Table of Contents

We can write routines that interact with table data. This is a bit beyond the topics for this class so this is option. But I do want you to know that a function can return a single value based on the data in a table.

1. Interacting with Tables (Optional)

1.1. Using a Query to Place a Value into a Variable

The SQL statement in the following routine will count the number of rows in the table. There is a new clause (`INTO variable`) that places the Count(*) value into a local MySQL variable. The Select statement can refer to MySQL variables in both the `INTO` clause and the `WHERE` clause.

These are not robust functions- they simply are designed to show you that we can use MySQL to retrieve data from a table.

Demo 01: Getting a single value from the query to place into a local MySQL variable and return it.

```
Drop function if exists a_emp.DeptEmployeeCount #

Create function a_emp.DeptEmployeeCount  (
   p_dept_id int)
   returns int
begin
    declare  v_row_count   int ;
    select   count(*)
    into     v_row_count
    from     a_emp.employees
    where    Dept_id = p_dept_id;

    return v_row_count;
end;
#
```

Demo 02: Testing this

```
Select a_emp.DeptEmployeeCount (80) #
+------------------------+
| DeptEmployeeCount (80) |
+------------------------+
|                      3 |
+------------------------+

Select a_emp.DeptEmployeeCount (1234) #
+-------------------------+
| DeptEmployeeCount (1234) |
+-------------------------+
|                        0 |
+-------------------------+
```

Demo 03: This is a function that uses a join between two tables in the select query.

```
Drop function if exists a_emp.empjobtitle#

create function a_emp.empjobtitle
   (in_emp_id int)
   returns varchar(100)
begin
   declare  v_job_title  varchar(35);
   declare  v_message    varchar(100);

   select   job_title
     into   v_job_title
     from   a_emp.employees
     join a_emp.jobs using (job_id)
     where  emp_id = in_emp_id;
     set  v_message := concat('employee ' , in_emp_id ,' has job ' , v_job_title);
   return v_message;
end;
#
```

Demo 04: Test this with a value that matches an employee

```
select  a_emp.EmpJobTitle(103)#
+--------------------------+
| EmpJobTitle(103)         |
+--------------------------+
| employee 103 has job DBA |
+--------------------------+
```

Demo 05: Test this with a value that does not match an employee

```
select  a_emp.EmpJobTitle(63) #
+-----------------+
| EmpJobTitle(63) |
+-----------------+
| NULL            |
+-----------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1329): No data - zero rows fetched, selected, or processed
```

In this case the select query that gets a value for the job_title from the table cannot find a matching row and returns a Null. That null is assigned to the variable v_job_title and causes the concatenated string to be null.

Demo 06: The error message is coming from the function.

```
Select  coalesce(a_emp.EmpJobTitle(63), 'bad value for Employee id') as Result#
+--------------------------+
| Result                   |
+--------------------------+
| bad value for Employee id |
+--------------------------+
1 row in set, 1 warning (0.00 sec)

Warning (Code 1329): No data - zero rows fetched, selected, or processed
```