Lecture 10: Stored Procedures - In-Class Task (15 minutes)

Database Setup

Use the **dvdrental** database for all exercises.

Task 1: Basic Function Creation (5 minutes)

Exercise 1.1: Simple Calculation Function

Create a stored function called `calculate_discount` that:

- Accepts two parameters: `original_price` (NUMERIC) and `discount_percent` (NUMERIC)
- Returns the discounted price as NUMERIC
- Formula: original_price - (original_price * discount_percent / 100)

Test your function:

sql
```
SELECT calculate_discount(100, 15);  -- Should return 85
SELECT calculate_discount(250.50, 20);  -- Should return 200.40
```

Task 2: Working with OUT Parameters (4 minutes)

Exercise 2.1: Film Statistics Function

Create a function called `film_stats` that:

- Accepts one IN parameter: `p_rating` (VARCHAR) - the film rating (e.g., 'PG', 'R', 'G')
- Has two OUT parameters:
  - `total_films` (INTEGER) - count of films with that rating
  - `avg_rental_rate` (NUMERIC) - average rental rate for that rating

Hint: Use the `film` table and aggregate functions.

Test your function:

sql
```
SELECT * FROM film_stats('PG');
SELECT * FROM film_stats('R');
```

Task 3: Function Returning a Table (4 minutes)

Exercise 3.1: Customer Rental History

Create a function called `get_customer_rentals` that:

- Accepts one parameter: `p_customer_id` (INTEGER)
- Returns a table with columns:
  - `rental_date` (DATE)
  - `film_title` (VARCHAR)
  - `return_date` (DATE)
- Retrieves rental history for the specified customer

Hint: You'll need to join `rental`, `inventory`, and `film` tables.

Test your function:

sql
```
SELECT * FROM get_customer_rentals(1);
SELECT * FROM get_customer_rentals(5) LIMIT 5;
```

Task 4: Challenge - Function Overloading (2 minutes)

Exercise 4.1: Overloaded Film Search

Create TWO versions of a function called `search_films`:

Version 1:

- Parameter: `p_title_pattern` (VARCHAR)
- Returns: TABLE with `title` and `release_year`
- Searches films by title pattern

Version 2:

- Parameters: `p_title_pattern` (VARCHAR), `p_rating` (VARCHAR)
- Returns: TABLE with `title`, `release_year`, and `rating`
- Searches films by both title pattern AND rating

Test both versions:

sql
```
SELECT * FROM search_films('A%');
SELECT * FROM search_films('A%', 'PG');
```