National College of Ireland

Configuration Manual

MSc Research Project
Cloud Computing

Franklin Ebuka Onyia
Student ID: X21221600

School of Computing
National College of Ireland

Supervisor: Rejwanul Haque

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	…Franklin Ebuka Onyia……
Student ID:	……x21221600……
Programme:	……Msc Cloud Computing…… **Year:** ……2023/2024……..
Module:	……Research Project……
Lecturer:	……Rejwanul Haque……
Submission Due Date:	……14/12/2023……
Project Title:	Data Exposure Analysis of Misconfigured S3 Buckets: A Quantitative Approach
Word Count:	……535…… **Page Count:** ……11……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: …Franklin Ebuka Onyia……

Date: ……14/12/2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only

Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

Franklin Ebuka Onyia
Student ID: 21221600

1. **Introduction**
    - **Project Overview**: This manual aids in running code to analyze misconfigured S3 buckets using Python in Google Colab.
    - **Audience**: For anyone that wants to run the analysis.

2. **System Requirements**
    - **Hardware**: computer with stable internet availability.
    - **Software**: use google chrome Web browser for accessing Google Colab.

3. **Environment Setup**
    - Use Google Colab in your browser.
    - Upload the Python notebook file: Click on File > Upload notebook.

4. **Configuration of Data**
    - Follow instructions in the notebook to set up directories and load data.
    - Run the blocks of code and setup the data.

5. **Execution of Code**
    - Execute the code blocks in sequence as shown in the notebook.
    - Ensure each block runs successfully before moving to the next.

6. **Analysis and Visualization of Data**
    - First Run the data cleaning, transformation, and analysis.
    - Visualization tools like Matplotlib and Seaborn are used within the notebook for data representation.

7. **Model Evaluation and Regression Analysis**
    - Follow the code for correlation and regression analysis.
    - Evaluate the outputs with respect to the instructions in the notebook.

8. **Screenshots and Visual Guide**
    How to Run Code Blocks Below: click the run button for each step so the code can be ran successfully.

Extracting and Saving AWS S3 Bucket Information Using Grayhat Warfare API

```python
import requests
import json

# API key and endpoint setup
api_key = "454286eef0b340f0b6b194152698d786"
url = "https://buckets.grayhatwarfare.com/api/v2/files"
headers = {"Authorization": f"Bearer {api_key}"}
params = {
    "type": "aws",  # Filter for Amazon S3 buckets
    "start": 0,     # Start from the beginning of the result set
    "limit": 1000   # Limit of 1000 results
}

# Making the API request
response = requests.get(url, headers=headers, params=params)

# Processing the response
if response.status_code == 200:
    data = response.json()
    file_info = []

    # Extracting file information and filtering for AWS buckets
    for file in data.get("files", []):
        if file.get("type") == "aws":  # Filter to include only AWS buckets
            file_info.append({
```

Converting AWS S3 Bucket Data from JSON to CSV Format

```python
import csv
import json

# Path to the JSON file
json_file_path = '/content/aws_bucket_file_info.txt'

# Function to load JSON data from a file
def load_json(file_path):
    with open(file_path, 'r') as file:
        return json.load(file)

# Load the JSON data
data = load_json(json_file_path)

# Path for the output CSV file in the accessible directory
csv_file_path = '/content/bucket_file_info.csv'

# Writing data to CSV
with open(csv_file_path, mode='w', newline='', encoding='utf-8') as file:
    writer = csv.writer(file)

    # Write the headers
    writer.writerow(["File ID", "Bucket URL", "Bucket ID", "Filename", "Full Path", "File URL", "Size", "Type", "Last Modified"])

    # Write the data
    for item in data:
        writer.writerow([item["File ID"], item["Bucket URL"], item["Bucket ID"], item["Filename"], item["Full Path"], item["File URL"]

csv_file_path

```

Data Preparation

Data Cleaning

```
[ ]
import pandas as pd
from datetime import datetime

# Load the dataset
df = pd.read_csv('/content/bucket_file_info.csv')

# Convert 'Last Modified' from Unix timestamp to readable date format
df['Last Modified'] = pd.to_datetime(df['Last Modified'], unit='s')

# Ensure 'Size' is an integer format if it's not already
df['Size'] = df['Size'].astype(int)

# Save the cleaned data back to a new CSV file
df.to_csv('/content/bucket_file_info_cleaned.csv', index=False)

# Display the first few rows to verify
print(df.head())

```

Data Transformation

```
[ ]
import pandas as pd

# Load the dataset
df = pd.read_csv('/content/bucket_file_info_cleaned.csv')

# Extract file extensions and save in a new column 'File Extension'
df['File Extension'] = df['Filename'].str.extract(r'(\.[^.]+)$')

# Save the updated dataframe back to a new CSV file
df.to_csv('/content/bucket_file_info_transformed.csv', index=False)

# Display the first few rows to verify
print(df.head())

```

Character of Exposed Data Analysis

Content Type Analysis

```
[ ]
```

```python
import pandas as pd
import matplotlib.pyplot as plt

# Step 1: Data Loading
df = pd.read_csv('/content/bucket_file_info_transformed.csv')

# Step 2: Data Inspection
# Checking for inconsistencies in 'File Extension' column
if df['File Extension'].isnull().any():
    print("There are missing file extensions.")

# Step 3: Data Categorization
# Grouping data by 'File Extension'
file_types = df.groupby('File Extension').size().reset_index(name='Count')

# Step 4: Frequency Calculation
# Sorting the file types by count
file_types = file_types.sort_values('Count', ascending=False)

# Step 5: Data Visualization
# Bar chart to display the frequency distribution of file types
plt.figure(figsize=(10, 8))
plt.bar(file_types['File Extension'], file_types['Count'], color='skyblue')
plt.xlabel('File Extension')
plt.ylabel('Frequency')
plt.title('Frequency Distribution of File Types')
plt.xticks(rotation=45)
plt.tight_layout()  # Adjusts plot to ensure everything fits without overlappin
```

Exposure Timeframe Analysis

Last Modified Date Analysis

```python
import pandas as pd
from datetime import datetime

# Load the data
data = pd.read_csv('/content/bucket_file_info_transformed.csv')

# Convert 'Last Modified' to datetime
data['Last Modified'] = pd.to_datetime(data['Last Modified'])

# Calculate the time elapsed since the last modification
current_time = datetime.now()
data['Time Elapsed'] = current_time - data['Last Modified']

# Calculate mean, median, and mode of 'Time Elapsed'
mean_elapsed = data['Time Elapsed'].mean()
median_elapsed = data['Time Elapsed'].median()
mode_elapsed = data['Time Elapsed'].mode()[0] # Taking the first value if there are multiple modes

# Print out the calculated statistics
print(f"Mean Time Elapsed: {mean_elapsed}")
print(f"Median Time Elapsed: {median_elapsed}")
print(f"Mode Time Elapsed: {mode_elapsed}")

# Save the DataFrame with the new calculations
output_file_path = '/content/bucket_file_info_with_elapsed_time.csv'
data.to_csv(output_file_path, index=False)

```

A table summarizing the statistics

```python
import pandas as pd
import matplotlib.pyplot as plt
from datetime import datetime

# Assuming 'data' is your DataFrame and it already contains 'Time Elapsed' as a timedelta
# Calculate the statistics again
mean_elapsed = data['Time Elapsed'].mean()
median_elapsed = data['Time Elapsed'].median()
mode_elapsed = data['Time Elapsed'].mode()[0]

# Create a summary DataFrame
summary_df = pd.DataFrame({
    'Statistic': ['Mean', 'Median', 'Mode'],
    'Time Elapsed': [mean_elapsed, median_elapsed, mode_elapsed]
})

# Display the summary table
print(summary_df)

```

A histogram and a boxplot to visualize the distribution of days since the last modification

```python
# Histogram of time elapsed
plt.figure(figsize=(10,6))
data['Time Elapsed'].dt.days.plot(kind='hist', bins=30)
plt.title('Histogram of Time Elapsed Since Last Modified')
plt.xlabel('Days')
plt.ylabel('Frequency')
plt.show()

# Boxplot of time elapsed
plt.figure(figsize=(10,6))
data['Time Elapsed'].dt.days.plot(kind='box')
plt.title('Boxplot of Time Elapsed Since Last Modified')
plt.ylabel('Days')
plt.grid(True)
plt.show()

```

Size Analysis

```python
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import pearsonr
from datetime import datetime

# Load the dataset
df = pd.read_csv('/content/bucket_file_info_transformed.csv')

df['Last Modified'] = pd.to_datetime(df['Last Modified'], format='%Y-%m-%d %H:%M:%S')


# Calculate the number of days since the oldest file for each entry
oldest_date = df['Last Modified'].min()
df['Days Since Oldest'] = (df['Last Modified'] - oldest_date).dt.days

# Summary statistics for file sizes
size_stats = df['Size'].describe()

# Correlational analysis
correlation = pearsonr(df['Size'], df['Days Since Oldest'])[0]

# Create scatter plot for the correlational analysis
plt.figure(figsize=(10, 6))
plt.scatter(df['Days Since Oldest'], df['Size'])
plt.title('File Size vs Days Since Oldest File')
plt.xlabel('Days Since Oldest File')
plt.ylabel('File Size (bytes)')
plt.yscale('log')  # Using a log scale due to wide range of file sizes
plt.grid(True)
```

Correlation and Regression Analysis

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Load the dataset
df = pd.read_csv('/content/bucket_file_info_transformed.csv')

# Convert 'Last Modified' to datetime and 'Size' to numeric if necessary
df['Last Modified'] = pd.to_datetime(df['Last Modified'])
df['Size'] = pd.to_numeric(df['Size'])

# Calculate 'Exposure Time' in days
df['Exposure Time'] = (pd.to_datetime('today') - df['Last Modified']).dt.days

# EDA: Basic stats and visualizations
print(df.describe())
sns.pairplot(df[['Size', 'Exposure Time']])
plt.show()

# Correlation Analysis
correlation_matrix = df.corr()
print(correlation_matrix)
```

9. **Troubleshooting and Common Issues**
    - In case of any problem, please re-check code syntax, also make sure all libraries are installed properly.
    - In case of any issue, please, checkout Google Colab's help resources or Python programming learning sites.

10. **Security Considerations**
    - Be sure you use libraries that are from their official website.
    - Also be careful with Security and privacy issues, most especially when you have used sensitive information.