Step 1: Update Ubuntu and Install Required Packages

1. Update the package repository:
sudo apt update && sudo apt upgrade -y

2. Install Python & Required Libraries
sudo apt install python3 python3-pip -y
pip3 install cryptography pandas faker

python3 --version
pip3 list | grep -E "cryptography|pandas|faker"

```
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ sudo apt update && sudo apt upgrade
[sudo] password for student:
Hit:1 http://in.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [129 kB]
Get:3 http://hp.archive.canonical.com jammy InRelease [337 kB]
Get:4 http://in.archive.ubuntu.com/ubuntu jammy-updates InRelease [128 kB]
Get:5 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [601 kB]
Get:6 http://in.archive.ubuntu.com/ubuntu jammy-backports InRelease [127 kB]
Get:7 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [2,417 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [2,150 kB]
Get:9 http://in.archive.ubuntu.com/ubuntu jammy-updates/main i386 Packages [777 kB]
Get:10 http://in.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [400 kB]
Get:11 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 DEP-11 Metadata [103 kB]
Get:12 http://in.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [18.5 kB]
Get:13 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted i386 Packages [41.9 kB]
Get:14 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [3,164 kB]
Get:15 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [558 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [334 kB]
Get:17 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 DEP-11 Metadata [212 B]
Get:18 http://in.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [676 B]
Get:19 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [43.1 kB]
Get:20 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [1,195 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [13.6 kB]
Get:22 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe i386 Packages [761 kB]
Get:23 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [294 kB]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [40.1 kB]
Get:25 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 DEP-11 Metadata [359 kB]
Get:26 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [2,997 kB]
Get:27 http://in.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [28.7 kB]
Get:28 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse i386 Packages [5,048 B]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [529 kB]
Get:30 http://in.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [46.5 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 DEP-11 Metadata [208 B]
```

Step 2: AES Encryption & Decryption

AES (Advanced Encryption Standard) is a symmetric encryption algorithm.

1. Generate an AES Key : (Create nano aes_key.py)

from cryptography.fernet import Fernet

# Generate AES key
key = Fernet.generate_key()

Save the key in a file

```
with open("aes_key.key", "wb") as key_file:
        key_file.write(key)
print(f"Generated AES Key: {key.decode()}")
```

2. Encrypt a Message (Create nano aes_encrypt.py)

```
from cryptography.fernet import Fernet

# Load AES key
key = open("aes_key.key", "rb").read()
cipher = Fernet(key)

# Message to encrypt
message = "Confidential Data: Do not share!"

# Encrypt the message
encrypted_message = cipher.encrypt(message.encode())

        print(f"Encrypted Message: {encrypted_message.decode()}")
```

3. Decrypt the Message (Create nano aes_decrypt.py)

```
# Decrypt the message
decrypted_message = cipher.decrypt(encrypted_message).decode()

        print(f"Decrypted Message: {decrypted_message}")
```

```
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ sudo apt install python3 python3-pip -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.10.6-1~22.04.1).
python3 set to manually installed.
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2 libllvm13
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev fakeroot g++ g++-11 gcc gcc-11 javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan6 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-11-dev libitm1 libjs-jquery
  libjs-sphinxdoc libjs-underscore liblsan0 liblsl-dev libpython3-dev libpython3.10-dev libquadmath0 libstdc++-11-dev libtirpc-dev libtsan0 libubsan1 linux-libc-dev lto-disabled-list make manpages-dev
  python3-dev python3-distutils python3-setuptools python3-wheel python3.10-dev rpcsvc-proto zlib1g-dev
Suggested packages:
  binutils-doc debian-keyring g++-multilib g++-11-multilib gcc-11-doc gcc-multilib autoconf automake libtool flex bison gcc-doc gcc-11-multilib gcc-11-locales glibc-doc git bzr libstdc++-11-doc make-doc
  python-setuptools-doc
The following NEW packages will be installed:
  binutils binutils-common binutils-x86-64-linux-gnu build-essential dpkg-dev fakeroot g++ g++-11 gcc gcc-11 javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl libalgorithm-merge-perl
  libasan6 libbinutils libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libctf-nobfd0 libctf0 libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libgcc-11-dev libitm1 libjs-jquery
  libjs-sphinxdoc libjs-underscore liblsan0 liblsl-dev libpython3-dev libpython3.10-dev libquadmath0 libstdc++-11-dev libtirpc-dev libtsan0 libubsan1 linux-libc-dev lto-disabled-list make manpages-dev
  python3-dev python3-distutils python3-pip python3-setuptools python3-wheel python3.10-dev rpcsvc-proto zlib1g-dev
0 upgraded, 53 newly installed, 0 to remove and 0 not upgraded.
```

Step 3: RSA Encryption & Decryption

1. Generate RSA Key Pair (Create nano rsa_key.py)

```
from cryptography.hazmat.primitives.asymmetric import rsa
from cryptography.hazmat.primitives import serialization

# Generate private key
```

```
private_key = rsa.generate_private_key(public_exponent=65537, key_size=2048)

# Save private key
with open("rsa_private.pem", "wb") as f:
f.write(private_key.private_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PrivateFormat.TraditionalOpenSSL,
        encryption_algorithm=serialization.NoEncryption()
))

# Generate public key
public_key = private_key.public_key()
with open("rsa_public.pem", "wb") as f:
f.write(public_key.public_bytes(
        encoding=serialization.Encoding.PEM,
        format=serialization.PublicFormat.SubjectPublicKeyInfo
))

        print("RSA Key Pair Generated Successfully!")
```

2. Encrypt Data Using RSA Public Key (Create nano rsa_encrypt.py)

```
from cryptography.hazmat.primitives.asymmetric import padding
from cryptography.hazmat.primitives import hashes

# Load the public key
public_key = serialization.load_pem_public_key(open("rsa_public.pem", "rb").read())

message = b"Secure Data Transfer"

# Encrypt using the public key
encrypted = public_key.encrypt(
        message,
padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
)
)

print(f"Encrypted Data: {encrypted}")
```

3. Decrypt Data Using RSA Private Key (Create nano rsa_decrypt.py)

```
# Load the private key
private_key = serialization.load_pem_private_key(open("rsa_private.pem", "rb").read(),
password=None)

# Decrypt the message
decrypted = private_key.decrypt(
encrypted,
padding.OAEP(
        mgf=padding.MGF1(algorithm=hashes.SHA256()),
        algorithm=hashes.SHA256(),
        label=None
)
)

print(f"Decrypted Data: {decrypted.decode()}")
```

```
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ pip3 install cryptography pandas faker
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: cryptography in /usr/lib/python3/dist-packages (3.4.8)
Collecting pandas
  Downloading pandas-2.2.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (13.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 13.1/13.1 MB 24.2 MB/s eta 0:00:00
Collecting faker
  Downloading faker-37.1.0-py3-none-any.whl (1.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 9.1 MB/s eta 0:00:00
Collecting python-dateutil>=2.8.2
  Downloading python_dateutil-2.9.0.post0-py2.py3-none-any.whl (229 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 229.9/229.9 KB 2.8 MB/s eta 0:00:00
Collecting numpy>=1.22.4
  Downloading numpy-2.2.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (16.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 16.4/16.4 MB 20.6 MB/s eta 0:00:00
Collecting tzdata>=2022.7
  Downloading tzdata-2025.2-py2.py3-none-any.whl (347 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 347.8/347.8 KB 3.8 MB/s eta 0:00:00
Requirement already satisfied: pytz>=2020.1 in /usr/lib/python3/dist-packages (from pandas) (2022.1)
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Installing collected packages: tzdata, python-dateutil, numpy, pandas, faker
  WARNING: The scripts f2py and numpy-config are installed in '/home/student/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
  WARNING: The script faker is installed in '/home/student/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed faker-37.1.0 numpy-2.2.4 pandas-2.2.3 python-dateutil-2.9.0.post0 tzdata-2025.2
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 --version
Python 3.10.12
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ pip3 list | grep -E "cryptography|pandas|faker"
cryptography          3.4.8
pandas                2.2.3
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$
```

Step 4: Data Anonymization

1. Install Faker Library

sudo apt install faker

2. Masking Sensitive Data (Anonymization) (Create nano mask.py)

import pandas as pd

data = {"SSN": ["123-45-6789", "987-65-4321", "555-44-3333"]}

```
df = pd.DataFrame(data)

# Masking SSN (showing only last 4 digits)
df["Masked_SSN"] = df["SSN"].str.replace(r"\d{3}-\d{2}", "***-**-",
regex=True)

print(df)

2. Generate Fake Data for Privacy Protection

from faker import Faker

fake = Faker()

for _ in range(5):
        print(fake.name(), "-", fake.email(), "-", fake.phone_number())
```

```
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano aes_key.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 aes_key.py
Generated AES Key: Nl-Z8xYApYVBSNZ2CC78rK_9MfUic69BcFEu9xqSXzs=
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano aes_encrypt.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 aes_encrypt.py
Encrypted Message: gAAAAABn4k7Z5xMlffN7r4t3X7L8hldCJmKq1oxTyFymKxOYVXwXd4-zdaRDo2KUmSzv6T4EvFn8KBdJ-6QZtgllAtuax6thEi3TGgrRtaeWOnFOUbASVZf8y1ABEJQKRnH7YnpyCTJc
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano aes_decrypt.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ ^C
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ ^C
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano aes_decrypt.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 aes_decrypt.py
Decrypted Message: Confidential Data: Do not share!
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$
```

Step 5: Test and Verify Data Privacy

- Run encryption and anonymization scripts.
- Try to decrypt and view masked/anonymized data.
- Observe that data security policies prevent unauthorized access.

```
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano rsa_key.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 rsa_key.py
RSA Key Pair Generated Successfully!
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano rsa_encrypt.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 rsa_encrypt.py
Encrypted Data: b' T\xfeW\x86v\xb8\xd2\x87\x95\xed}H\n\x1fo\x9a\xbd\xe6\xddV\xaaY\xb0J\x85]vLkGbo\xc5\xa6S\x8a\xbf\x89b\xe9\xf7\xa6\xff\xa9\x81\x04\xd4\x11}\xfc\x9e\x9c\x03\x98y\x98qP\xaf\x1fT\xb6\\\xa6\x
d1#\x87\x8d\xe1#\xf5\xd4\x9d9a\xa3!\xdb\x82\xec\xa5\xf1 \xeaK\xb6\xd3E\xe1C"ff\xa3\xe9%\x08A\x1b\xb9\xc1B\x87zth\xc7P\x91W\xd1bi\x08\xa2\xfcli\xdb\x94+\xbe\xf3\xa2@\xc0y\xfd\xa9b\x13\x82v\x08\x83<\x83\xbc
\x0c6\x97b\'\x9a\xb3k\xef\xef\xaf\x8f\xa2\x9f\xa2i\x9f\xd9|S\x97\x06\xa5\x8d\xe2"\x16^\x10m\x8dR\xf1\x83K\x87av\xb2\x8d:\xc9\xa3J0\xba)\xfb\xe0\xfc\xe648\x18w\x97\xa8\xdd\xa7*\xdf\x14v\xd8rQ\xe9f\xd9\xfd\
xb1W\x0c\xd4\xbfv\x8e\xf9\x8bx\x1ff\xf7a\xd4\x8d\x84\xe2w \xf8\xbc\n\x85?x\xd5V\xcc\x92J^\xd3\x8e\xc4\'\xb6\x7f\xda\xc0\x83 \x0f\xd2\x9bG\xd6'
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano rsa_decrypt.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 rsa_decrypt.py
Decrypted Data: Secure Data Transfer
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$
```

```
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ nano mask.py
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$ python3 mask.py
           SSN    Masked_SSN
0  123-45-6789  ***-**--6789
1  987-65-4321  ***-**--4321
2  555-44-3333  ***-**--3333
student@student-HP-Z2-Tower-G9-Workstation-Desktop-PC:~$
```