Programming, Ops and Database Exercise

Using AWS RDS create a simple mysql database with the following SQL commands:

```
-- Drop tables if they exist
DROP TABLE IF EXISTS order_items;
DROP TABLE IF EXISTS orders;
DROP TABLE IF EXISTS products;
DROP TABLE IF EXISTS customers;

-- Customers table
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    country VARCHAR(50)
);

-- Products table
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    category VARCHAR(50),
    price DECIMAL(10,2)
);

-- Orders table
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    order_date DATE,
    status VARCHAR(20),
```

```
    FOREIGN KEY (customer_id) REFERENCES customers(customer_id)
);

-- Order Items table
CREATE TABLE order_items (
    order_item_id INT PRIMARY KEY,
    order_id INT,
    product_id INT,
    quantity INT,
    unit_price DECIMAL(10,2),
    FOREIGN KEY (order_id) REFERENCES orders(order_id),
    FOREIGN KEY (product_id) REFERENCES products(product_id)
);
```

Insert data into your database using the following script:

```
-- Customers
INSERT INTO customers VALUES
(1, 'Alice Smith', 'alice@example.com', 'USA'),
(2, 'Bob Jones', 'bob@example.com', 'Canada'),
(3, 'Charlie Zhang', 'charlie@example.com', 'UK');

-- Products
INSERT INTO products VALUES
(1, 'Laptop', 'Electronics', 1200.00),
(2, 'Smartphone', 'Electronics', 800.00),
(3, 'Desk Chair', 'Furniture', 150.00),
(4, 'Coffee Maker', 'Appliances', 85.50);

-- Orders
INSERT INTO orders VALUES
(1, 1, '2023-11-15', 'Shipped'),
```

(2, 2, '2023-11-20', 'Pending'),

(3, 1, '2023-12-01', 'Delivered'),

(4, 3, '2023-12-03', 'Cancelled');

-- Order Items

INSERT INTO order_items VALUES

(1, 1, 1, 1, 1200.00), -- Laptop

(2, 1, 4, 2, 85.50), -- Coffee Maker

(3, 2, 2, 1, 800.00), -- Smartphone

(4, 3, 3, 2, 150.00), -- Desk Chair

(5, 4, 1, 1, 1200.00); -- Laptop

Write and execute the following Queries:

- Top Customers by Spending
- Monthly Sales Report (Only Shipped/Delivered)
- Products Never Ordered
- Average Order Value by Country
- Frequent Buyers (More Than One Order)

  Hint: These are complex queries.

Challenge (Optional)

Set up a simple API using whatever programming language of your choice with endpoints for each of these queries. Document this and share the API documentation in a sub directory for grading.

Submission: Package all database and query scripts in a well-documented manner. Package screenshots of your query results in a well-documented manner. Include the screenshots in a sub directory in your scripts package and push to GitHub. Submit GitHub link in a form that will be provided.

Bash Scripting Challenge Lab

Here's a comprehensive lab exercise for learners to write a Bash script to automate Identity and Access Management (IAM) tasks in Linux.

Lab: Automating Identity and Access Management in Linux with Bash

Objective

By the end of this lab, learners will:

- Understand the basics of automating IAM tasks.

- Write a Bash script to automate user and group management.

- Implement password policies and permission settings via script.

Prerequisites

- A Linux machine (physical, VM, or WSL)

- Basic knowledge of Bash scripting

- Familiarity with `useradd`, `usermod`, `passwd`, and `chage`

Scenario

You are a system administrator for a mid-sized company. A new department requires multiple user accounts to be created, each with a specific home directory, group membership, and password policy. Your task is to automate this process to save time and reduce human error.

Tasks

Step 1: Setup

Create a file named `users.txt` with the following format:

```
username,fullname,group
```

jdoe,John Doe,engineering

asmith,Alice Smith,engineering

mjones,Mike Jones,design

Step 2: Script Requirements

Create a Bash script named `iam_setup.sh` that does the following:

1. **Create groups**: If a group in the file doesn’t exist, create it.

2. **Create users**:

   - Add users with the given username.

   - Set their full name.

   - Assign them to the specified group.

   - Create a home directory for each user.

   - Set a temporary password (e.g., `ChangeMe123`).

   - Force password change on first login.

3. **Set permissions**:

   - Ensure each user's home directory is only accessible by that user (`chmod 700`).

4. **Logging**:

   - Log each action to a file named `iam_setup.log` with a timestamp.

Deliverables

- `users.txt` input file.

- `iam_setup.sh` script.

- `iam_setup.log` with sample run logs.

Testing

Run the script and verify:

- Users are created.

- Groups are created.

- Password policy is enforced.

- Permissions on `/home/username` are `700`.

- Log file contains all relevant actions.

Challenge (Optional)

- Add email notifications for each user created (hint: `mail` command).

- Integrate password complexity check.

- Accept CSV as a command-line argument.

Submission: Package all scripts and logs in a well-documented manner. Package screenshots of your script execution manner. Include the screenshots in a sub directory in your scripts package and push to GitHub. Submit GitHub link in a form that will be provided.