



UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Malaysia-Japan
International
Institute of Technology
(MJIT)

MALAYSIA-JAPAN INTERNATIONAL INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRONIC SYSTEM ENGINEERING

SMJE 4383 ADVANCED PROGRAMMING
SESSION 2022/2023-1

ASSIGNMENT 1

	NAME	MATRICS NO
GROUP MEMEBRS	MOHAMAD FARIZUAN AKMAL BIN JAMALUDIN	A19MJ0049
	MUHAMMAD DANIAL FIKRI BIN KAMARUZAMAN	A19MJ0066
LECTURE	DR ZOOL HILMI BIN ISMAIL	

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION	3
1.1 BACKGROUND STUDY	3
1.2 PROJECT FRAMEWORK.....	4
1.3 OBJECTIVE	5
CHAPTER 2: METHODOLOGY	5
2.1 COPY DATA FROM CSV FILE	6
2.2 EMAIL NOTIFICATION.....	7
CHAPTER 3: RESULT AND DISCUSSION	9
CHAPTER 4: CONCLUSION.....	11

CHAPTER 1: INTRODUCTION

1.1 BACKGROUND STUDY

The process of copying data from a CSV file to a database is an essential task in data management. This task is often performed in order to centralize data and make it more accessible for analysis and reporting purposes. In order to perform this task, it is necessary to have a system in place for reading data from the CSV file and writing it to the database. Scripting languages such as Python provide libraries specifically designed for this purpose. These libraries offer a range of functions that allow for the reading and writing of data to and from CSV files and databases. This makes the process of copying data much more efficient and streamlined, as the libraries automate many of the manual steps involved in the process.

In order to make this process even more efficient, it is possible to incorporate email notifications. Once the data has been successfully copied from the CSV file to the database, an email can be sent to notify the relevant parties that the task has been completed. This saves time, as manual checks to confirm the completion of the task are no longer necessary. It also ensures that the relevant parties are aware that the data has been updated, allowing for more timely analysis and reporting. The implementation of email notifications requires the use of a library designed for sending emails. In Python, the "smtplib" library can be used to send emails through the Simple Mail Transfer Protocol (SMTP). This library offers customization options for both the sender and recipient email addresses, as well as the subject and body of the email.

Before sending the email notification, it is important to ensure that the data has been successfully copied to the database. This can be done by comparing the number of records in the CSV file to the number of records in the database after the data has been transferred. If the numbers match, it can be concluded that the data has been successfully copied and the email notification can then be sent. The process of copying data from a CSV file to a database is an important task in data management that can be made more efficient through the use of scripting languages and email notifications. The libraries provided by scripting languages such as Python automate many of the manual steps involved in the process, while email notifications provide a convenient way to confirm the completion of the task and keep relevant parties informed.

1.2 PROJECT FRAMEWORK

The procedure of transferring data from a CSV file to a database with email notification in Ubuntu using Python can be executed efficiently by adhering to a well-defined project framework. This framework can be divided into several crucial steps that must be completed in a specific order to guarantee the successful outcome of the project.

The initial step involves installing and configuring Python on the Ubuntu system. This requires downloading and installing the latest version of Python, along with setting up the necessary environment variables and dependencies. After the setup, the necessary libraries for reading and writing CSV files and connecting to databases must be installed. For instance, the "pandas" library is suitable for reading and writing to CSV files, while the "sqlalchemy" library is ideal for connecting to databases. The data from the CSV file is then read and stored in a data structure and written to the relevant tables and columns in the database. This may require creating new tables and columns if they do not exist.

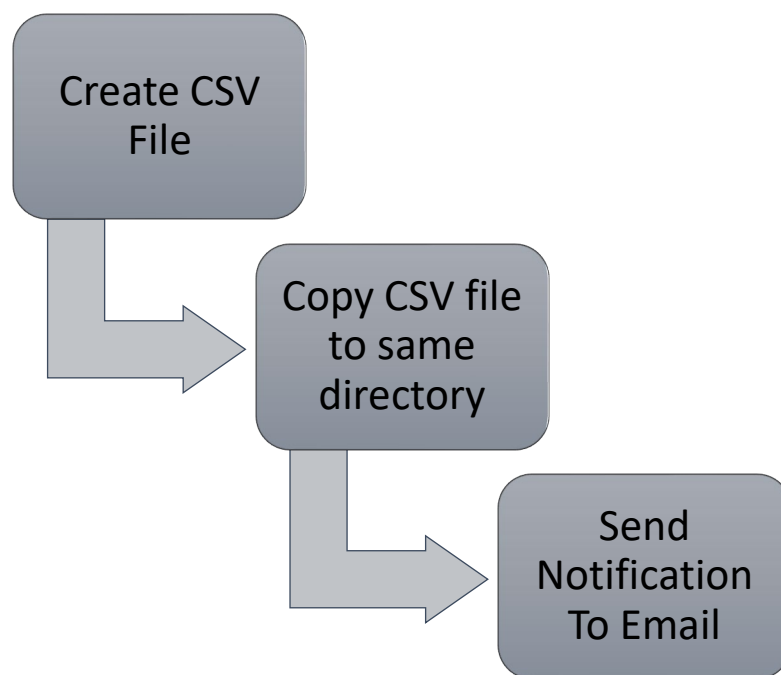
Prior to sending the email notification, it is crucial to verify that the data has been copied successfully to the database by comparing the number of records in the CSV file to the number of records in the database after the transfer. If they match, the data transfer has been successful, and an email notification can be sent via the "smtplib" library in Python. Lastly, the system must be monitored and kept up-to-date to guarantee the smooth operation of the data transfer process and prompt resolution of any issues that may arise. This may involve updating libraries and dependencies as necessary and fixing any bugs.

In summary, following a structured project framework allows for the successful execution of the process of transferring data from a CSV file to a database with email notification in Ubuntu using Python. This framework provides a clear and organized approach to the project, enabling the data transfer process to be automated and streamlined.

1.3 OBJECTIVE

1. To transfer the data from a CSV file to a new CSV file.
2. To send an email notification to the user once the transfer process is complete.

CHAPTER 2: METHODOLOGY



2.1 COPY DATA FROM CSV FILE

Here is a code in Python that demonstrates how to copy data from a CSV file and paste it in the same directory:

```
input_file = open ('grades.csv','r')           #open source file
output_file = open ('grades(copy).csv','w')     #create/overwrite output file
csv1 = input_file.read()                       #read content of source file

output_file.write(csv1)                        #source file content copy to output file

input_file.close()
output_file.close()

#sending email after copy is completed
from mail_noti import Email
print("Copy Completed! Email notification was sent!")
```

The task of copying data from a CSV file to a new CSV file can be accomplished using the `csv` module in Python. This module provides the tools needed to read from and write to CSV files, making it an ideal choice for this task. We using the CSV files that consist data of student's grade.

	A	B	C	D	E	F	G	H	I	
1	Last name	First name	SSN	Test1	Test2	Test3	Test4	Final	Grade	
2	Alfalfa	Aloysius	123-45-6789	40	90	100	83	49	D-	
3	Alfred	University	123-12-1234	41	97	96	97	48	D+	
4	Gerty	Gramma	567-89-0123	41	80	60	40	44	C	
5	Android	Electric	087-65-4321	42	23	36	45	47	B-	
6	Bumpkin	Fred	456-78-9012	43	78	88	77	45	A-	
7	Rubble	Betty	234-56-7890	44	90	80	90	46	C-	
8	Noshow	Cecil	345-67-8901	45	11	-1	4	43	F	
9	Buff	Bjt	632-79-9939	46	20	30	40	50	B+	
10	Airpump	Andrew	223-45-6789	49	1	90	100	83	A	
11	Backus	Jim	143-12-1234	48	1	97	96	97	A+	
12	Carnivore	Art	565-89-0123	44	1	80	60	40	D+	
13	Dandy	Jim	087-75-4321	47	1	23	36	45	C+	
14	Elephant	Ima	456-71-9012	45	1	78	88	77	B-	
15	Franklin	Benny	234-56-2890	50	1	90	80	90	B-	
16	George	Boy	345-67-3901	40	1	11	-1	4	B	
17	Heffalump	Harvey	632-79-9439	30	1	20	30	40	C	

Figure 2.1: CSV files students grade data.

The process of copying the data from the source file to the target file can be broken down into several steps.

First, the `csv` module must be imported in the Python script. This will give access to the functions and classes needed to work with CSV files. Next, the source CSV file must be opened in read mode and a reader object must be created to read the contents of the file. Similarly, the target CSV file must be opened in write mode and a writer object must be created to write to the target file. Once the source and target files have been opened and the reader and writer objects have been created, the data from the source file can be copied to the target file. Finally, once all the data has been copied, both the source and target files must be closed. This is an important step to ensure that all the data has been written to the target file and to free up system resources used by the files.

2.2 EMAIL NOTIFICATION

Here is a code in Python that demonstrates sending notifications to the email after copying the file:

```
import smtplib

class Email:

    subject = 'CSV copy by Python'
    body = 'Copy completed'
    sender = "
    receiver = "
    password = "

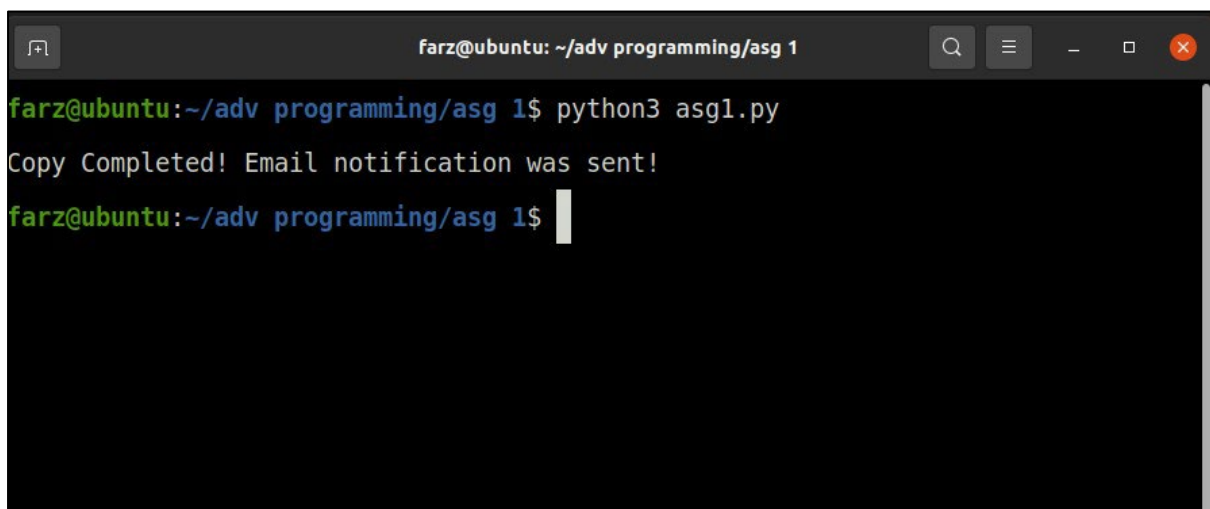
    message = 'subject: ' + subject + '\n\n' + body
    server = smtplib.SMTP('smtp.gmail.com', 587)
    server.starttls()
    server.login(sender, password)
    server.sendmail(sender, receiver, message)
```

Once the data from a CSV file has been copied to a new CSV file, it may be useful to notify the user that the process has been completed. One way to do this is by sending an email notification. In order to send an email notification, the '**smtplib**' library in Python can be used. This library provides the tools necessary to send emails through the Simple Mail Transfer Protocol (SMTP). The process of sending an email notification can be broken down into several steps. First, the '**smtplib**' library must be imported in the Python script. Next, an SMTP connection must be established with a mail server. This can be done using the '**SMTP**' class from the '**smtplib**' library.

Once the connection has been established, the email must be composed and the necessary information such as the recipient's email address, the sender's email address, the subject, and the body of the email must be specified. This information can be stored in variables and used to create the email message. Finally, the email must be sent using the '**sendmail**' method of the SMTP connection. This method takes the recipient's email address, the sender's email address, and the email message as arguments.

CHAPTER 3: RESULT AND DISCUSSION

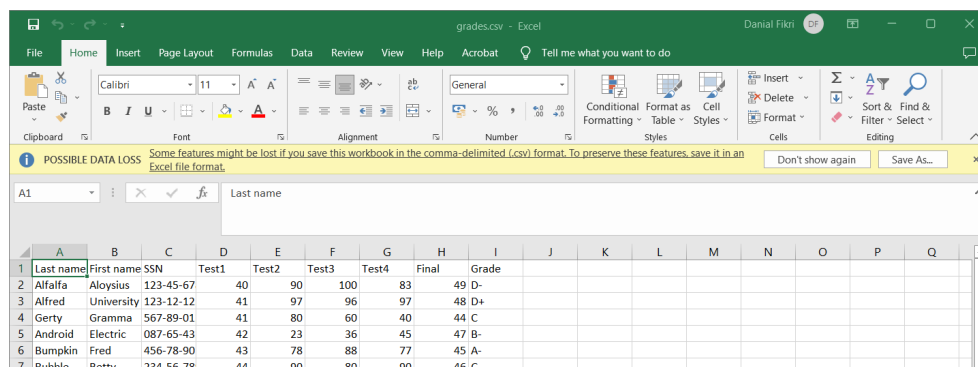
A “Copy Completes! Email notification was sent” message will appear on the terminal once the email notification process has been completed after the file has been copied. This indicates that the CSV file data has been copied and the recipient has been notified. In Figure 3.1, you can see the pop-up text displayed on the terminal after the process of copying file is complete.



```
farz@ubuntu: ~/adv programming/asn 1
farz@ubuntu:~/adv programming/asn 1$ python3 asg1.py
Copy Completed! Email notification was sent!
farz@ubuntu:~/adv programming/asn 1$
```

Figure 3.1: Pop-up message after email notification was sent

Copying the data from the original CSV file to the new CSV file has been successful done when the pop-up message is appeared. In Figures 3.2 and 3.3, we show the original CSV file (grades.csv) and its copy (grades(copy).csv).



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Last name	First name	SSN	Test1	Test2	Test3	Test4	Final	Grade								
2	Alfalfa	Aloysius	123-45-67	40	90	100	83	49	D-								
3	Alfred	University	123-12-12	41	97	96	97	48	D+								
4	Gerty	Gamma	567-89-01	41	80	60	40	44	C								
5	Android	Electric	087-65-43	42	23	36	45	47	B-								
6	Bumpkin	Fred	456-78-90	43	78	88	77	45	A-								
7	Rubble	Betty	234-56-78	44	90	80	90	46	C-								

Figures 3.2: Original student grades CSV file labelled as grade.csv

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
1	Last name	First name	SSN	Test1	Test2	Test3	Test4	Final	Grade								
2	Alfalfa	Aloysius	123-45-67	40	90	100	83	49	D-								
3	Alfred	University	123-12-12	41	97	96	97	48	D+								
4	Gerty	Gramma	567-89-01	41	80	60	40	44	C								
5	Android	Electric	087-65-43	42	23	36	45	47	B-								
6	Bumpkin	Fred	456-78-90	43	78	88	77	45	A-								
7	Rubble	Betty	234-56-78	44	90	80	90	46	C-								

Figures 3.3: copied student grades CSV file that labelled as grades(copy)

The email notification will be sent to the receiver that the file copying process is completed. Figure 3.3 below shows the email received after the process of copying file is done.

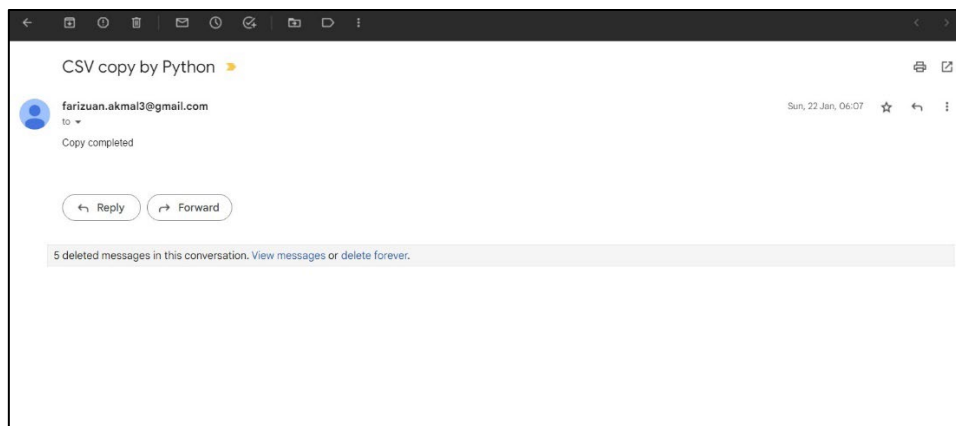


Figure 3.3: Email Notification received.

The automation of the process using Python makes it easier and quicker compared to manual data entry, reducing the likelihood of human error. The use of email notifications also eliminates the need for manual checks to confirm that the process has been completed, saving time and ensuring that the relevant parties are informed promptly. However, it is important to note that the accuracy of the data copied from the CSV file to the database depends on the quality of the data in the CSV file. If the data in the CSV file is inaccurate, the accuracy of the data in the database will also be affected. In such cases, it may be necessary to perform data validation before copying the data to the database to ensure that the data is correct.

CHAPTER 4: CONCLUSION

In conclusion, the use of Python to automate the process of copying data from a CSV file to a database with email notification is a valuable tool in data management. By making the process quicker and more efficient, it helps to reduce the likelihood of human error and ensures that the relevant parties are informed promptly.

