Lecture 9 Part 1: Advanced SQL

Introduction

This lecture explores advanced SQL techniques, focusing on:

- Filtering, sorting, and aggregation
- Removing duplicates and handling NULL values
- Complex subqueries
- Joins and relationships
- Data constraints and table modifications

Advanced Querying

Filtering Rows

Retrieve employees from department 90:

```
SELECT employee_id, last_name, job_id, department_id
FROM employees
WHERE department_id = 90;
```

Find employees with a salary between 2500 and 3500:

```
SELECT last_name, salary
FROM employees
WHERE salary BETWEEN 2500 AND 3500;
```

String Operations

Find employees whose last names start with 'M':

```
SELECT last_name
FROM employees
WHERE last_name LIKE 'M%';
```

Sorting

Sort employees by hire date (newest first):

```
SELECT last_name, department_id, hire_date
FROM employees
ORDER BY hire_date DESC;
```

Removing Duplicates

Find distinct job titles:

```
SELECT DISTINCT job_title
FROM jobs;
```

Aggregation and Grouping

Find the average salary for each job title:

```
SELECT job_id, AVG(salary) AS avg_salary
FROM employees
GROUP BY job_id;
```

Count employees in each department:

```
SELECT department_id, COUNT(*) AS employee_count
FROM employees
GROUP BY department_id;
```

Joins and Relationships

Inner Join

Get employee names and their department names:

```
SELECT e.first_name, e.last_name, d.department_name
FROM employees e
INNER JOIN departments d
ON e.department_id = d.department_id;
```

Left Join

Show all departments and employees, even if no employees exist:

```
SELECT d.department_name, e.first_name, e.last_name
FROM departments d
LEFT JOIN employees e
ON d.department_id = e.department_id;
```

Self Join

Find employees and their managers:

```
SELECT e1.first_name AS Employee, e2.first_name AS Manager
FROM employees e1
LEFT JOIN employees e2
ON e1.manager_id = e2.employee_id;
```

Subqueries

Single-Row Subquery

Find employees who have the same job as employee 141 and earn more than employee 143:

```
SELECT last_name, job_id, salary
FROM employees
WHERE job_id = (SELECT job_id FROM employees WHERE employee_id =
   141)
AND salary > (SELECT salary FROM employees WHERE employee_id =
   143);
```

Multiple-Row Subquery

Find employees with a salary higher than any employee in IT department:

```
SELECT employee_id, last_name, job_id, salary
FROM employees
WHERE salary > ANY (SELECT salary FROM employees WHERE job_id = '
   IT_PROG');
```

Using EXISTS

Find departments that have at least one employee:

```
SELECT department_name
FROM departments d
WHERE EXISTS (SELECT 1 FROM employees e WHERE e.department_id = d
   .department_id);
```

Using NOT EXISTS

Find departments that have no employees:

```
SELECT department_name
FROM departments d
WHERE NOT EXISTS (SELECT 1 FROM employees e WHERE e.department_id
    = d.department_id);
```

Data Modifications

Updating Data

Increase salary of all sales representatives by 10%:

```
UPDATE employees
SET salary = salary * 1.1
WHERE job_id = 'SALES_REP';
```

Deleting Data

Delete employees who were hired before 2008:

```
DELETE FROM employees
WHERE hire_date < '2008-01-01';
```

Copying Data

Create a backup of the Employees table:

```
SELECT * INTO employees_backup FROM employees;
```

Constraints and Table Modifications

Adding Primary Key

```
ALTER TABLE employees ADD CONSTRAINT pk_employees PRIMARY KEY (
    employee_id);
```

Adding Foreign Key

```
ALTER TABLE employees ADD CONSTRAINT fk_employees_department
    FOREIGN KEY (department_id)
REFERENCES departments(department_id);
```

Dropping a Table

```
DROP TABLE IF EXISTS employees;
```

Conclusion

This lecture covered:

- Advanced querying techniques.
- Aggregations and filtering.
- Complex joins and subqueries.
- Data manipulation and constraints.