samples from test data and check the model predictions (ground_truth_behavior vs model_predicted_behavior)
- final_app_agent_nlp.py
  - In the terminal, type streamlit run final_app_agent_nlp.py
  - **Note:** Make sure to use your own OPENAI API KEY from OPENAI
  - a webpage has been created to query different data related csv files
  - Functionality:
    - You can downloaded multiple CSVs at once

        - You can choose on what csv you need to perform query. Accordingly agent will provide you the answers
        - Agents' modules are used instead of chains modules of LangChain. Agents are not rule based models unlike chains where users have to define a set of prompts in order to get answers from their query. Surprisingly, agents handles this straightaway
        - **Note:** You can query any questions from any csv files
        - **Note:** You cannot update any value. I doubt if this functionality exists or not
        - **Note:**
            - My OpenAI free credits are finished, so I am not getting responses back from LLM. Let me fulfill it, I will update it if a demo needs to be shown
            - However, check simple and cool website design to upload and view csvs

Condensed steps for deploying the Yak Shop with AI/ML features on AWS:

Deployment Steps

(Providing various deployment options, considering AWS as prime):

Backend Setup:

- Choose EC2 or Lambda for hosting the Yak Shop backend.
- Install necessary dependencies (Python, Flask/Django)

Data Processing and Management:

- Develop code to process herd data based on elapsed time - DONE
- Create APIs for stock calculations and herd view after T days.

Order Fulfillment:

- Implement logic to fulfill orders, check stock availability - DONE
- Create HTTP endpoints for order fulfillment

AI/ML Integration:

- Model Preparation:
    - Serialize models for deployment and version control for organization
    - Optionally, containerize models using Docker for consistency
- Deployment on AWS:
    - Amazon Sage Maker (Anomaly Detection & Behavior Analysis):

    - Host models on Sage Maker endpoints, configuring instance types and scaling.
- AWS Lambda (Recommendation Engines)
    - Package models as Lambda functions and set up triggers for Yak Shop backend.
- API Gateway Integration:
    - Create API Gateway endpoints to link Yak Shop with deployed AI/ML models.
- Security and Access Control:
    - IAM Roles and Policies:
    - Define permissions and security measures using IAM for controlled access.
- Testing and Validation:
    - Unit and Integration Testing:
        - Validate model accuracy and performance through testing datasets and end-to-end scenarios.
- Monitoring and Maintenance:
    - Cloud Watch Metrics:
        - Set up monitoring through Cloud Watch for endpoint health, errors, and logging.

Database and Storage:

- Using Amazon S3 for file storage and RDS/Dynamo DB for dynamic data.

Networking and Security:

- By configure VPC, security groups for secure communication.

Deployment and Scaling:

- Setting up CI/CD pipelines for automated deployments.
- Implement auto-scaling and load balancing.

Documentation and Testing:

- Preparing documentation and user guides - DONE
- Performing extensive testing across functionalities – BASIC TESTING COVERED

Cost Optimization and Compliance:

- Optimizing costs using AWS Cost Explorer.