PROG8850- Database Automation

Assignment 1 (Total: 20 Points)

Assignment 1: Database Automation and Scripting

This assignment assesses your understanding of database automation concepts and your ability to create Python scripts for automating database management tasks.

Question 1: Understanding Database Automation (10 Points)

1.1) Explain database automation and its significance in modern data management. Highlight the role of automation in handling large volumes of data efficiently and securely.

Any database automation refers to the application of tools and scripts for carrying out various database operations. Hence, all processes are now manual, at least with some operators' intervention. These processes would include automatic backups, checks on performance, updates, and the modification of security settings.

Modern times mean that companies are dealing with huge volumes of data. It takes an awful lot of time and is a potential source of errors, hence the various database operations should be automated. This gives speed and accuracy, especially in cases of large systems.

The core benefit of automation is to automate operations, diminish or eliminate manual intervention, and associate the consistency in these processes. From round to round, automation offers businesses through consistent, reliable, fast, and secure database services. Scheduled housekeeping operations, such as backup and update, can be performed automatically at the right time and in a reliable manner. This reduces the chances of data loss and system downtimes.

Performance-tuning software automatedly identifies slow queries, inefficient index usage, and other such problems, commonly fixing some of them without any human intervention. From a security perspective, automation means that patches, control access, and monitor must be applied in a timely manner to avoid vulnerabilities and exploitation. Automation also guarantees compliance with regulatory standards that apply to the industry.

With automation, scaling is a task of sheer simplicity at its best. On increased user demand, automated systems can immediately scale resources or clone databases without even a bit of time wastage.

The fully automated database services from AWS, Azure, and Google Cloud sit on top of their major cloud infrastructure services. They implement all measures to ascertain the high availability, disaster recovery, and performance optimization of the services to the minimal effort of the user.

Database automation, summarized in essence, is a must for any modern data operation today. It allows for the immense dataset to be securely, efficiently, and reliably managed, thus giving room to focus on innovation instead of maintenance.

1.2) Analyze the benefits of automating database tasks, including reduced errors, increased reliability, faster deployments, and cost efficiency. Support your analysis with real-world examples or case studies where possible.

Database-process automation has many of benefits augmenting performance and lessening operational burden. The main advantages include fewer errors, consistency, quicker implementation, and cost reduction.
Fewer Errors: When humans are involved, often there is a risk of human error: an incorrect command might be typed in, or an important update could be overlooked. Scripted procedures automate these tasks, thereby reducing possibilities for human error.
Example: In finance, automation ensures transactional data protection through scheduled backups and replications.
Improved Reliability: It treats repetitive manual actions consistently, even in the dead of the night. Different forms of health checks, backup routines, and patch installations are set on auto-run to increase the stability of systems.
Example: Amazon RDS automates maintenance and failovers to ensure operations remain smooth without disrupting user experience.
Faster Implementations: Terraform and Flyway, etc., allow rapid deployment and updating of databases. This decreases the length of the development cycle, therefore putting new features or a fix immediately to production.
Cost Savings: With automation, constant manual oversight can be reduced, plus intelligent use of resources further helps in slashing costs related to labor and infrastructure.
Example: Netflix auto-scales database resources depending on traffic.
Smarter Monitoring: Automated activities monitor the performance metrics 24/7, raising alerts, and corrective measures when required, thus, minimizing downtime and major failure occurrences.
Basically, automated database operations provide a more intelligent and better way of managing complex systems, giving an edge to companies on optimization of resources, cost reduction, and guarantee of stability and compliance.

Question 2: Scripting for Database Automation (10 points)

2.1) **Python Scripting for Database Backup Automation** (5 points)

Write a Python script to automate the process of creating backups for MySQL databases. Explain the script's logic, focusing on how it connects to the MySQL database and ensures a unique filename for each backup.

```
import os

import datetime

import subprocess



MYSQL_USER = 'root'

MYSQL_PASSWORD = 'pass'

MYSQL_HOST = 'localhost'

DATABASE_NAME = 'database_name'

BACKUP_DIR = 'backups'



def Create_b():

    if not os.path.exists(BACKUP_DIR):

        os.makedirs(BACKUP_DIR)



    time_s = datetime.datetime.now().strftime('%Y%m%d_%H%M%S')

    File_backup = f"{DATABASE_NAME}_backup_{time_s}.sql"

    path_backup = os.path.join(BACKUP_DIR, File_backup)



    dump_command = [

        'mysqldump',

        f'-h{MYSQL_HOST}',

        f'-u{MYSQL_USER}',

        f'-p{MYSQL_PASSWORD}',
```

```
        DATABASE_NAME
    ]

    with open(path_backup, 'w') as backup_file:
        try:
            subprocess.run(dump_command, stdout=backup_file, check=True)
            print(f"backup_succes: {path_backup}")
        except subprocess.CalledProcessError as e:
            print("backup_failed:", e)

if __name__ == "__main__":
    Create_b()
```

2.2) **Python Scripting for Database Change Deployment** (5 points)

Write a Python script to automate the deployment of database changes, such as adding a new table or column. Describe the script's functionality and how it handles the deployment process.

```
import mysql.connector
from mysql.connector import Error

DB_CONFIG = {
    'host': 'localhost',
    'user': 'root',
    'password': 'pass',
    'database': 'database'
```

```
}


SQL_CHANGES = [

]


def Dep_change():

    try:

        connection = mysql.connector.connect(**DB_CONFIG)

        cursor = connection.cursor()


        print("Connected to the database.")


        for change in SQL_CHANGES:

            try:

                print(f"Executing:\n{change.strip()}")

                cursor.execute(change)

                print("Change applied.")

            except Error as err:

                print(f"Failed to apply change:\n{err}")


        connection.commit()

        print("Changes Commited")
```

```
    except Error as e:
        print(f"Error encounter in SQL: {e}")
    finally:
        if connection.is_connected():
            cursor.close()
            connection.close()
            print("Connection closed.")


if __name__ == "__main__":
    Dep_change()
```

Submission Instructions:

- Submit a single ZIP file containing:
    - A word/ PDF file with your answers.
    - Python scripts named ‘backup_script.py’ and ‘deploy_changes_script.py’ -

  Ensure all answers are clear and well-organized - Cite any references used.