AWS offers several managed database services, including:

1. **Amazon RDS (Relational Database Service)**:
   * Supports multiple database engines like MySQL, PostgreSQL, Oracle, and SQL Server.
   * Automated backups, scaling, and patching.
2. **Amazon DynamoDB**:
   * A fully managed NoSQL database service that provides fast and predictable performance with seamless scalability.
   * Ideal for applications that need low-latency data access.
3. **Amazon Aurora**:
   * Amazon Aurora is a relational database management system (RDBMS) built for the cloud with full MySQL and PostgreSQL compatibility. Aurora gives you the performance and availability of commercial-grade databases at one-tenth the cost.
4. **Amazon Redshift**:
   * Amazon Redshift uses SQL to analyze structured and semi-structured data across data warehouses, operational databases, and data lakes, using AWS-designed hardware and machine learning to deliver the best price performance at any scale.
5. **Amazon ElastiCache**:
   * Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale a distributed in-memory data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution. At the same time, it helps remove the complexity associated with deploying and managing a distributed cache environment.

Using Amazon RDS with Python

**Prerequisites**

1. **AWS Account**: You need an AWS account.
2. **RDS Instance**: Create a MySQL RDS instance via the AWS Management Console.
3. **Python Environment**: Ensure you have Python installed along with mysql-connector-python.

**Step 1: Setting Up RDS**

1. Log into the AWS Management Console.
2. Navigate to RDS and create a new MySQL database instance.
3. Take note of the endpoint, username, and password for your database.

**Step 2: Install Required Packages**

Install the MySQL connector for Python:

pip install mysql-connector-python

**Step 3: Create a Python Program**

Here’s a simple program that connects to your RDS MySQL database, creates a table, inserts some data, and retrieves it.

import mysql.connector

from mysql.connector import Error

def create\_connection(host\_name, user\_name, user\_password, db\_name):

connection = None

try:

connection = mysql.connector.connect(

host=host\_name,

user=user\_name,

password=user\_password,

database=db\_name

)

print("Connection to MySQL DB successful")

except Error as e:

print(f"The error '{e}' occurred")

return connection

def execute\_query(connection, query):

cursor = connection.cursor()

try:

cursor.execute(query)

connection.commit()

print("Query executed successfully")

except Error as e:

print(f"The error '{e}' occurred")

def read\_query(connection, query):

cursor = connection.cursor()

cursor.execute(query)

return cursor.fetchall()

# Database credentials

host = "your-rds-endpoint" # e.g. "your-db-instance.abcdefghijk.us-west-2.rds.amazonaws.com"

user = "your-username"

password = "your-password"

database = "your-database"

# Create connection

connection = create\_connection(host, user, password, database)

# Create table

create\_users\_table = """

CREATE TABLE IF NOT EXISTS users (

id INT AUTO\_INCREMENT PRIMARY KEY,

name TEXT NOT NULL,

age INT,

created\_at TIMESTAMP DEFAULT CURRENT\_TIMESTAMP

) ENGINE = InnoDB

"""

execute\_query(connection, create\_users\_table)

# Insert data

insert\_user = "INSERT INTO users (name, age) VALUES ('Alice', 30)"

execute\_query(connection, insert\_user)

insert\_user2 = "INSERT INTO users (name, age) VALUES ('Bob', 25)"

execute\_query(connection, insert\_user2)

# Read data

select\_users = "SELECT \* FROM users"

users = read\_query(connection, select\_users)

for user in users:

print(user)

### Step : Run the Python script to create the table, insert users, and print the user data.