**Assignment**

**CSA0805 – Python Programming**

|  |  |
| --- | --- |
| **Register Number** | **192324313** |
| **Name** | **Korasika sudheer** |

**Title:** **Automated Database Backup Script with Compression and Encryption in Python**

**Problem Statement:Develop a Python program that connects to a database and exports its schema and data to a SQL dump file for backup purposes, allowing users to specify options such as compression and encryption.**

**Code:**

**import os**

**import subprocess**

**import gzip**

**import shutil**

**from cryptography.fernet import Fernet**

**def backup\_database(host, user, password, database, output\_file, compress=False, encrypt=False, encryption\_key=None):**

**"""**

**Backup a database by exporting its schema and data to a SQL dump file.**

**Parameters:**

**- host: Database host (e.g., localhost)**

**- user: Database user**

**- password: Database user's password**

**- database: Database name**

**- output\_file: Output file path for the SQL dump**

**- compress: Boolean indicating if the output should be compressed**

**- encrypt: Boolean indicating if the output should be encrypted**

**- encryption\_key: Key for encryption (if encryption is enabled)**

**"""**

**# Step 1: Create a dump of the database**

**dump\_command = f"mysqldump -h {host} -u {user} -p{password} {database} > {output\_file}"**

**subprocess.run(dump\_command, shell=True, check=True)**

**# Step 2: Compress the dump file if compression is enabled**

**if compress:**

**with open(output\_file, 'rb') as f\_in:**

**with gzip.open(output\_file + '.gz', 'wb') as f\_out:**

**shutil.copyfileobj(f\_in, f\_out)**

**os.remove(output\_file)**

**output\_file += '.gz'**

**# Step 3: Encrypt the file if encryption is enabled**

**if encrypt:**

**if not encryption\_key:**

**raise ValueError("Encryption key must be provided if encryption is enabled")**

**encrypt\_file(output\_file, encryption\_key)**

**if compress:**

**os.remove(output\_file) # Remove the unencrypted .gz file**

**output\_file += '.enc'**

**print(f"Backup completed successfully. File saved to: {output\_file}")**

**def encrypt\_file(file\_path, key):**

**"""**

**Encrypt the given file using the provided encryption key.**

**Parameters:**

**- file\_path: Path to the file to be encrypted**

**- key: Encryption key**

**"""**

**cipher = Fernet(key)**

**with open(file\_path, 'rb') as f:**

**data = f.read()**

**encrypted\_data = cipher.encrypt(data)**

**with open(file\_path + '.enc', 'wb') as f:**

**f.write(encrypted\_data)**

**os.remove(file\_path)**

**def main():**

**"""**

**Main function to gather user input and perform the backup operation.**

**"""**

**# Collect user input for database connection and backup options**

**host = input("Enter database host (e.g., localhost): ")**

**user = input("Enter database user: ")**

**password = input("Enter database password: ")**

**database = input("Enter database name: ")**

**output\_file = input("Enter output file name (without extension): ") + ".sql"**

**compress = input("Enable compression? (y/n): ").lower() == 'y'**

**encrypt = input("Enable encryption? (y/n): ").lower() == 'y'**

**encryption\_key = None**

**if encrypt:**

**encryption\_key = input("Enter encryption key (or type 'generate' to create a new one): ")**

**if encryption\_key.lower() == 'generate':**

**encryption\_key = Fernet.generate\_key()**

**print(f"Generated encryption key: {encryption\_key.decode()}")**

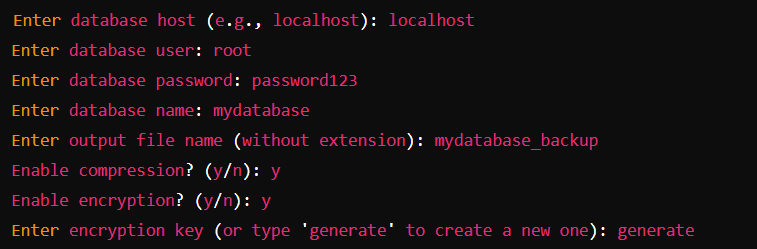
**# Perform the database backup**

**backup\_database(host, user, password, database, output\_file, compress, encrypt, encryption\_key)**

**if \_\_name\_\_ == "\_\_main\_\_":**

**main()**

**Output Screen Shots:**

****

****

**Conclusion:**

**This Python script is a versatile tool for automating secure and efficient database backups. With options for compression and encryption, it ensures space-efficient and protected backups, making it easy to integrate into any database management routine to safeguard critical data.**