1. Creating a new Docker Image for the project:

```
PS Project-Folder> docker build -t claude-docker-ai-agents .
[+] Building 16.6s (11/11) FINISHED                                                                          docker:desktop-linux
 => [internal] load build definition from Dockerfile                                                                        0.0s
 => => transferring dockerfile: 721B                                                                                        0.0s
 => [internal] load metadata for docker.io/library/python:3.12-slim                                                         0.9s
 => [auth] library/python:pull token for registry-1.docker.io                                                               0.0s
 => [internal] load .dockerignore                                                                                           0.0s
 => => transferring context: 521B                                                                                           0.0s
 => [1/5] FROM docker.io/library/python:3.12-slim@sha256:7cebfa23db8cfd6cc1dd56986a3d2a4f3baee6bff3bf7eacbaaa4b42a89c6db5   0.1s
 => => resolve docker.io/library/python:3.12-slim@sha256:7cebfa23db8cfd6cc1dd56986a3d2a4f3baee6bff3bf7eacbaaa4b42a89c6db5   0.1s
 => [internal] load build context                                                                                           0.0s
 => => transferring context: 4.38kB                                                                                         0.0s
 => CACHED [2/5] WORKDIR /app                                                                                               0.0s
 => [3/5] COPY . /app                                                                                                       0.1s
 => [4/5] RUN pip install --no-cache-dir -r requirements.txt                                                               10.6s
 => [5/5] RUN mkdir /app/data                                                                                               0.4s
 => exporting to image                                                                                                      4.0s
 => => exporting layers                                                                                                     2.3s
 => => exporting manifest sha256:7a7a10640554697b56d500b1af6c40bb00d441538b62f0d26cd357a20e966cb4                           0.0s
 => => exporting config sha256:86a7826e59c1103e3dc10e161af250ac55701d825ff8677d93dc5db201c1ed50                             0.0s
 => => exporting attestation manifest sha256:5e8dd63b45f0b50ccd5503ace9d6d7e9c65ababfd246b4961722af49c85250f1               0.1s
 => => exporting manifest list sha256:1fd7e952b611b30dededddd192a69602d0502c54b0e13dcd2f5ee6bd1b524df9                      0.0s
 => => naming to docker.io/library/claude-docker-ai-agents:latest                                                           0.0s
 => => unpacking to docker.io/library/claude-docker-ai-agents:latest                                                        1.5s

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/j54zeuuaxo3k94a2z12mb0fi5

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview
```

2. Creating Docker Container from the docker Image:

```
# Open Terminal in Project Folder

PS %user%> # Set the path to your data directory
PS %user%> $path = "C:\GitHub_Repos\Test_Folder_CSV_dataset"
PS %user%> # Docker run command
PS %user%> docker run -it `
>>     --rm `
>>     -v "${path}:/app/data" `
>>     -e ANTHROPIC_API_KEY `
>>     --name ai-agents-container `
>>     claude-docker-ai-agents
```

```
# Alternative command using the current working directory
# docker run -it `
#     --rm `
#     -v "${PWD}/data:/app/data" `
#     -e ANTHROPIC_API_KEY `
#     --name ai-agents-container `
#     claude-docker-ai-agents

Enter your Anthropic API key: claude_api_key
Enter the CSV file name:input.csv
Enter the number of rows to generate: 110
Analyzing...Calling Analyzer Agent
Analysis Result:
 Here's a concise summary of the dataset:

1. Formatting of the dataset:
  - The data is presented in CSV (Comma-Separated Values) format.
  - The first row contains column headers.
  - Each subsequent row represents a single exercise.
  - There are 8 columns, separated by commas.
  - There are no quotation marks around the values.
  - Some cells contain spaces (e.g., in exercise names or equipment).

2. Dataset representation and column meanings:
  - Exercise: Name of the exercise (string)
  - Category: Type of exercise (string)
  - Equipment: Required equipment, if any (string)
  - Difficulty: Level of difficulty (string)
  - MuscleGroup: Primary muscle group targeted (string)
  - CaloriesBurnedPer30Min: Estimated calories burned in 30 minutes (integer)
  - RecommendedSets: Suggested number of sets (integer)
  - RecommendedReps: Suggested number of repetitions or duration (mixed: integer or string with unit)

3. How new data should look:
  - New entries should follow the same structure as existing rows.
  - Exercise names should be capitalized and use spaces between words.
  - Category should be one of the existing categories (e.g., Bodyweight, Strength, Cardio, Plyometric).
  - Equipment should be "None" if not required, or the specific equipment name.
  - Difficulty should be Beginner, Intermediate, or Advanced.
  - MuscleGroup should be specific (e.g., Chest, Legs) or "Full Body".
  - CaloriesBurnedPer30Min should be a whole number.

Generating...Calling Generator Agent
Generating 25 rows
```

```
Generated 25 rows out of 110 desired rows
Generating 25 rows
Generated 50 rows out of 110 desired rows
Generating 25 rows
Generated 75 rows out of 110 desired rows
Generating 25 rows
Generated 100 rows out of 110 desired rows
Generating 10 rows
Generated 110 rows out of 110 desired rows
Generated CSV file saved to /app/data/output.csv
{'/app/data/output.csv'}
PS %user%>
```

3. Publishing the docker for public use on DockerHub:

```
# Open Terminal in Project Folder

PS %user%> docker login
Authenticating with existing credentials...
Login Succeeded
PS %user%> docker tag claude-docker-ai-agents jaysinghvi/claude-data-gen-agent:latest
PS %user%> docker push jaysinghvi/claude-data-gen-agent:latest
The push refers to repository [docker.io/jaysinghvi/claude-data-gen-agent]
dc6b21a4b822: Pushed
a5db25c196df: Pushed
827c7d589fb0: Pushed
8826814689ee: Pushed
f11c1adaa26e: Pushed
e089ddb13a52: Pushed
955e715c3964: Pushed
59eca9927c21: Pushed
f2d21d0ffa2c: Pushed
53d35ac689a9: Pushed
latest: digest: sha256:5c7fd4690a53e25f219045c2eed375577c4e0097d4b2a9059a30847d6530cdae size: 856
```

4. Testing the Dockerhub image:

```
# Delete all the tested docker images and containers

# Open Terminal in Project Folder

PS %user%> docker pull jaysinghvi/claude-data-gen-agent:latest
latest: Pulling from jaysinghvi/claude-data-gen-agent
f2d21d0ffa2c: Already exists
53d35ac689a9: Already exists
```

```
a5db25c196df: Already exists
8826814689ee: Already exists
Digest: sha256:5c7fd4690a53e25f219045c2eed375577c4e0097d4b2a9059a30847d6530cdae
Status: Downloaded newer image for jaysinghvi/claude-data-gen-agent:latest
docker.io/jaysinghvi/claude-data-gen-agent:latest

What's next:
    View a summary of image vulnerabilities and recommendations → docker scout quickview jaysinghvi/claude-data-gen-agent:latest
PS %user%> $path = "C:\GitHub_Repos\Test_Folder_CSV_dataset" # Path of sample data
PS %user%> docker run -it `
>>     --rm `
>>     -v "${path}:/app/data" `
>>     -e ANTHROPIC_API_KEY `
>>     --name ai-agents-container `
>>     jaysinghvi/claude-data-gen-agent
Enter your Anthropic API key: claude_api_key
Enter the CSV file name:input.csv
Enter the number of rows to generate: 55
Analyzing...Calling Analyzer Agent
Analysis Result:
 Here's a concise summary of the dataset:

1. Formatting of the dataset:
   - The data is presented in CSV (Comma-Separated Values) format.
   - The first row contains column headers.
   - Each subsequent row represents a single exercise.
   - There are 8 columns, separated by commas.
   - There are no quotation marks around the values.
   - Some cells contain spaces (e.g., in exercise names or equipment).

2. Dataset representation and column meanings:
   - Exercise: Name of the exercise (string)
   - Category: Type of exercise (string)
   - Equipment: Required equipment, if any (string)
   - Difficulty: Level of difficulty (string)
   - MuscleGroup: Primary muscle group targeted (string)
   - CaloriesBurnedPer30Min: Estimated calories burned in 30 minutes (integer)
   - RecommendedSets: Suggested number of sets (integer)
   - RecommendedReps: Suggested number of repetitions or duration (mixed: integer or string with unit)

3. How new data should look:
   - New entries should follow the same structure as existing rows.
   - Exercise names should be capitalized and use spaces between words.
   - Category should be one of the existing categories (e.g., Bodyweight, Strength, Cardio, Plyometric).
   - Equipment should be "None" if not required, or the specific equipment name.
```

```
  - Difficulty should be Beginner, Intermediate, or Advanced.
  - MuscleGroup should be specific (e.g., Chest, Legs) or "Full Body".
  - CaloriesBurnedPer30Min should be a whole number.

Generating...Calling Generator Agent
Generating 25 rows
Generated 25 rows out of 55 desired rows
Generating 25 rows
Generated 50 rows out of 55 desired rows
Generating 5 rows
Generated 55 rows out of 55 desired rows
Generated CSV file saved to /app/data/output.csv
{'/app/data/output.csv'}
PS %user%>
```