System Requirements Specification Index

For

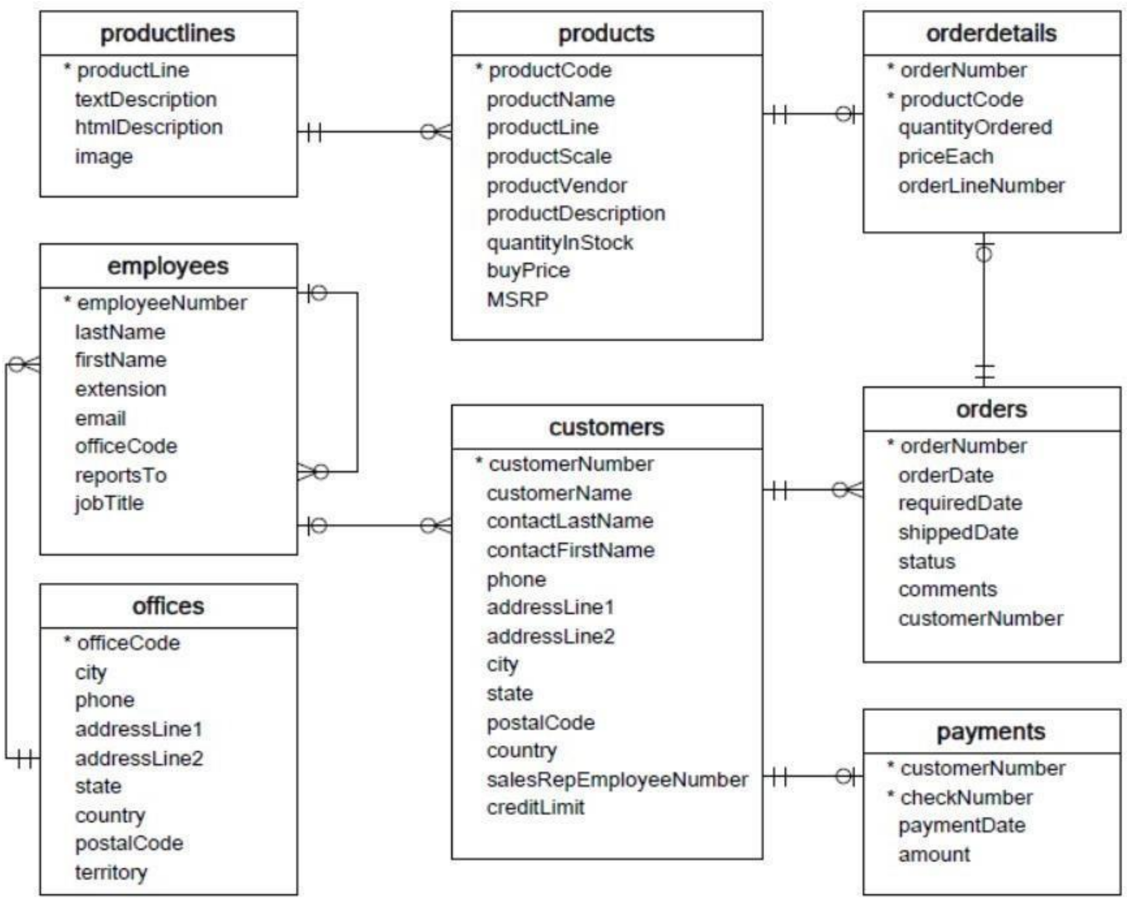
Data Engineering with Pyspark for L1 level

Version 1.0

IIHT Pvt. Ltd.
fullstack@iiht.com

Problem Statement : Perform basic load, transformation and data analytics using Pyspark.

You are given a database that you need to extract from multiple sources and perform some transformations and analysis. The database has the following the schema.



Customers: stores customer's data.
Products: stores a list of scale model cars.
ProductLines: stores a list of product line categories.
Orders: stores sales orders placed by customers.
OrderDetails: stores sales order line items for each sales order.
Payments: stores payments made by customers based on their accounts.
Employees: stores all employee information as well as the organisation structure such as who reports to whom.
Offices: stores sales office data.

The assessment contains the following folder structure.

```
DE_with_pyspark |
                |--src
                   |--__init__.py
                   |-- DE_with_pyspark.ipynb – your code goes here
                   |--constants.py – defines few constants
                   |-- requirements.txt
                |--data
                   |--data.csv – data files for the problem
                |--docs – contains documents
                   |--DE with Pyspark Document.docx – this document
```

DE_with_pyspark.ipynb

The **DE_with_pyspark.ipynb** has the following methods that you need to implement.

Important instructions:

- Please DO NOT change any method signature/filename, as it would result in failed submission.
- Please return the dataframe columns as specified in the comments with same names, order doesn't matter.
- For questions involving transformations just return the transformed data. Kindly not overwrite original data files.
- Your system comes with pyspark installed.
- Do not initiate multiple spark sessions. If terminal doesn't exit press CTRL+C

You will also find below signatures in the code notebook. Scroll down for execution instructions.

Load data:

Setting up the environment:

Login to the MySQL shell of Workbench and run below commands to create a mysql table. (Credentials for the same are available README.txt file on Desktop)

```
1. create database classicmodels;

2. use classicmodels;

3. create table orderdetails(orderNumber int,
productCode varchar(100),
quantityOrdered int,
priceEach decimal(10,2),
orderLineNumber int
);

# Before running next command, you are required to copy the file
'orderdetails.csv' present in data folder of project to the following
location in VM:
'C:/ProgramData/MySQL/MySQL Server8.0/Uploads/'
Once done, run the next set of commands.

4. LOAD DATA INFILE 'C:/ProgramData/MySQL/MySQL Server 8.0/Uploads/orderdetails.csv'
INTO TABLE orderdetails
FIELDS TERMINATED BY ','
ENCLOSED BY '"'
LINES TERMINATED BY '\n'
IGNORE 1 LINES

5. select * from orderdetails; -test
```

NOTE: mysql-connector-jar file is available at: "D:/MySQL folder". Pls make sure you copy the same in C drive root.

```
Q1) def load_data_from_mysql(spark :pyspark.sql.SparkSession, db_name :str,
    table_name :str) -> pyspark.sql.DataFrame:
    '''
    - load data from MySQL table 'table_name' from database 'db_name'
      and return the table as a spark dataframe


     For the MySQL connection use below information:
     jdbc driver: 'com.mysql.cj.jdbc.Driver'
     Hostname: 'localhost'
     Port: 3306
     Database: 'classicmodels'
     Table_name: 'order_details'
     Username: 'mysql_user'
     Password: 'user'
    '''
```

Q2) def load_data_from_csv(spark :pyspark.sql.SparkSession, csv_file_name: str) -> pyspark.sql.DataFrame:
```
    '''
    Load data from CSV file 'csv_path' and return a spark Dataframe
    PS: The data files for this assignment are in 'data' folder
    You can access full path of 'data/' folder using 'DATA_FOLDER' variable
    from constants.py.
    '''
```

Q3) def load_data_from_flatfile(spark :pyspark.sql.SparkSession, txt_file_name: str) -> pyspark.sql.DataFrame:
```
    '''
    Load data from flat file 'txt_file_name' separated with ':' and return a
    spark Dataframe.
    PS: The data files for this assignment are in 'data' folder
    You can access full path of 'data/' folder using 'DATA_FOLDER' variable
    from constants.py
    '''
```

Transformations:

Q4) def clean_product_MSRP_column(spark :pyspark.sql.SparkSession) -> pyspark.sql.DataFrame:
```
    '''
    Due to a data entry issues MSRP, the selling price is lower than its
    buyPrice for some products. Change MSRP to 1.4 times of the buyPrice for
    such products and cast it to two decimal places.
    Note: Please do not change original file/dataframe.

    Return a spark dataframe with following columns.
    |productCode|productName|productLine|productScale|productVendor|productDescription|quantityInStock|buyPrice|MSRP|
    '''
```

Q5) def get_customer_info(spark :pyspark.sql.SparkSession) -> pyspark.sql.DataFrame:
```
    '''
    Return a consolidated customer info using structs.

    Return a spark dataframe with following columns.
    |custID|custName|country|#orders|totalMoneySpent|creditLimit|
    '''
```

Q6) def return_top_5_big_spend_countries(spark :pyspark.sql.SparkSession) ->pyspark.sql.DataFrame:
```
    '''
    Return top 5 big countries which had spent the most $(highest order
    value).

    Return a spark dataframe with following columns.
    |country|totalOrderValue|
```

Code structure to run successful testcase

Function Definition

```
def load_data_from_flatfile(spark: pyspark.sql.SparkSession, txt_file_name: str) ->
pyspark.sql.DataFrame:

    # Docstring describing the function
    '''
    Load data from flat file 'txt_file_name' separated with ':' and return a spark
Dataframe
    PS: The data files for this assignment are in 'data' folder
    You can access full path of 'data/' folder using 'DATA_FOLDER' variable
    from constants.py
    '''

    # Function Body
    return spark.read.csv(
        # Path to the file (concatenating the DATA_FOLDER path with the file name)
        constants.DATA_FOLDER + txt_file_name,

        # File format details
        sep=':',                # Delimiter used in the flat file
        header=True,            # Indicates that the first row of the file is a header
        inferSchema=True        # Infers the schema of the data based on the content
    )
```

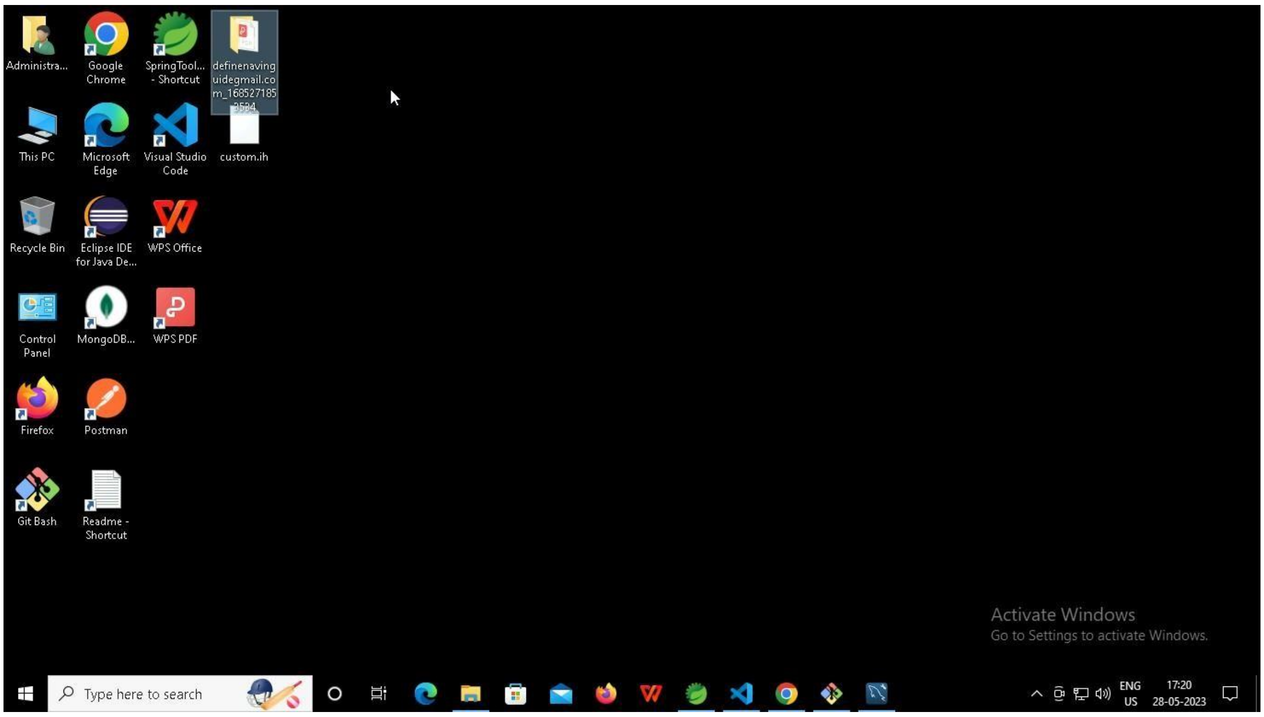
Execution Steps to Follow:

1. All actions like build, compile, running application, running test cases will be through Command Terminal.
2. These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
3. Add your code in src/DE_with_pyspark.ipynb:
4. **It is mandatory to run test cases on your project for scores to be evaluated before submission.**

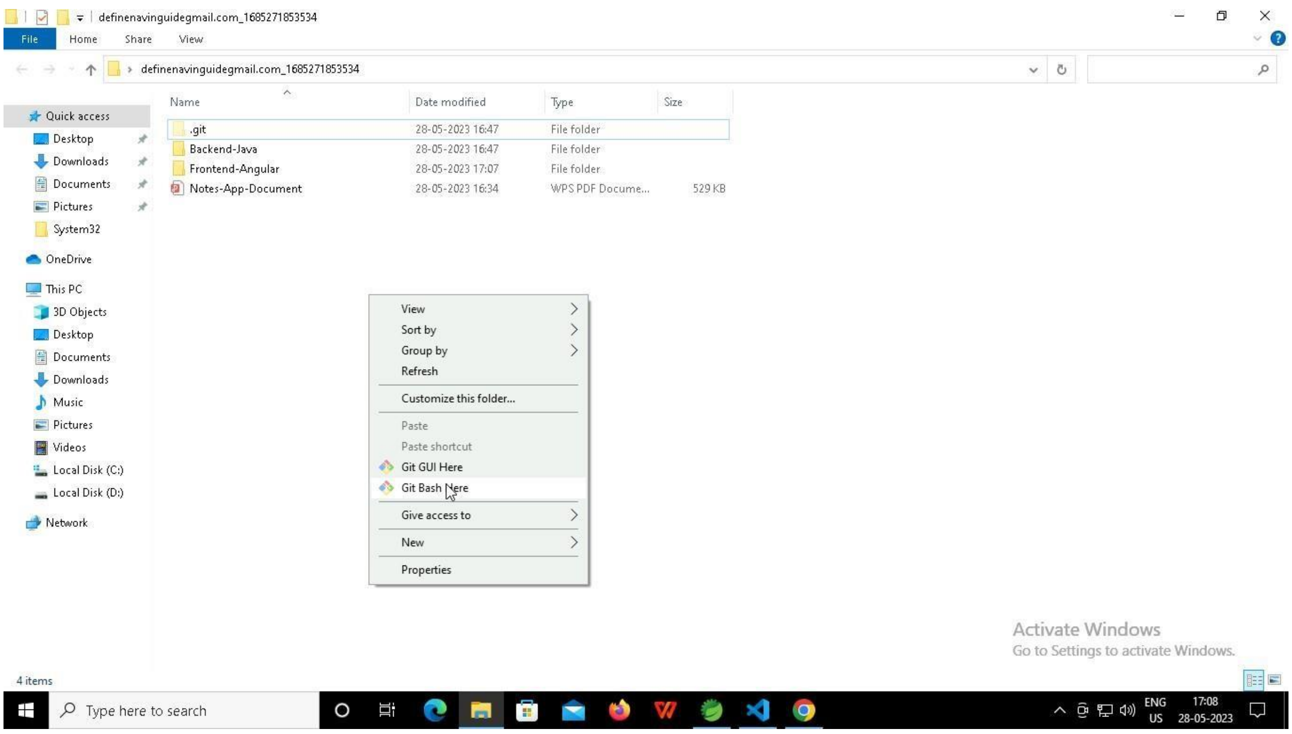
   To run Test cases, use following command
   python -m pytest tests

   You can run test cases as many numbers of times and at any stage of development, to check how many test cases are passed/failed and accordingly refactor your code.

5. **Make sure before final submission you commit all changes to git**. For that open the project folder available on desktop



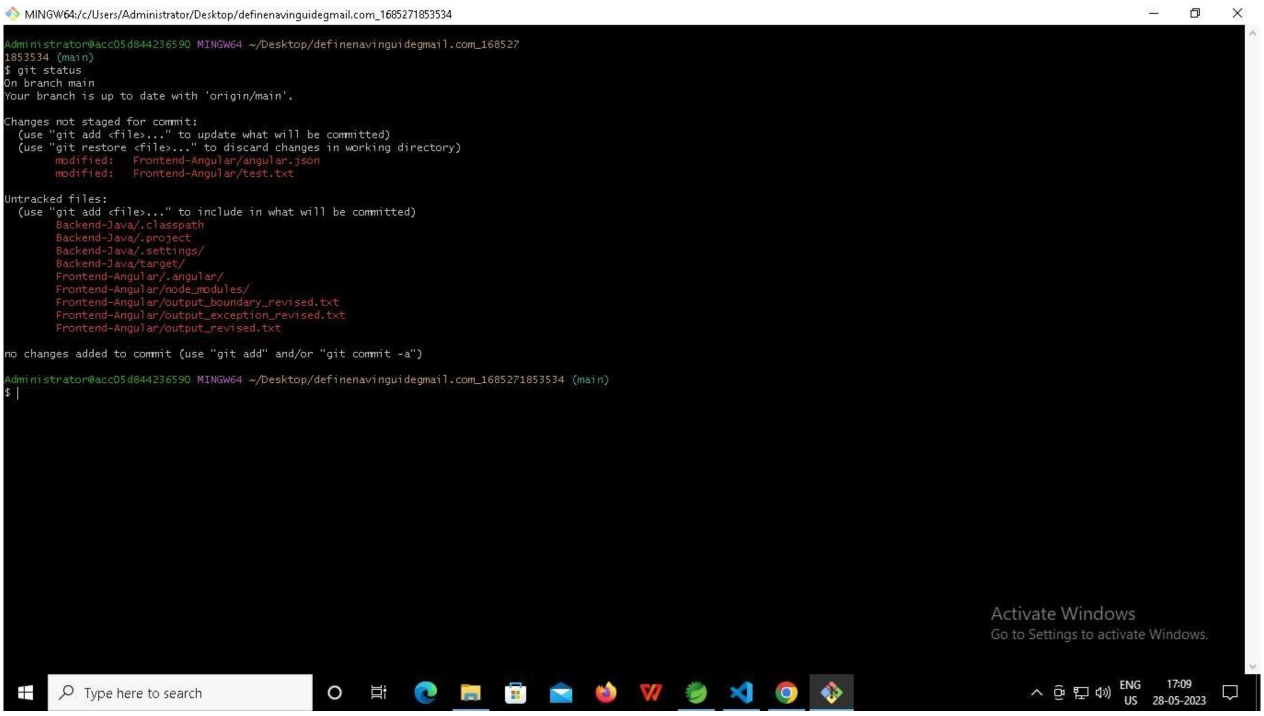
a. Right click in folder and open Git Bash



Name	Date modified	Type	Size
.git	28-05-2023 16:47	File folder	
Backend-Java	28-05-2023 16:47	File folder	
Frontend-Angular	28-05-2023 17:07	File folder	
Notes-App-Document	28-05-2023 16:34	WPS PDF Docume...	529 KB

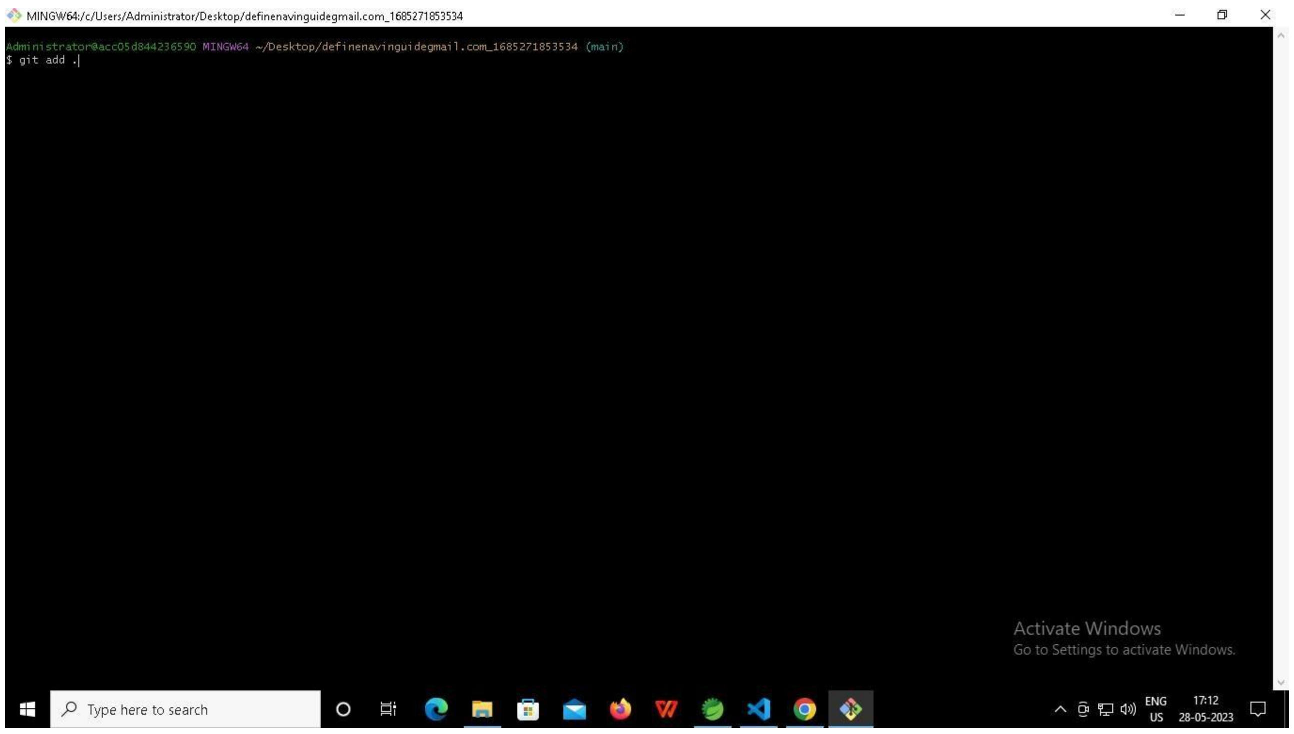
b. In Git bash terminal, run following commands

c. git status


MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534
Type here to search

```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_168527
1853534 (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   Frontend-Angular/angular.json
        modified:   Frontend-Angular/test.txt

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Backend-Java/.classpath
        Backend-Java/.project
        Backend-Java/.settings/
        Backend-Java/target/
        Frontend-Angular/.angular/
        Frontend-Angular/node_modules/
        Frontend-Angular/output_boundary_revised.txt
        Frontend-Angular/output_exception_revised.txt
        Frontend-Angular/output_revised.txt

no changes added to commit (use "git add" and/or "git commit -a")

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ |
```

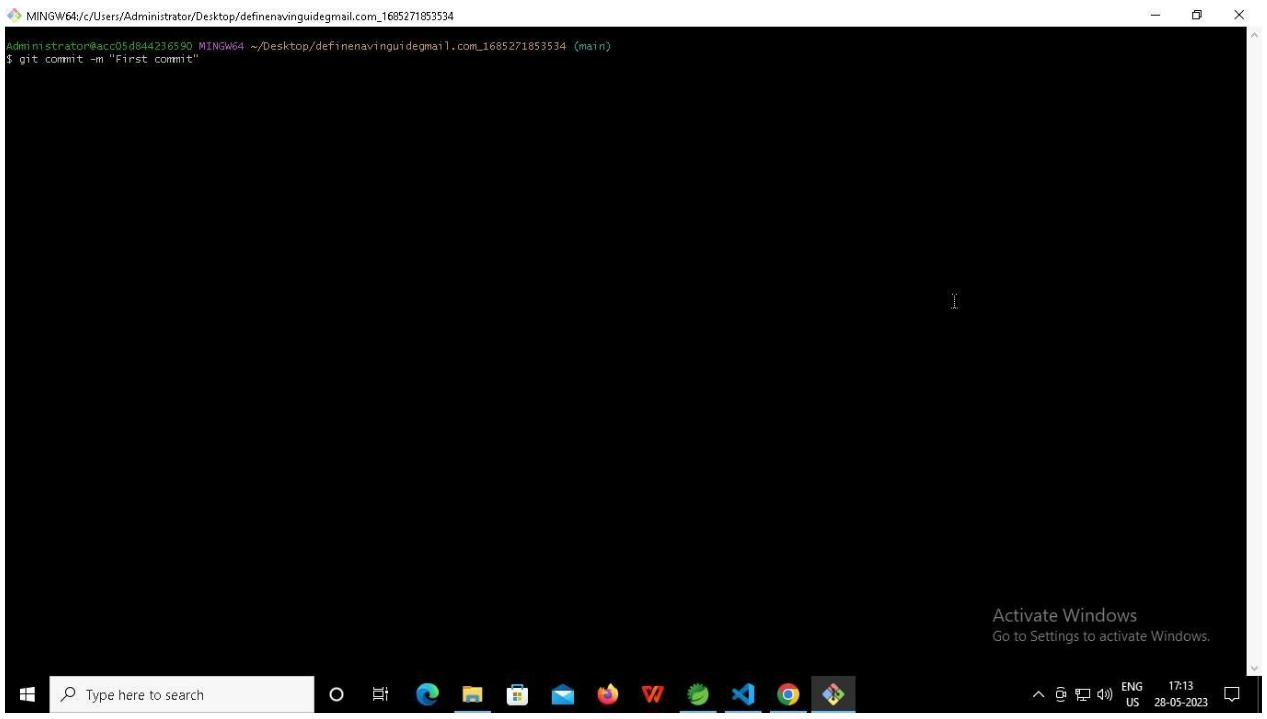
d. git add .



MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534

```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git add .
```

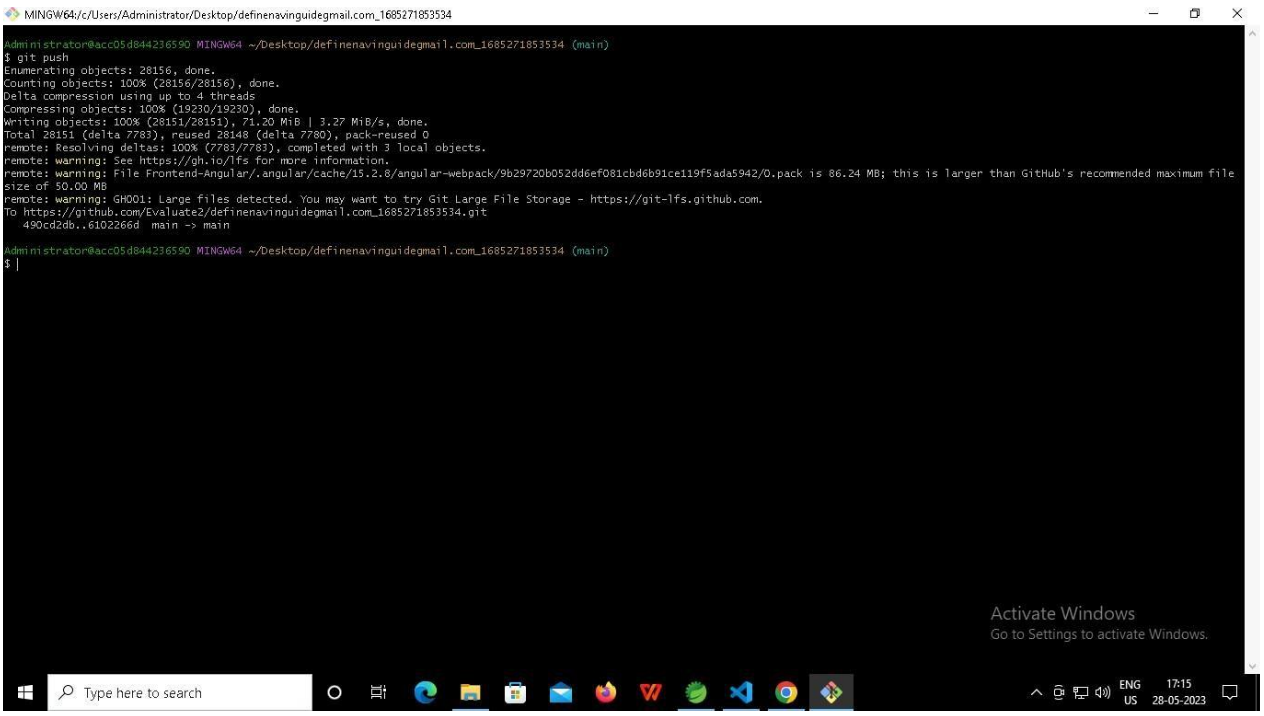
e. git commit -m “First commit”
(You can provide any message every time you commit)



MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534

```
Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git commit -m "First commit"
```

f. git push



```
MINGW64:/c/Users/Administrator/Desktop/definenavinguidegmail.com_1685271853534

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ git push
Enumerating objects: 28156, done.
Counting objects: 100% (28156/28156), done.
Delta compression using up to 4 threads
Compressing objects: 100% (19230/19230), done.
Writing objects: 100% (28151/28151), 71.20 MiB | 3.27 MiB/s, done.
Total 28151 (delta 7783), reused 28148 (delta 7780), pack-reused 0
remote: Resolving deltas: 100% (7783/7783), completed with 3 local objects.
remote: warning: See https://gh.io/lfs for more information.
remote: warning: File Frontend-Angular/.angular/cache/15.2.8/angular-webpack/9b29720b052dd6ef081cbd6b91ce119f5ada5942/0.pack is 86.24 MB; this is larger than GitHub's recommended maximum file
size of 50.00 MB
remote: warning: GH001: Large files detected. You may want to try Git Large File Storage - https://git-lfs.github.com.
To https://github.com/Evaluate2/definenavinguidegmail.com_1685271853534.git
   490cd2db..6102266d  main -> main

Administrator@acc05d844236590 MINGW64 ~/Desktop/definenavinguidegmail.com_1685271853534 (main)
$ |
```

-----x-----