Shape

Description automatically generated with medium confidence

 **Task 1**

* **Objective:** Retrieve data from a table.
* **Exercise:** Write a query to select all columns from a table named employees.

 **Task 2**

* **Objective:** Retrieve specific columns.
* **Exercise:** Write a query to select the name and salary columns from the employees table.

 **Task 3**

* **Objective:** Filter results using the WHERE clause.
* **Exercise:** Write a query to select employees with a salary greater than 50,000.

 **Task 4**

* **Objective:** Use the LIKE operator for pattern matching.
* **Exercise:** Write a query to select employees whose names start with 'J'.

 **Task 5**

* **Objective:** Use the IN operator to filter data.
* **Exercise:** Write a query to select employees who belong to departments 1, 2, or 3.

 **Task 6**

* **Objective:** Sort query results.
* **Exercise:** Write a query to select all employees and order them by their hire date in descending order.

 **Task 7**

* **Objective:** Count the number of rows.
* **Exercise:** Write a query to count the number of employees.

 **Task 8**

* **Objective:** Calculate the sum of a column.
* **Exercise:** Write a query to calculate the total salary of all employees.

 **Task 9**

* **Objective:** Calculate the average of a column.
* **Exercise:** Write a query to find the average salary of employees.

 **Task 10**

* **Objective:** Find the minimum and maximum values.
* **Exercise:** Write queries to find the minimum and maximum salary of employees.

 **Task 11**

* **Objective:** Differentiate between WHERE and HAVING.
* **Exercises:**
  + Use WHERE to filter employees with a salary greater than 50,000.
  + Use HAVING to filter departments with an average salary greater than 50,000 after grouping.

 **Task 12**

* **Objective:** Understand and create one-to-one relationships.
* **Exercise:** Write SQL queries to create tables employees and employee\_details with a one-to-one relationship using a foreign key.

 **Task 13**

* **Objective:** Understand and create one-to-many relationships.
* **Exercise:** Write SQL queries to create tables departments and employees with a one-to-many relationship where one department has many employees.

 **Task 14**

* **Objective:** Understand and create many-to-many relationships.
* **Exercise:** Write SQL queries to create tables students, courses, and student\_courses to represent students enrolled in multiple courses and courses having multiple students.

Shape

Description automatically generated with medium confidence

 **Task 15**

* **Objective:** Implement foreign keys to enforce referential integrity.
* **Exercise:** Write SQL queries to add a foreign key to the employees table referencing the departments table.

 **Task 16**

* **Objective:** Ensure certain columns cannot have NULL values.
* **Exercise:** Write SQL queries to modify the employees table to ensure the name column cannot be NULL.

 **Task 17**

* **Objective:** Ensure all values in a column are unique.
* **Exercise:** Write SQL queries to modify the employees table to ensure the email column has unique values.

 **Task 18**

* **Objective:** Enforce specific rules for column values.
* **Exercise:** Write SQL queries to add a CHECK constraint to the employees table to ensure the salary is greater than 0.

 **Task 19**

* **Objective:** Set default values for columns.
* **Exercise:** Write SQL queries to modify the employees table to set a default value of 'Active' for the status column.

 **Task 20**

* **Objective:** Apply all learned concepts in a complete database system.
* **Task:** Write SQL queries to design and create a database for a company with the following requirements:
  + Tables: departments, employees, projects, employee\_projects.
  + Relationships: One-to-many between departments and employees, many-to-many between employees and projects.
  + Constraints: Appropriate use of NOT NULL, UNIQUE, CHECK, DEFAULT, and foreign keys.
  + Queries: Write queries to demonstrate use of SELECT, WHERE, LIKE, IN, ORDER BY, aggregation functions, WHERE vs. HAVING.