Certainly! Views in a relational database are virtual tables that are based on the result of a SELECT query. They allow you to present data in a structured way without actually storing the data physically. Here are two examples for each of the tasks you mentioned:

1. Creating Views:

Example 1: Basic View

```sql
-- Creating a basic view that selects specific columns from a table
CREATE VIEW employee_view AS
SELECT employee_id, first_name, last_name
FROM employees;
```

In this example, a view named `employee_view` is created, which includes only the `employee_id`, `first_name`, and `last_name` columns from the `employees` table.

Example 2: Joining Tables in a View

```sql
-- Creating a view that joins two tables
CREATE VIEW sales_summary AS
SELECT s.sales_id, s.product_id, p.product_name, s.quantity, p.unit_price
FROM sales s
JOIN products p ON s.product_id = p.product_id;
```

This example creates a view named `sales_summary` by joining the `sales` and `products` tables on the `product_id` column. It includes information about sales, such as the sales ID, product ID, product name, quantity, and unit price.

2. Updating Views:

Example 1: Updating Data Through a View

```sql
-- Updating data through a view
UPDATE employee_view
SET last_name = 'Doe'
WHERE employee_id = 101;
```

Here, the `employee_view` is a view that includes the `last_name` column. You can update the `last_name` of an employee directly through the view.

Example 2: Using Joins in an Updatable View

```sql
-- Updating data through a view with a join
UPDATE sales_summary
SET unit_price = 25.99
WHERE product_id = 102;
```

In this example, the `sales_summary` view is updated by changing the `unit_price` for products with a specific `product_id`.

3. Dropping Views:

Example 1: Dropping a Basic View

```sql
-- Dropping a view
DROP VIEW employee_view;
```

This SQL statement removes the `employee_view` from the database, effectively dropping the view.

Example 2: Dropping a View with Cascade

```sql
-- Dropping a view with CASCADE to automatically drop dependent views
DROP VIEW IF EXISTS sales_summary CASCADE;
```

The `CASCADE` option is used here to automatically drop dependent views. If other views depend on `sales_summary`, they will be dropped as well.

These examples provide a basic understanding of creating, updating, and dropping views in a relational database using SQL. The actual syntax may vary depending on the database management system you are using (e.g., MySQL, PostgreSQL, Oracle).

Certainly! Let's explore additional examples for creating, updating, and dropping views:

4. Creating Views:

Example 3: Aggregating Data in a View

```sql
-- Creating a view with aggregated data
CREATE VIEW department_summary AS
SELECT department_id, COUNT(employee_id) AS num_employees, AVG(salary) AS
avg_salary
FROM employees
GROUP BY department_id;
```

This view, named `department_summary`, provides a summary of each department by counting the number of employees and calculating the average salary.

Example 4: Creating a View with a WHERE Clause

```sql
-- Creating a view with a WHERE clause
CREATE VIEW high_salary_employees AS
SELECT employee_id, first_name, last_name, salary
FROM employees
WHERE salary > 80000;
```

This example creates a view named `high_salary_employees` that includes employees with a salary greater than 80,000.

5. Updating Views:

Example 3: Updating Data Through a Join View

```sql
-- Updating data through a view with a join
UPDATE department_summary
SET avg_salary = avg_salary + 5000
WHERE department_id = 2;
```

Here, the `department_summary` view is updated by increasing the average salary for employees in a specific department (department_id = 2).

Example 4: Updating Data Through a Conditional View

```sql
-- Updating data through a view with a condition
UPDATE high_salary_employees
SET salary = salary * 1.1
WHERE last_name = 'Smith';
```

This example updates the salary of employees in the `high_salary_employees` view, specifically those with the last name 'Smith', by increasing it by 10%.

6. Dropping Views:

Example 3: Dropping Views with Dependencies

```sql
-- Dropping a view without CASCADE, showing an error if there are dependencies
DROP VIEW IF EXISTS department_summary;
```

If there are dependencies on the `department_summary` view, such as other views or procedures, this statement will fail. You'll need to drop the dependent objects first or use `CASCADE` if you want to drop them automatically.

Example 4: Dropping Views in a Specific Schema

```sql
-- Dropping a view in a specific schema
DROP VIEW schema_name.view_name;
```

If your database supports schemas, this example demonstrates how to drop a view located in a specific schema (`schema_name` in this case).

These additional examples provide a more comprehensive view of creating, updating, and dropping views in a database using SQL. Remember to adapt the syntax to the specific database management system you are working with.