Database administration

DCL

1. **Create a Role for Database Read Operations:**

```
CREATE ROLE readonly;
```

   - `CREATE ROLE`: Creates a new role in the database.
   - Creates a role named `readonly` for database read operations.

2. **Create a Role for Database Write Operations:**

```
CREATE ROLE readwrite;
```

   - Creates a role named `readwrite` for database write operations.

3. **Create a User with a Password:**

```
CREATE USER mykola WITH PASSWORD 'securePassword';
```

   - `CREATE USER`: Creates a new database user.
   - Creates a user named `mykola` with the password 'securePassword'.

4. **Grant Role to a User:**

```
GRANT readonly TO mykola;
```

   - `GRANT`: Assigns privileges or roles to users or roles.
   - Grants the `readonly` role to the user `mykola`.

5. **Create Another User with More Privileges:**

```
CREATE USER patron WITH PASSWORD 'anotherSecurePassword';
GRANT readwrite TO patron;
```

- Creates a user named `patron` with the password 'anotherSecurePassword' and grants the `readwrite` role to this user.

6. **Grant Select Privilege to Readonly Role:**

```
GRANT SELECT ON ALL TABLES IN SCHEMA myshop TO readonly;
```

- Grants the `SELECT` privilege on all tables in the `myshop` schema to the `readonly` role.

7. **Grant Select, Insert, Update, Delete Privileges to Readwrite Role:**

```
GRANT SELECT, INSERT, UPDATE, DELETE ON ALL TABLES IN SCHEMA myshop TO
readwrite;
```

- Grants `SELECT`, `INSERT`, `UPDATE`, and `DELETE` privileges on all tables in the `myshop` schema to the `readwrite` role.

8. **Grant Privileges on Future Tables Automatically:**

```
ALTER DEFAULT PRIVILEGES IN SCHEMA myshop GRANT SELECT ON TABLES TO
readonly;
ALTER DEFAULT PRIVILEGES IN SCHEMA myshop GRANT SELECT, INSERT, UPDATE,
DELETE ON TABLES TO readwrite;
```

- **`ALTER DEFAULT PRIVILEGES`**: Changes the default access privileges for objects created in the future.
- Ensures that any new tables created in the `myshop` schema will automatically grant `SELECT` privilege to the `readonly` role and `SELECT`, `INSERT`, `UPDATE`, `DELETE` privileges to the `readwrite` role.

9. **Grant Update Privileges on Specific Table:**

```
GRANT UPDATE ON myshop.products TO readwrite;
```

- Grants `UPDATE` privilege on the `products` table in the `myshop` schema to the `readwrite` role.

10. **Revoke Delete Privileges from Specific Table:**

```
REVOKE DELETE ON myshop.clients FROM readwrite;
```

- `REVOKE`: Removes privileges or roles from users or roles.
- Revokes the `DELETE` privilege on the `clients` table in the `myshop` schema from the `readwrite` role.

11. **Temporarily Add User to Readwrite Group:**

```
GRANT readwrite TO mykola;
```

- Temporarily grants the `readwrite` role to the user `mykola`.

12. **Remove User from Readwrite Group:**

```
REVOKE readwrite FROM mykola;
```

- Removes the `readwrite` role from the user `mykola`.

13. **Allow Readwrite Role to Create New Tables:**

```
ALTER ROLE readwrite CREATEROLE CREATEDB;
```

- `ALTER ROLE`: Modifies attributes of a database role.
- Grants the `readwrite` role the ability to create new roles (`CREATEROLE`) and databases (`CREATEDB`).

14. **Set a Role to Login:**

```
ALTER ROLE readonly LOGIN;
```

- Enables the `readonly` role to be used for logging into the database.

15. **View Role Table Grants:**

```sql
SELECT grantee, privilege_type
FROM information_schema.role_table_grants
WHERE grantee IN ('readonly', 'readwrite');
```

- `information_schema`: A system schema that provides access to database metadata.
- `role_table_grants`: A view that shows the privileges granted on tables.
- Retrieves the list of privileges (`privilege_type`) granted to the roles `readonly` and `readwrite`.

Transactions

1. **Simple Transaction with Commit:**

```sql
BEGIN;

INSERT INTO products (name, quantity) VALUES ('Tablet', 10);

COMMIT;
```

- `BEGIN`: Starts a new transaction.
- `INSERT INTO`: Adds new rows to a table.
- `COMMIT`: Ends the transaction, making all changes permanent.
- Inserts a product named 'Tablet' with a quantity of 10 into the `products` table and commits the transaction.

2. **Transaction with Rollback Due to Error:**

```sql
BEGIN;

INSERT INTO clients (name) VALUES ('Petro');
-- Assuming there is a business rule that an order can't be placed without specifying a product
INSERT INTO orders (order_id, client_id) VALUES (3, (SELECT client_id FROM clients WHERE name = 'Petro'));

-- If the order addition fails or is invalid
ROLLBACK;
```

- `ROLLBACK`: Ends the transaction, discarding all changes made during the transaction.
- Starts a transaction, inserts a client named 'Petro', and attempts to insert an order for 'Petro'. If the order insertion fails due to a business rule or error, the transaction is rolled back, discarding all changes.

3. **Transaction with Savepoints:**

```sql
BEGIN;

-- Add a new client
INSERT INTO clients (name) VALUES ('Maria');

-- Create a savepoint after adding the client
SAVEPOINT client_added;

-- Attempt to add an order
INSERT INTO orders (order_id, client_id)
VALUES (4, (SELECT client_id FROM clients WHERE name = 'Maria'));

-- Assume the order addition failed due to some validation or check
ROLLBACK TO SAVEPOINT client_added;

-- Try adding a different order
INSERT INTO orders (order_id, client_id)
VALUES (5, (SELECT client_id FROM clients WHERE name = 'Maria'));

COMMIT;
```

- `SAVEPOINT`: Sets a savepoint within a transaction to which you can later roll back.
- `ROLLBACK TO SAVEPOINT`: Rolls back part of a transaction to a savepoint.
- Starts a transaction, inserts a client named 'Maria', and creates a savepoint. Attempts to insert an order for 'Maria' and if it fails, rolls back to the savepoint, then tries to insert a different order and commits the transaction.

4. **Transaction with Multiple Inserts and Conditional Commit or Rollback:**

```sql
BEGIN;

-- Add a new category
INSERT INTO categories (name) VALUES ('Gadgets');

-- Add a new product linked to the newly added category
INSERT INTO products (name, quantity) VALUES ('Smartwatch', 50);
INSERT INTO product_category (product_id, category_id)
VALUES ((SELECT product_id FROM products WHERE name = 'Smartwatch'),
        (SELECT category_id FROM categories WHERE name = 'Gadgets'));

-- Add a new client and an order for that client
INSERT INTO clients (name) VALUES ('Eva');
INSERT INTO orders (order_id, client_id)
VALUES (6, (SELECT client_id FROM clients WHERE name = 'Eva'));

-- After checking all operations succeed
COMMIT;

-- If there's an error in any step
ROLLBACK;
```

- Starts a transaction, inserts a new category 'Gadgets', inserts a new product 'Smartwatch' and links it to 'Gadgets', inserts a new client 'Eva', and creates an order for 'Eva'. If all operations succeed, commits the transaction. If any step fails, rolls back the entire transaction.