SQL – Subqueries

Dr. Andrea Villanes

Questions

Q16, Q17

Subqueries

(also known as inner queries or nested queries)

What is a Subquery?

A subquery:

- is a query within another SQL query
- returns values to be used:
  - SQLite: You can use a subquery in the SELECT, WHERE, or JOIN clause.
  - SAS: You can use a subquery in the WHERE or HAVING clause
- must return only a single column
- can return multiple values or a single value.

Example of a Subquery

outer query

subquery

```
SELECT trackid, name, albumid
FROM tracks
WHERE albumid = (SELECT albumid
                 FROM albums
                 WHERE title =
          'Let There Be Rock');
```

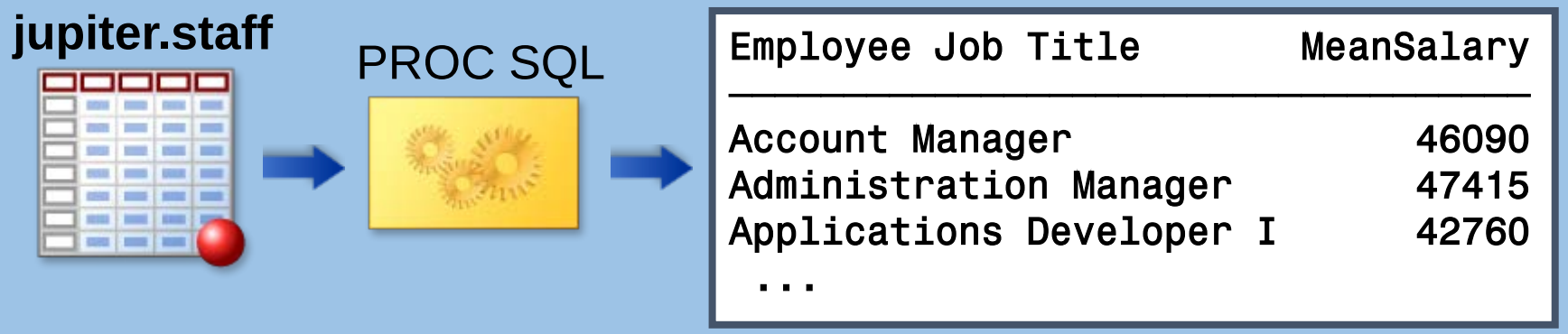
Two Types of Subqueries

There are two types of subqueries:

- A *noncorrelated subquery* is a self-contained query. It executes independently of the outer query.
- A *correlated subquery* requires a value or values to be passed to it by the outer (main) query before it can be successfully resolved.

Non-correlated query example

Generate a report that displays **Job_Title** for job groups with an average salary greater than the average salary of the company as a whole.



Step 1: Subquery

1. Calculate the company's average salary

```
proc sql;
select avg(Salary) as CompanyMeanSalary
   from jupiter.staff;
quit;
```

Step 2: Outer Query

2. Determine the job titles whose average salary exceeds the company's average salary.

```
proc sql;
select Job_Title,
       avg(Salary) as MeanSalary
   from jupiter.staff
   group by Job_Title
   having MeanSalary>38041.51;
quit;
```

Step 3: piece it together

The subquery is resolved before the outer query can be resolved

```
proc sql;
select Job_Title,
       avg(Salary) as MeanSalary
   from jupiter.staff
   group by Job_Title
   having avg(Salary) >
      (select avg(Salary)
          from jupiter.staff);
quit;
```

Evaluate the subquery first.

Correlated query example

A *correlated subquery* requires a value or values to be passed to it by the outer (main) query before it can be successfully resolved.

```
proc sql;
select Employee_ID, avg(Salary) as MeanSalary
   from jupiter.employee_addresses
   where 'AU'=
      (select Country
          from work.supervisors
          where employee_addresses.Employee_ID=
                supervisors.Employee_ID)
group by 1;
quit;
```

This query is not stand-alone.
It needs additional information from the main query.

Returning multiple rows from the subquery

A subquery can return multiple values or a single value.

However, subqueries that return more than one row can only be used with multiple value operators, such as the IN operator.

```
select Employee_Name, City, Country
from employee_addresses
where Employee_ID IN
      (select Employee_ID
          from employee_payroll
          where Birth_Month=2)
order by 1;
```

The NOT IN operator displays a record if the condition(s) is NOT TRUE.

In-Line Views

(also known as subquery)

What is an In-Line View?

An *in-line view* is a query expression (SELECT statement) that resides in a FROM clause:

- It acts as a virtual table, used in place of a physical table in a query.
- An in-line view can return more than just one column

```
SELECT sub.*
FROM (SELECT date, location, resolution
          FROM tutorial.sf_crime_incidents
          WHERE day_of_week = 'Friday')as
          sub
WHERE sub.resolution = 'NONE'
```

In-Line View Exercise

List all active Sales Department employees who have annual salaries significantly lower (less than 95%) than the average salary for everyone with the same job title.

Employee_Name	Employee Job Title	Employee Annual Salary	Job_Avg
Ould, Tulsidas	Sales Rep. I	22,710	26,576
Polky, Asishana	Sales Rep. I	25,110	26,576
Voron, Tachaun	Sales Rep. I	25,125	26,576

Step 1

Calculate the average salaries for active employees in the Sales Department, grouped by job title.

```
select Job_Title,
       avg(Salary) as Job_Avg
   from jupiter.employee_payroll as p,
        jupiter.employee_organization as o
   where p.Employee_ID=o.Employee_ID
         and Employee_Term_Date is missing
         and Department="Sales"
   group by Job_Title;
```

Step 2

Match each employee to a job title group and compare the employee's salary to the group's average to determine whether it is less than 95% of the group average.

```
select Employee_Name, emp.Job_Title,
       Salary, Job_Avg
   from (select Job_Title,
                avg(Salary) as Job_Avg format=comma7.
           from jupiter.employee_payroll as p,
                jupiter.employee_organization as o
           where p.Employee_ID=o.Employee_ID
                 and Employee_Term_Date is missing
                 and Department="Sales"
           group by Job_Title) as job,
        jupiter.salesstaff as emp
   where emp.Job_Title=job.Job_Title
         and Salary<Job_Avg*.95
         and Emp_Term_Date is missing
   order by Job_Title, Employee_Name;
```