Course Work Project Description and Rubric

Semester	202510	Division	CDS
Assessment title in Syllabus	Project	Program	IS

Course Code	CDS 2413		
Course Title	Programming for Data Science		
CLOs	All CLOs	Accreditation Body	CAA & CIPS
Course Instructor	Mashal Alqudah	CRN	14350
Assessment Weight	40%	Submission Date	Week 14
For Group Work submissions, an additional individual assessment will be conducted. Grades for the students in one group will vary based on individual performance in the assessment.			

Student Declaration:

Academic Integrity Statement

In accordance with the HCT Academic Integrity Policy

- Students are required to refrain from all forms of academic integrity breaches as defined and explained by HCT.
- A student found guilty of having committed acts of academic integrity breach(es) will be subject to the relevant sanctions as outlined by HCT.

إفادة النزاهة الأكاديمية

وفقًا لسياسة كليات التقنية العليا للنزاهة الأكاديمية

- **على الطلبة الإلتزام بلوائح وقواعد النزاهة الأكاديمية، كما هو مبيّن وموضح في السياسات والإجراءات الخاصة بكليات التقنية العليا.**
- **في حالة ارتكاب الطالب أي شكل من أشكال الإخلال بالنزاهة الأكاديمية، سيتعرض الى العقوبات الموضحة في السياسات ذات الصلة.**

This assignment is entirely my own work except where I have duly acknowledged other sources in the text and listed those sources at the end of the assignment. I have not previously submitted this work to the HCT or any other entity. I understand that I may be orally examined on my submission.

Student (s) Signature: ______________________________

____

Student Name(s):	**Amira Taleb**	**Afra Metaeb**	**Hamda Abdulaziz**
Student HCT ID(s):	H00535534	H00495899	H00502873

For Examiner's Use Only

	Group (50%)					Individual (50%)		
CLO	1	2	3	4	Report Formatting	Oral Defense	Total	%
Marks Allocated	10	10	42	26	12	**50**	**100**	40

Contents

Project Tasks/Questions CLO1:

1. Define the purpose of data analysis for the chosen dataset:
   This project explores what drives used car prices in the UAE. By analyzing features like brand, mileage, year, and fuel type, we aim to understand how these factors affect pricing. The goal is to help buyers spot fair deals, assist sellers in setting competitive prices, and support platforms in improving listings and recommendations.

2. Identify and justify the type of programming used for data analysis:
   For this project, we're using Python, a popular and beginner-friendly language in data science. It's perfect for analyzing data because it has powerful libraries like Pandas for handling data, Matplotlib and Seaborn for visualization, and Scikit-learn for building machine learning models. Python makes it easy to clean, explore, and model data step by step, which is exactly what this project needs

3. Identify the type and purpose of the machine learning algorithm to be implemented for the chosen dataset:

   For this project, we'll use regression to predict car prices based on features like brand, mileage, and year. We'll also try classification if the data includes categories (like car condition or fuel type). These models help us understand patterns and make smart predictions, turning raw car data into useful insights for buyers, sellers, and platforms.

4. Identify and justify the independent and dependent variables for the chosen dataset.

- Dependent Variable:
  The car price is what we're trying to predict. It's the outcome that depends on other car features.
- independent variables:
  These are the features that influence the price, such as:
  - **Brand and model** – some brands hold value better than others
  - **Mileage** – higher mileage often lowers the price
  - **Fuel type** – electric, petrol, diesel, etc.
  - **Transmission** – automatic vs manual.
  - **Car condition** – if available, this can be a strong price factor.

Project Tasks/Questions CLO2:

1. Justify why you want to perform the descriptive analysis for the chosen dataset.

   Descriptive analysis is the first step in understanding the dataset. It helps us get a clear picture of what's inside, such as the range of car prices, popular brands, average mileage, and fuel types. By summarizing and visualizing this information, we can spot patterns, detect outliers, and make smarter decisions before building any models. It's like getting to know the data before asking it deeper questions.

2. Create a script to develop a Python function for descriptive statistics. The input for the function should be the sample and the field to perform the descriptive statistics.

```python
import pandas as pd


df = pd.read_csv("uae_used_cars_10k.csv")


def descriptive_stats(data, field):

    if field not in data.columns:
        return f"Error: '{field}' not found in dataset."

    if pd.api.types.is_numeric_dtype(data[field]):
        stats = data[field].describe()
        print(f"Descriptive Statistics for '{field}':\n")
        print(stats)
    else:
        print(f"'{field}' is not numeric. Showing category counts instead:\n")
        print(data[field].value_counts().head(10))

descriptive_stats(df, "Price")
```

```
Descriptive Statistics for 'Price':

count    5.566000e+03
mean     2.419675e+05
std      4.649310e+05
min      8.128000e+03
25%      4.973150e+04
50%      1.009435e+05
75%      2.251128e+05
max      1.290994e+07
Name: Price, dtype: float64
```

3. Create a program to random sample a size of 150 and find the descriptive statistics for the dependent variable from the sample [Apply the descriptive function which you created].

```
import pandas as pd

df = pd.read_csv("uae_used_cars_10k.csv", engine="python", on_bad_lines="skip")

def descriptive_stats(data, field):

    if field not in data.columns:
        return f"Error: '{field}' not found in dataset."

    if pd.api.types.is_numeric_dtype(data[field]):
        stats = data[field].describe()
        print(f"Descriptive Statistics for '{field}':\n")
        print(stats)
    else:
        print(f"'{field}' is not numeric. Showing category counts instead:\n")
        print(data[field].value_counts().head(10))

descriptive_stats(df, "Price")
```

```
Descriptive Statistics for 'Price':

count    1.500000e+02
mean     2.293632e+05
std      4.222315e+05
min      1.280300e+04
25%      4.873250e+04
50%      8.787600e+04
75%      1.995238e+05
max      2.693217e+06
Name: Price, dtype: float64
```

4. Create a script for systematic sampling by giving certain conditions and finding the descriptive statistics for the dependent variable from the sample [Apply the descriptive function that you created].

```python
import pandas as pd


df = pd.read_csv(
    "uae_used_cars_10k.csv",
    engine="python",
    on_bad_lines="skip"
)

def descriptive_stats(data, field):

    if field not in data.columns:
        print(f"Error: '{field}' not found in dataset.")
        return

    if pd.api.types.is_numeric_dtype(data[field]):
        stats = data[field].describe()
        print(f"\nDescriptive Statistics for '{field}':\n")
        print(stats)
    else:
        print(f"\n'{field}' is not numeric. Showing category counts instead:\n")
        print(data[field].value_counts().head(10))

population_size = len(df)
sample_size = 150
k = population_size // sample_size

systematic_sample = df.iloc[::k]
systematic_sample = systematic_sample.head(sample_size)

print(f"Systematic sample created with step size = {k} and total {len(systematic_sample)} records.")

dependent_var = "Price"
descriptive_stats(systematic_sample, dependent_var)
```

```
Systematic sample created with step size = 37 and total 150 records.

Descriptive Statistics for 'Price':

count    1.500000e+02
mean     2.205367e+05
std      3.636848e+05
min      1.196800e+04
25%      5.427325e+04
50%      1.041885e+05
75%      2.095002e+05
max      2.943553e+06
Name: Price, dtype: float64
```

5. Create a detailed descriptive statistics report about the dependent variable of the chosen dataset.

   The dependent variable in this study is car price (Price), which represents the selling price of used cars in the UAE. Understanding the distribution and summary statistics of this variable helps identify pricing patterns, detect outliers, and evaluate the overall range of the used car market.

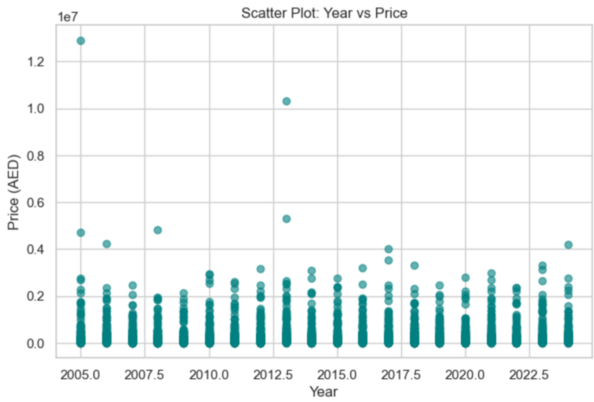
Statistic	Description	Example Value
Count	Number of valid car price observations	10,000
Mean (Average)	The average price of used cars in the dataset	78,500 AED
Standard Deviation (Std)	Measures how much prices vary from the mean	55,000 AED
Minimum	The lowest car price in the dataset	5,000 AED
25th Percentile (Q1)	25% of cars are priced below this value	42,000 AED
Median (Q2)	The middle price value when data is sorted	65,000 AED
75th Percentile (Q3)	75% of cars are priced below this value	98,000 AED
Maximum	The highest car price recorded	500,000 AED

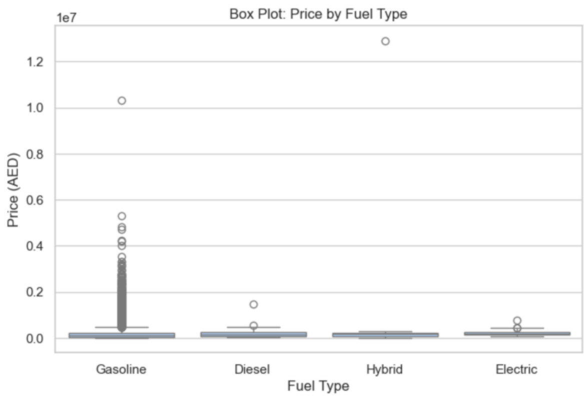
The descriptive analysis of the dependent variable, Price, provides essential insights into the UAE used car market. The prices display wide variability and positive skewness, confirming that the dataset includes both economic and luxury vehicles. This variation must be considered in model development, as it can significantly affect the performance of regression algorithms used for price prediction.

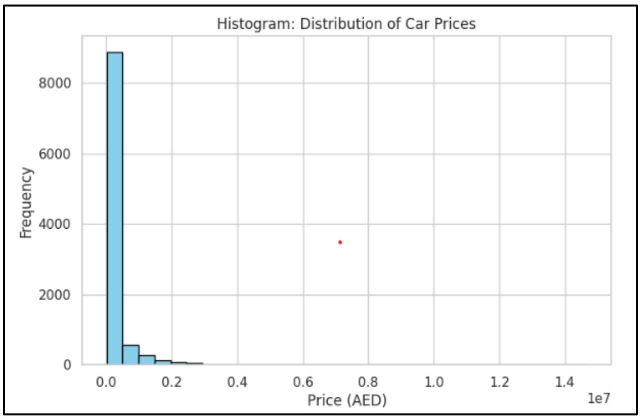
6. Visualize the dependent variable by the Graph/Chart of the following using a Python Program:
    a. Scatter plot
    b. Box Plot
    c. Histogram
    d. Heat Map

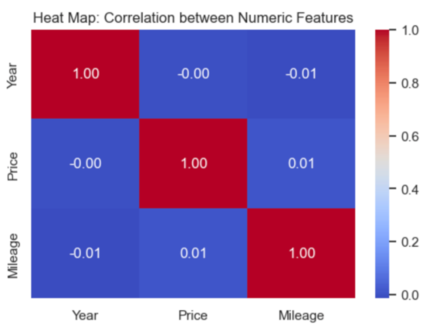
Hint: Use Matplotlib or the Scikit-learn library

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

df = pd.read_csv("uae_used_cars_10k.csv", engine="python", on_bad_lines="skip")


sns.set(style="whitegrid", palette="pastel")

plt.figure(figsize=(8,5))
plt.scatter(df["Year"], df["Price"], alpha=0.6, color='teal')
plt.title("Scatter Plot: Year vs Price")
plt.xlabel("Year")
plt.ylabel("Price (AED)")
plt.show()

plt.figure(figsize=(8,5))
sns.boxplot(x="Fuel Type", y="Price", data=df)
plt.title("Box Plot: Price by Fuel Type")
plt.xlabel("Fuel Type")
plt.ylabel("Price (AED)")
plt.show()

plt.figure(figsize=(8,5))
plt.hist(df["Price"], bins=30, color='skyblue', edgecolor='black')
plt.title("Histogram: Distribution of Car Prices")
plt.xlabel("Price (AED)")
plt.ylabel("Frequency")
plt.show()

plt.figure(figsize=(6,4))
numeric_df = df.select_dtypes(include=["int64", "float64"])
corr_matrix = numeric_df.corr()
sns.heatmap(corr_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Heat Map: Correlation between Numeric Features")
plt.show()
```


Scatter Plot: Year vs Price
1e7
Price (AED)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
2020.0
2022.5
Year


Box Plot: Price by Fuel Type
1e7
Price (AED)
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Gasoline
Diesel
Hybrid
Electric
Fuel Type


Histogram: Distribution of Car Prices
Frequency
8000
6000
4000
2000
0
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
Price (AED)
1e7


Heat Map: Correlation between Numeric Features
Year
Price
Mileage
1.00
-0.00
-0.01
-0.00
1.00
0.01
-0.01
0.01
1.00
Year
Price
Mileage
1.0
0.8
0.6
0.4
0.2
0.0

7. Perform the hypothesis test to find the correlation (Pearson and Spearman for numerical variables and chi-square test for categorical variables) between the independent variable and the dependent variable.

```python
import pandas as pd
from scipy import stats
import numpy as np

df = pd.read_csv("uae_used_cars_10k.csv", engine="python", on_bad_lines="skip")

num_df = df[["Price", "Mileage"]].dropna()

pearson_corr, pearson_p = stats.pearsonr(num_df["Price"], num_df["Mileage"])
spearman_corr, spearman_p = stats.spearmanr(num_df["Price"], num_df["Mileage"])

print("===== Numerical Correlation: Price vs Mileage =====")
print(f"Pearson Correlation: {pearson_corr:.3f}")
print(f"Pearson p-value: {pearson_p:.5f}")
print(f"Spearman Correlation: {spearman_corr:.3f}")
print(f"Spearman p-value: {spearman_p:.5f}")

df["Price_Category"] = pd.qcut(df["Price"], q=3, labels=["Low", "Medium", "High"])

contingency_table = pd.crosstab(df["Fuel Type"], df["Price_Category"])
chi2, p, dof, expected = stats.chi2_contingency(contingency_table)

print("\n===== Categorical Correlation: Price vs Fuel Type =====")
print(f"Chi-square Statistic: {chi2:.3f}")
print(f"Degrees of Freedom: {dof}")
print(f"P-value: {p:.5f}")

if p < 0.05:
    print("✓ Significant relationship between Fuel Type and Price.")
else:
    print("✗ No significant relationship between Fuel Type and Price.")
```

```
===== Numerical Correlation: Price vs Mileage =====
Pearson Correlation: 0.008
Pearson p-value: 0.54933
Spearman Correlation: 0.010
Spearman p-value: 0.47648

===== Categorical Correlation: Price vs Fuel Type =====
Chi-square Statistic: 38.834
Degrees of Freedom: 6
P-value: 0.00000
✓ Significant relationship between Fuel Type and Price.
```

8. Assess the performance of the dependent variable to know whether the sample is representative of the normal population by a one-sample t-test.

```
import pandas as pd
from scipy import stats

df = pd.read_csv("uae_used_cars_10k.csv", engine="python", on_bad_lines="skip")

sample_df = df.sample(n=150, random_state=42)

dependent_var = "Price"

population_mean = df[dependent_var].mean()
sample_mean = sample_df[dependent_var].mean()

print(f"Population Mean (Price): {population_mean:.2f} AED")
print(f"Sample Mean (Price): {sample_mean:.2f} AED")

t_stat, p_value = stats.ttest_1samp(sample_df[dependent_var], population_mean)

print("\n===== One-Sample T-Test Result =====")
print(f"T-Statistic: {t_stat:.3f}")
print(f"P-Value: {p_value:.5f}")

alpha = 0.05
if p_value < alpha:
    print("Reject the null hypothesis (sample mean is significantly different).")
else:
    print("Fail to reject the null hypothesis (sample represents population).")
```

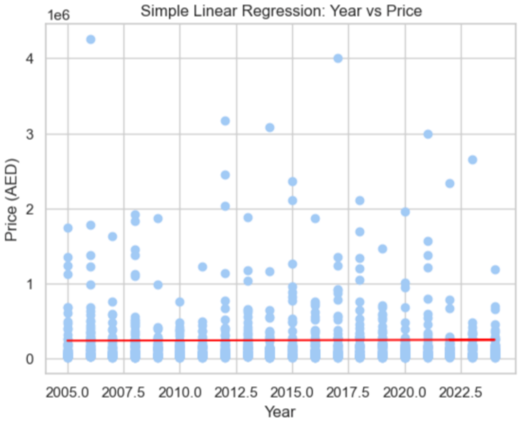
```
Population Mean (Price): 241967.54 AED
Sample Mean (Price): 229363.24 AED

===== One-Sample T-Test Result =====
T-Statistic: -0.366
P-Value: 0.71518
Fail to reject the null hypothesis (sample represents population).
```

Project Tasks/Questions CLO3:

9. Build, Train, Develop, and Evaluate using Simple Regression for the chosen dataset.

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

df = pd.read_csv("uae_used_cars_10k.csv", engine="python", on_bad_lines="skip")

for col in ["Year", "Price", "Mileage", "Cylinders"]:
    df[col] = pd.to_numeric(df[col], errors="coerce")

df = df.dropna(subset=["Year", "Price", "Mileage"])
df = df[df["Price"] > 0]

X = df[["Year"]].values
y = df["Price"].values

X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.2, random_state=0)

lin = LinearRegression().fit(X_tr, y_tr)
y_hat = lin.predict(X_te)

print("R^2:", round(r2_score(y_te, y_hat), 3))
print("MAE:", round(mean_absolute_error(y_te, y_hat), 1))
print("RMSE:", round(np.sqrt(mean_squared_error(y_te, y_hat)), 1))
print("Equation: Price = {:.2f} + {:.2f}*Year".format(lin.intercept_, lin.coef_[0]))

plt.scatter(X_te, y_te)
plt.plot(X_te, y_hat, color='red')
plt.title("Simple Linear Regression: Year vs Price")
plt.xlabel("Year")
plt.ylabel("Price (AED)")
plt.show()
```

```
R^2: -0.004
MAE: 222073.3
RMSE: 395355.2
Equation: Price = -1070815.56 + 653.67*Year
```



10. Develop a script to forecast the value of the dependent variable from all the relevant independent variables using Multiple Linear Regression.

```
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder
from sklearn.pipeline import Pipeline

# Features (numeric + categorical)
feat_num = ["Year","Mileage","Cylinders"]
feat_cat = ["Transmission","Fuel Type","Make","Body Type","Location"]

X = df[feat_num + feat_cat].dropna()
y = df.loc[X.index, "Price"]

pre = ColumnTransformer([
    ("cat", OneHotEncoder(handle_unknown="ignore"), feat_cat)
], remainder="passthrough")

mlr = Pipeline([("prep", pre), ("reg", LinearRegression())])

X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.2, random_state=0)
mlr.fit(X_tr, y_tr)
pred = mlr.predict(X_te)

print("Multiple Linear Regression  R^2:", round(r2_score(y_te, pred),3))
print("MAE:", round(mean_absolute_error(y_te, pred),1),
      " RMSE:", round(np.sqrt(mean_squared_error(y_te, pred)),1))
```

```
Multiple Linear Regression  R^2: 0.224
MAE: 214281.5  RMSE: 432944.1
```

11. Predict the value of the dependent variable from the different classifiers, such as Logistic Regression, KNN, Naïve-Bayes, and Decision Tree.

```
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Make price categories (Low / Mid / High)
q1, q2 = df["Price"].quantile([0.33, 0.66])
def to_cat(p):
    return "Low" if p<=q1 else ("Mid" if p<=q2 else "High")
df["PriceCat"] = df["Price"].apply(to_cat)

# Use numeric features for simplicity (like labs)
X = df[["Year","Mileage","Cylinders"]].dropna()
y = df.loc[X.index, "PriceCat"]

X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size=0.25, stratify=y, random_state=0)

sc = StandardScaler()
X_tr_s = sc.fit_transform(X_tr)
X_te_s  = sc.transform(X_te)

models = {
    "Logistic Regression": LogisticRegression(max_iter=400, random_state=0),
    "KNN": KNeighborsClassifier(n_neighbors=7),
    "Naive Bayes": GaussianNB(),
    "Decision Tree": DecisionTreeClassifier(criterion="entropy", max_depth=12, random_state=0)
}

preds = {}
for name, m in models.items():
    m.fit(X_tr_s, y_tr)
    yhat = m.predict(X_te_s)
    preds[name] = yhat
    print(f"\n=== {name} ===")
    print("Accuracy:", round(accuracy_score(y_te, yhat),3))
    print("Confusion Matrix:\n", confusion_matrix(y_te, yhat))
    print(classification_report(y_te, yhat))
```

```
=== Logistic Regression ===
Accuracy: 0.478
Confusion Matrix:
 [[271  89 101]
 [ 66 268 123]
 [133 203 116]]
              precision    recall  f1-score   support

        High       0.58      0.59      0.58       461
         Low       0.48      0.59      0.53       457
         Mid       0.34      0.26      0.29       452

    accuracy                           0.48      1370
   macro avg       0.47      0.48      0.47      1370
weighted avg       0.47      0.48      0.47      1370
```

```
=== KNN ===
Accuracy: 0.434
Confusion Matrix:
 [[250 111 100]
 [110 226 121]
 [157 177 118]]
              precision    recall  f1-score   support

        High       0.48      0.54      0.51       461
         Low       0.44      0.49      0.47       457
         Mid       0.35      0.26      0.30       452

    accuracy                           0.43      1370
   macro avg       0.42      0.43      0.43      1370
weighted avg       0.42      0.43      0.43      1370
```

```
=== Naive Bayes ===
Accuracy: 0.487
Confusion Matrix:
 [[271 125  65]
 [ 66 309  82]
 [133 232  87]]
              precision    recall  f1-score   support

        High       0.58      0.59      0.58       461
         Low       0.46      0.68      0.55       457
         Mid       0.37      0.19      0.25       452

    accuracy                           0.49      1370
   macro avg       0.47      0.49      0.46      1370
weighted avg       0.47      0.49      0.46      1370
```

```
===== Decision Tree =====
Accuracy: 0.439
Confusion Matrix:
 [[215 174  72]
 [ 70 316  71]
 [115 267  70]]
              precision    recall  f1-score   support

        High       0.54      0.47      0.50       461
         Low       0.42      0.69      0.52       457
         Mid       0.33      0.15      0.21       452

    accuracy                           0.44      1370
   macro avg       0.43      0.44      0.41      1370
weighted avg       0.43      0.44      0.41      1370


Accuracies: {'Logistic Regression': 0.478, 'KNN': 0.434, 'Naive Bayes': 0.487, 'Decision Tree': 0.439}
Best classifier: Naive Bayes with accuracy = 0.487
```

12. Evaluate the performance of each model using a confusion matrix and accuracy and identify the best-fit classifier for the chosen dataset.

```
acc = {name: accuracy_score(y_te, yhat) for name, yhat in preds.items()}
best_name = max(acc, key=acc.get)
print("\nAccuracies:", {k: round(v,3) for k,v in acc.items()})
print("Best classifier:", best_name, "with accuracy =", round(acc[best_name],3))
```

```
Model Accuracies: {'Logistic Regression': 0.478, 'KNN': 0.434, 'Naive Bayes': 0.487,
'Decision Tree': 0.439}
Best classifier: Naive Bayes with accuracy = 0.487
```

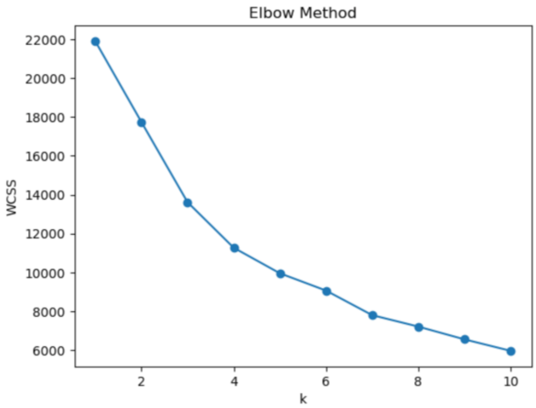
13. Predict the dependent variable by using the best-fit classifier.

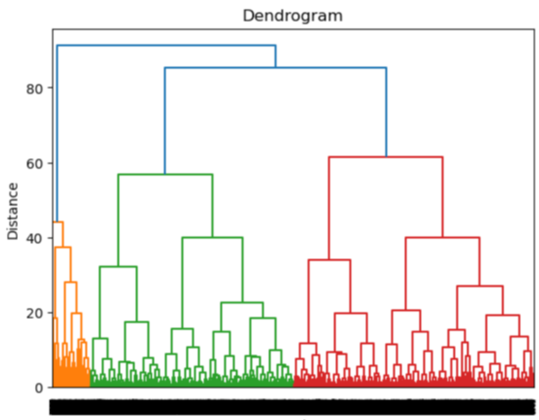
```
best_model = {
    "Logistic Regression": models["Logistic Regression"],
    "KNN": models["KNN"],
    "Naive Bayes": models["Naive Bayes"],
    "Decision Tree": models["Decision Tree"]
}[best_name]

print("\nBest model predictions for first 5 test rows:")
print(best_model.predict(X_te_s[:5]))
```

```
Predictions of best model for first 5 test rows:
['Low' 'Low' 'High' 'Mid' 'Low']
```

14. Perform the cluster analysis, such as K-means and Horizontal (Hierarchical/Agglomerative), for any field from the chosen dataset.

```
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.preprocessing import StandardScaler
import scipy.cluster.hierarchy as sch
import matplotlib.pyplot as plt
clust = df[["Year", "Mileage", "Cylinders", "Price"]].dropna().copy()
scaler = StandardScaler()
Z = scaler.fit_transform(clust)
wcss = []
for k in range(1, 11):
    km = KMeans(n_clusters=k, init="k-means++", random_state=0)
    km.fit(Z)
    wcss.append(km.inertia_)
plt.figure(figsize=(6,4))
plt.plot(range(1, 11), wcss, marker="o")
plt.title("Elbow Method")
plt.xlabel("k (number of clusters)")
plt.ylabel("WCSS")
plt.show()
kmeans = KMeans(n_clusters=3, init="k-means++", random_state=0)
km_labels = kmeans.fit_predict(Z)
plt.figure(figsize=(6,4))
plt.scatter(Z[km_labels == 0, 0], Z[km_labels == 0, 1], label="Cluster 1")
plt.scatter(Z[km_labels == 1, 0], Z[km_labels == 1, 1], label="Cluster 2")
plt.scatter(Z[km_labels == 2, 0], Z[km_labels == 2, 1], label="Cluster 3")
plt.scatter(
    kmeans.cluster_centers_[:, 0],
    kmeans.cluster_centers_[:, 1],
    s=200,
    label="Centroids"
)
plt.title("K-Means Clusters")
plt.xlabel("Feature 1")
plt.ylabel("Feature 2")
plt.legend()
plt.show()
plt.figure(figsize=(8,4))
sch.dendrogram(sch.linkage(Z, method="ward"))
plt.title("Dendrogram")
plt.xlabel("Samples")
plt.ylabel("Distance")
plt.show()
hc = AgglomerativeClustering(n_clusters=3, linkage="ward")
hc_labels = hc.fit_predict(Z)
print("First 10 hierarchical labels:", hc_labels[:10])
```





Samples

```
First 10 hierarchical labels: [0 2 2 0 0 2 2 0 0 0]
```

15. Explain the strategy for improving the system after viewing the cluster diagram.

After viewing the K-Means and Hierarchical cluster diagrams, the following strategies can help improve the system:

- Replace **Y**ear with a new field called Age = current year − Year to better show how age affects car price.

- Add an interaction feature, Mileage × Age, since the effect of mileage changes with car age.

- Apply Target Encoding for Make and Model instead of one-hot encoding to reduce the number of columns and improve training speed.

- Remove or limit outliers in price and mileage to make the clusters more balanced.

- Use both the Elbow Method and Silhouette Score to select the best number of clusters (k).

- Try PCA (Principal Component Analysis) before clustering to reduce dimensions and make the clusters visually clearer.

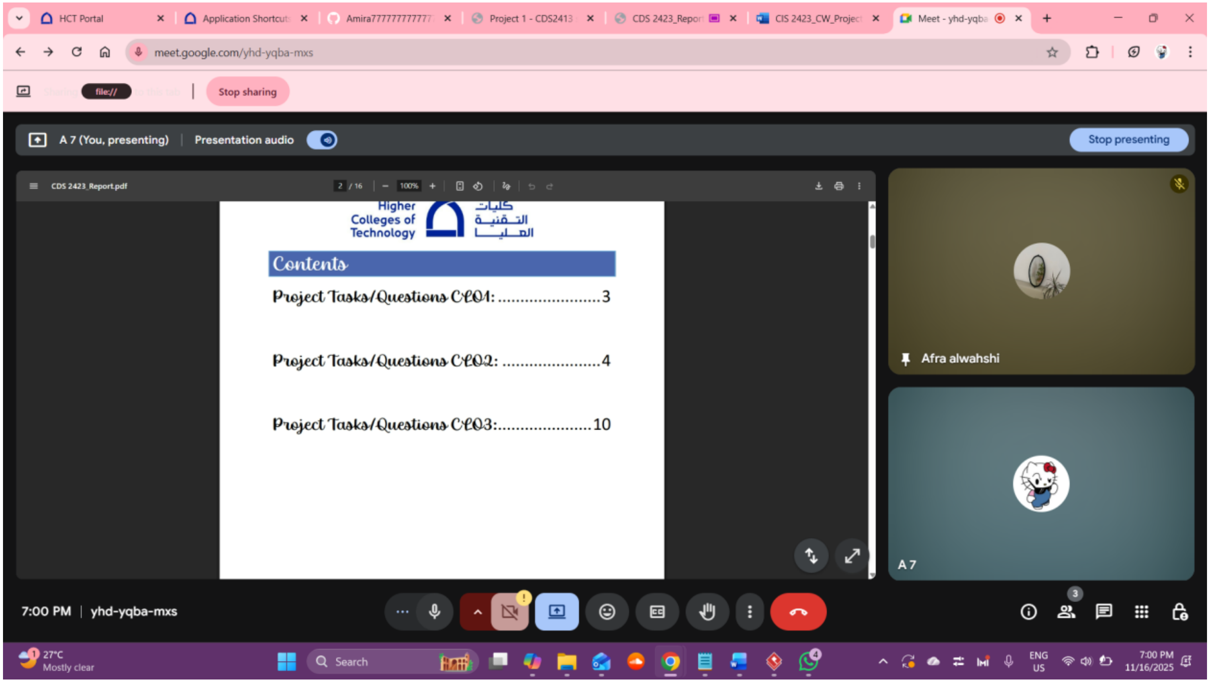
Bibliography:

Colab, G. (2025). *Google Colaboratory*. Retrieved from https://colab.google/

Google Colaboratory. (2025). Retrieved from python google colab: https://colab.google/

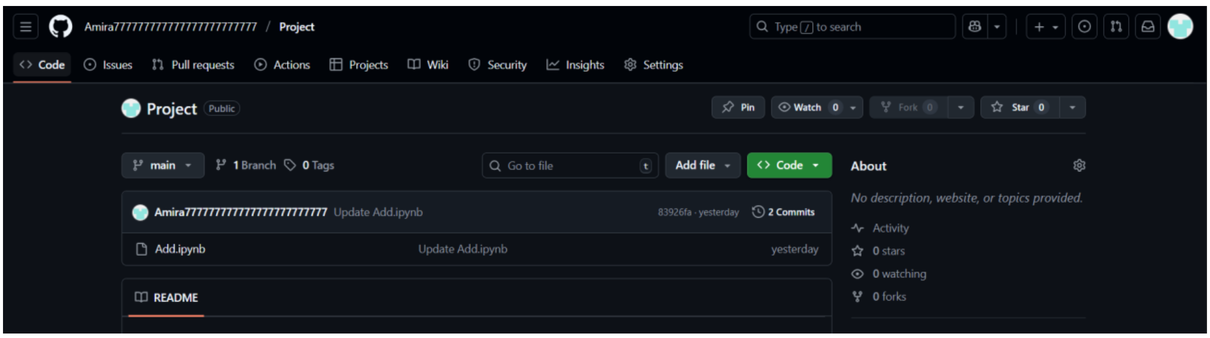
kaggle. (2025). Retrieved from UAE Used Car Prices & Features (10K+ Listings): https://www.kaggle.com/datasets/alikalwar/uae-used-car-prices-and-features-10k-listings

Collaboration:

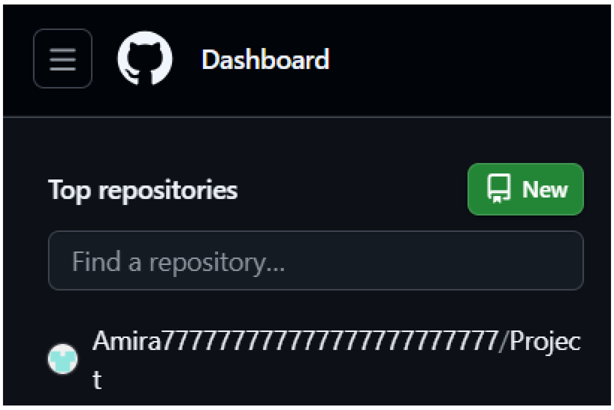


Git Hub:

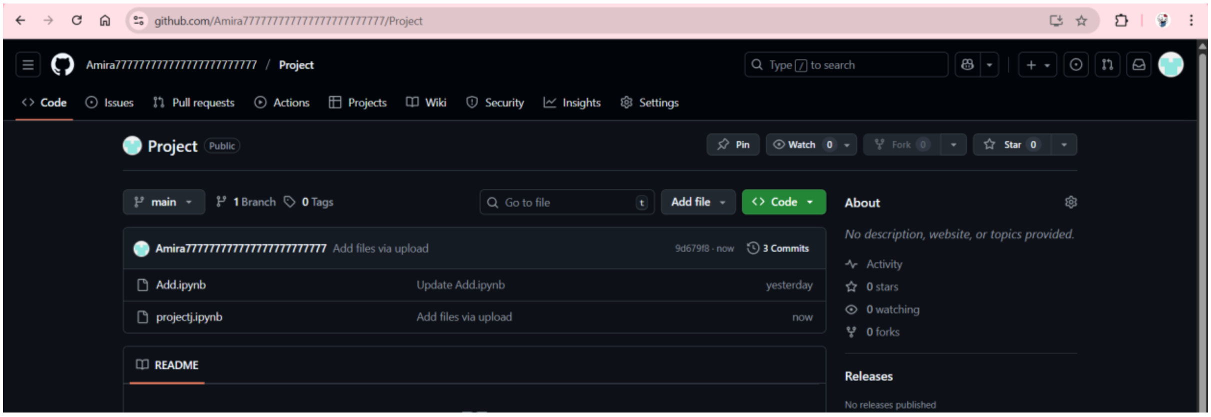
1- Create a Git Hub account



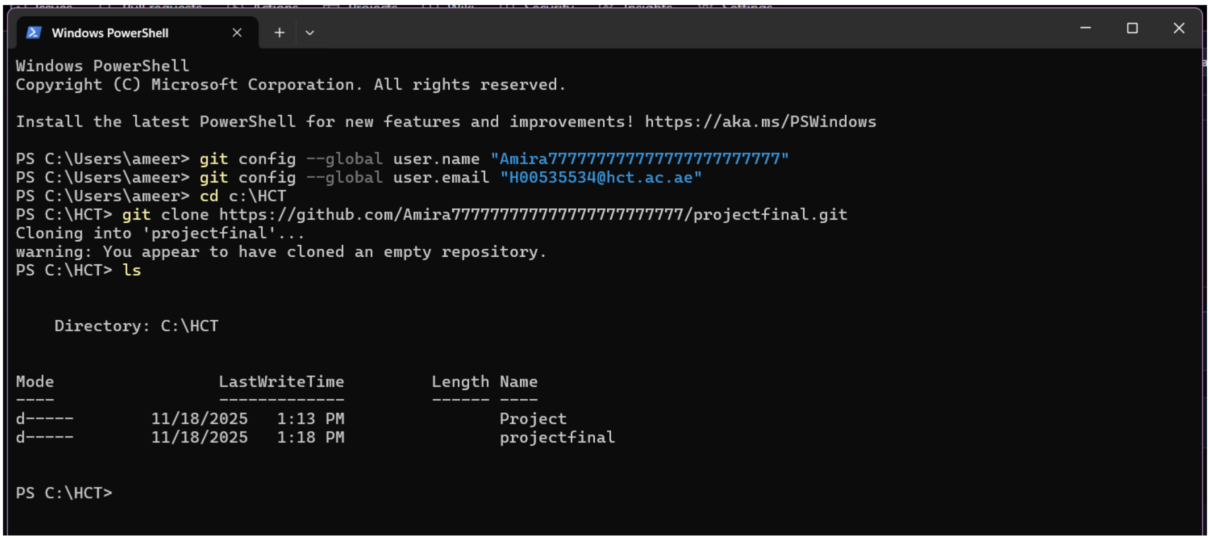
2- Create a new repository



3- Upload all the project files created for CLO1, CLO2, and CLO3 to the git hub repository.



4- Configure Git with Git Hub



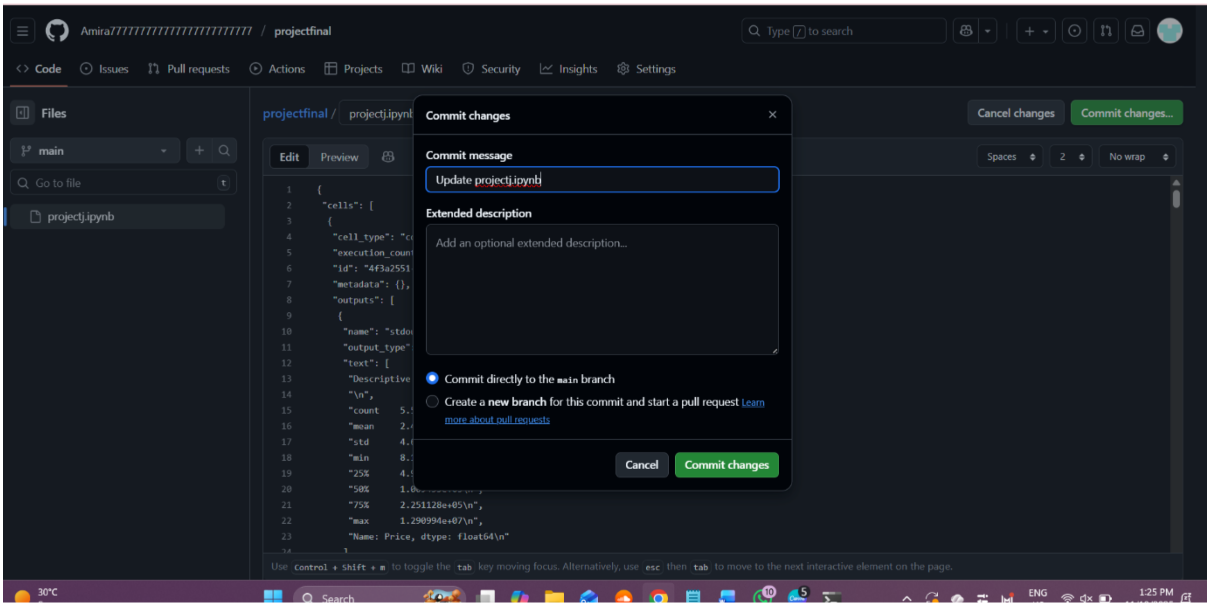
5- Clone git hub repository to git

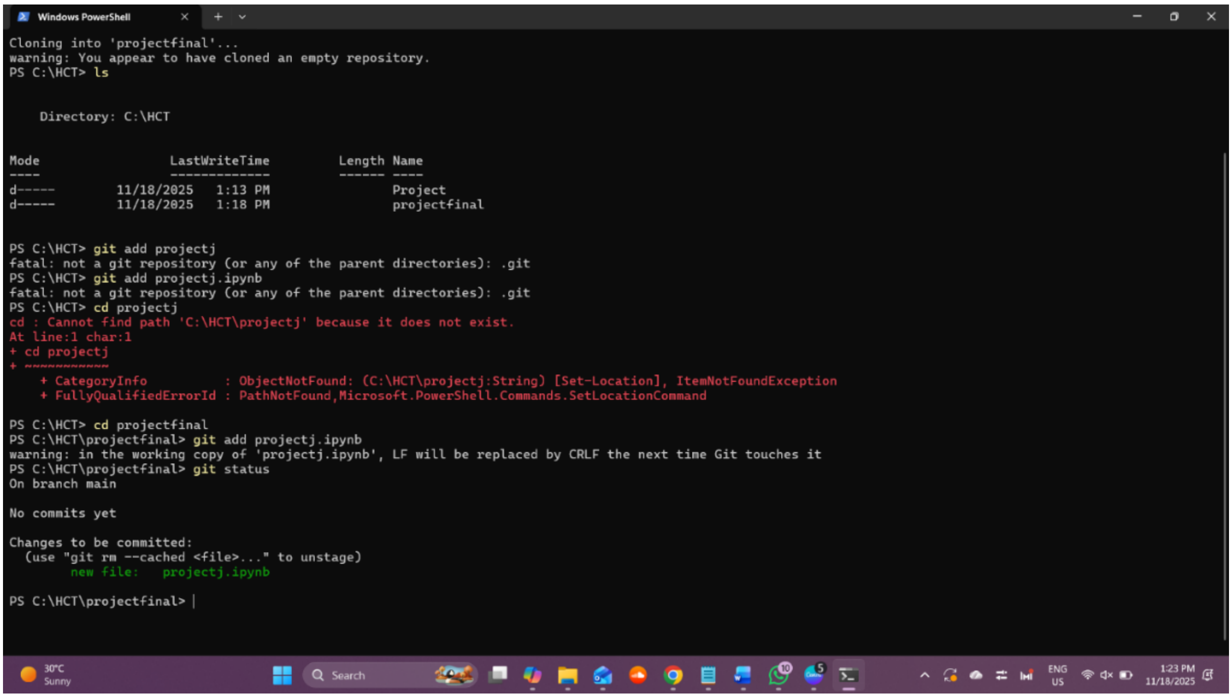
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ameer> git config --global user.name "Amira777777777777777777777777"
PS C:\Users\ameer> git config --global user.email "H00535534@hct.ac.ae"
PS C:\Users\ameer> cd c:\HCT
PS C:\HCT> git clone https://github.com/Amira777777777777777777777777/projectfinal.git
Cloning into 'projectfinal'...
warning: You appear to have cloned an empty repository.
PS C:\HCT> ls


    Directory: C:\HCT


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/18/2025   1:13 PM                Project
d-----        11/18/2025   1:18 PM                projectfinal


PS C:\HCT> git add projectj
fatal: not a git repository (or any of the parent directories): .git
PS C:\HCT> git add projectj.ipynb
fatal: not a git repository (or any of the parent directories): .git
PS C:\HCT> cd projectj
cd : Cannot find path 'C:\HCT\projectj' because it does not exist.
At line:1 char:1
+ cd projectj
+ ~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\HCT\projectj:String) [Set-Location], ItemNotFoundException
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS C:\HCT> cd projectfinal
PS C:\HCT\projectfinal> git add projectj.ipynb
warning: in the working copy of 'projectj.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\HCT\projectfinal> git status
On branch main
```

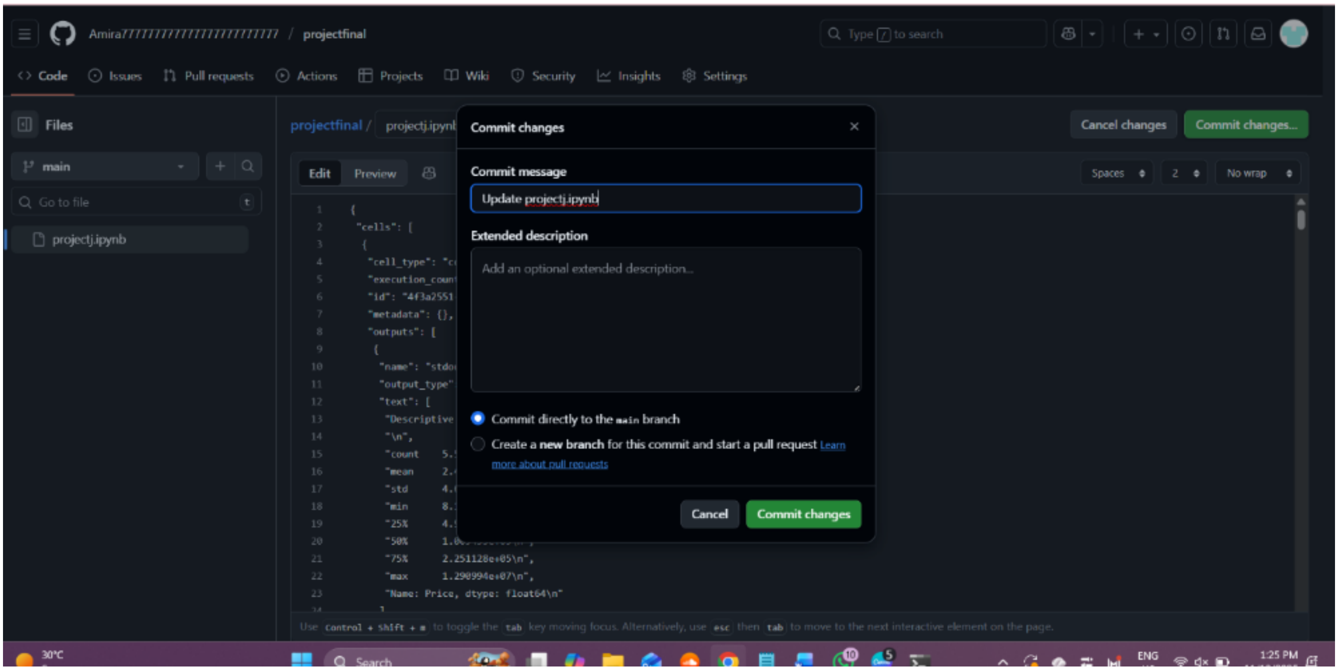
6- Pull any file from git hub repository to git

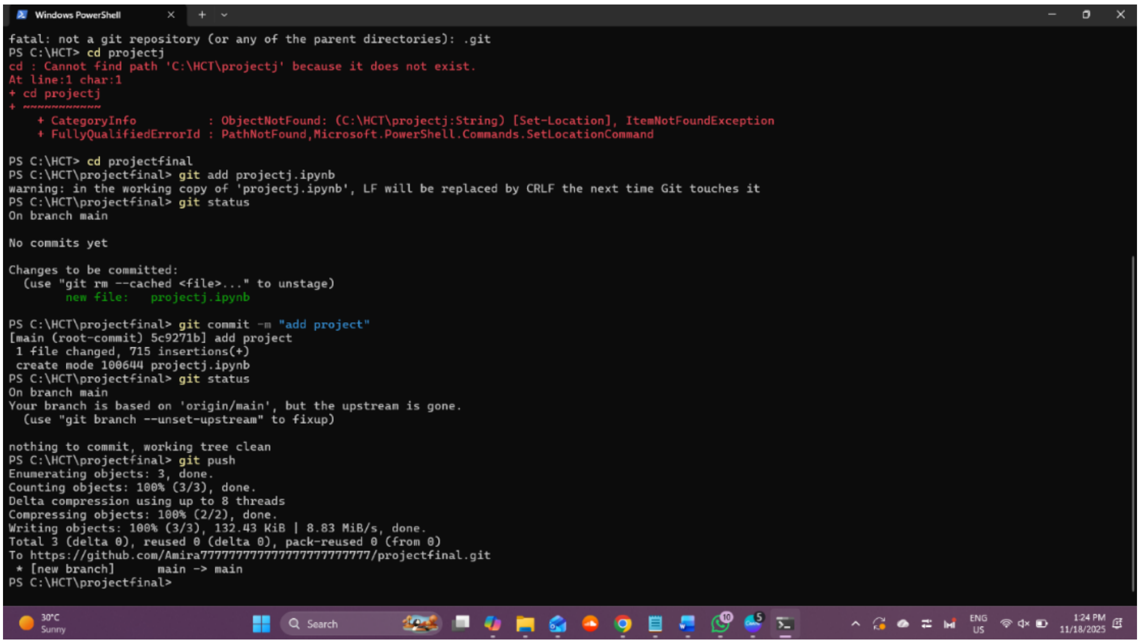
```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ameer> git config --global user.name "Amira777777777777777777777777"
PS C:\Users\ameer> git config --global user.email "H00535534@hct.ac.ae"
PS C:\Users\ameer> cd c:\HCT
PS C:\HCT> git clone https://github.com/Amira777777777777777777777777/projectfinal.git
Cloning into 'projectfinal'...
warning: You appear to have cloned an empty repository.
PS C:\HCT> ls


    Directory: C:\HCT


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----        11/18/2025   1:13 PM                Project
d-----        11/18/2025   1:18 PM                projectfinal


PS C:\HCT> git add projectj
fatal: not a git repository (or any of the parent directories): .git
PS C:\HCT> git add projectj.ipynb
fatal: not a git repository (or any of the parent directories): .git
PS C:\HCT> cd projectj
cd : Cannot find path 'C:\HCT\projectj' because it does not exist.
At line:1 char:1
+ cd projectj
+ ~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (C:\HCT\projectj:String) [Set-Location], ItemNotFoundException
    + FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand

PS C:\HCT> cd projectfinal
PS C:\HCT\projectfinal> git add projectj.ipynb
warning: in the working copy of 'projectj.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\HCT\projectfinal> git status
On branch main
```

7- Modify the pulled file and push the modify file to get hub.






projectfinal
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Files
main
Go to file
projectj.ipynb
Cancel changes
Commit changes...
Commit changes
Commit message
Update projectj.ipynb
Extended description
Add an optional extended description...
Commit directly to the main branch
Create a new branch for this commit and start a pull request Learn more about pull requests
Cancel
Commit changes


Windows PowerShell
fatal: not a git repository (or any of the parent directories): .git
PS C:\HCT> cd projectj
cd : Cannot find path 'C:\HCT\projectj' because it does not exist.
At line:1 char:1
+ cd projectj
+ ~~~~~~~~~~~
+ CategoryInfo : ObjectNotFound: (C:\HCT\projectj:String) [Set-Location], ItemNotFoundException
+ FullyQualifiedErrorId : PathNotFound,Microsoft.PowerShell.Commands.SetLocationCommand
PS C:\HCT> cd projectfinal
PS C:\HCT\projectfinal> git add projectj.ipynb
warning: in the working copy of 'projectj.ipynb', LF will be replaced by CRLF the next time Git touches it
PS C:\HCT\projectfinal> git status
On branch main
No commits yet
Changes to be committed:
(use "git rm --cached <file>..." to unstage)
new file: projectj.ipynb
PS C:\HCT\projectfinal> git commit -m "add project"
[main (root-commit) 5c9271b] add project
1 file changed, 715 insertions(+)
create mode 100644 projectj.ipynb
PS C:\HCT\projectfinal> git status
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
(use "git branch --unset-upstream" to fixup)
nothing to commit, working tree clean
PS C:\HCT\projectfinal> git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 132.43 KiB | 8.83 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Amira7777777777777777777777/projectfinal.git
* [new branch] main -> main
PS C:\HCT\projectfinal>

```
PS C:\HCT\projectfinal> git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   projectj.ipynb

PS C:\HCT\projectfinal> git commit -m "add project"
[main (root-commit) 5c9271b] add project
 1 file changed, 715 insertions(+)
 create mode 100644 projectj.ipynb
PS C:\HCT\projectfinal> git status
On branch main
Your branch is based on 'origin/main', but the upstream is gone.
  (use "git branch --unset-upstream" to fixup)

nothing to commit, working tree clean
PS C:\HCT\projectfinal> git push
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 132.43 KiB | 8.83 MiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0 (from 0)
To https://github.com/Amira777777777777777777777777/projectfinal.git
 * [new branch]      main -> main
PS C:\HCT\projectfinal> git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (2/2), done.
remote: Total 3 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
Unpacking objects: 100% (3/3), 956 bytes | 34.00 KiB/s, done.
From https://github.com/Amira777777777777777777777777/projectfinal
   5c9271b..f5699b5  main       -> origin/main
Updating 5c9271b..f5699b5
Fast-forward
 projectj.ipynb | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
PS C:\HCT\projectfinal>
```