PCRS

Prep 5 - Part 1 of 1

Null Values

When we studied relational algebra, we assumed every attribute in every tuple had a value. But in a real database, sometimes information is missing. First, read the section of our Readings called **Null Values in SQL**. (All readings are available via the Lectures page on Querucs.) Then answer the questions below.

Not high salary

Suppose this is the content of the Employee table:

```
 eid |      name       | salary | dept
-----+-----------------+--------+------
   2 | Marissa Mayer   |     82 |   55
   3 | Sheryl Sandberg |     17 |   55
   4 | Larry Ellison   |     55 |   33
   5 | Tim Cook        |     48 |   55
   6 | Mark Zuckerberg |     95 |   22
   7 | Jeff Weiner     |     28 |   33
   8 | Larry Page      |    145 |   33
  25 | Julia           |        |   55
   1 | Bill Gates      |    100 |   55
(9 rows)
```

What is the output of this query?

```
select * from employee
where not salary > 100;
```

☐
```
 eid |      name       | salary | dept
-----+-----------------+--------+------
   2 | Marissa Mayer   |     82 |   55
   3 | Sheryl Sandberg |     17 |   55
   4 | Larry Ellison   |     55 |   33
   5 | Tim Cook        |     48 |   55
   6 | Mark Zuckerberg |     95 |   22
   7 | Jeff Weiner     |     28 |   33
```

☑
```
 eid |      name       | salary | dept
-----+-----------------+--------+------
   2 | Marissa Mayer   |     82 |   55
```

```
     3 | Sheryl Sandberg |     17 |   55
     4 | Larry Ellison   |     55 |   33
     5 | Tim Cook        |     48 |   55
     6 | Mark Zuckerberg |     95 |   22
     7 | Jeff Weiner     |     28 |   33
     1 | Bill Gates      |    100 |   55
```

☐

```
  eid |      name       | salary | dept
-----+-----------------+--------+------
     2 | Marissa Mayer   |     82 |   55
     3 | Sheryl Sandberg |     17 |   55
     4 | Larry Ellison   |     55 |   33
     5 | Tim Cook        |     48 |   55
     6 | Mark Zuckerberg |     95 |   22
     7 | Jeff Weiner     |     28 |   33
    25 | Julia           |        |   55
     1 | Bill Gates      |    100 |   55
```

☐

```
  eid |      name       | salary | dept
-----+-----------------+--------+------
     2 | Marissa Mayer   |     82 |   55
     3 | Sheryl Sandberg |     17 |   55
     4 | Larry Ellison   |     55 |   33
     5 | Tim Cook        |     48 |   55
     6 | Mark Zuckerberg |     95 |   22
     7 | Jeff Weiner     |     28 |   33
    25 | Julia           |        |   55
```

History Submit

✔ Your solution is complete.

Comparing counts ✔

This SQL query runs without error:

```
SELECT count(stuff), count(*), count(distinct stuff)
FROM StuffAndNonsense;
```

(We learned about using distinct inside an aggregation last week.)

Which of the following statements are true?

- [x] **count(stuff) <= count(*)**
- [ ] **count(*) <= count(stuff)**
- [ ] **count(*) <= count(distinct stuff)**
- [x] **count(distinct stuff) <= count(*)**

- [ ] **count(stuff) <= count(distinct stuff)**
- [x] **count(distinct stuff) <= count(stuff)**

History Submit

✔ Your solution is complete.

Different Kinds of Joins

To prepare for the next questions, read the section of our Readings called **Joins in SQL**. All readings are available via the Lectures page on Querucs.

For the following queries, follow these rules:

- **Duplicates**: Do not remove any duplicates.
- **Technique**: Use the technique specified. You could solve these in other ways, for instance with GROUP BY and aggregation, but the purpose of this prep is to practise various kinds of joins. If you don't use the specified technique, your answer may be graded as incorrect due to having differences in what duplicates are or are not present.
- **Order**: The order in which your tuples are returned does not matter, unless explicitly specified.

We begin with inner joins, for which there is no padding with NULL values.

Double Manager ✔

sales(eid, day, amount)
employee(eid, name, salary, dept)
manages(manager, junior)
department(did, name, division)
employee[dept] ⊆ department[did]
manages[manager] ⊆ employee[eid]
manages[junior] ⊆ employee[eid]
sales[eid] ⊆ employee[eid]

Report the eid of every employee who manages at least two different people. Use the usual technique of table renaming and self-join, except use the keywords `CROSS JOIN` to take the Cartesian product of the two tables:

```
SELECT * FROM Table1 CROSS JOIN Table2
```

`Table1 CROSS JOIN Table2` is completely equivalent to `Table1, Table2` - the latter is simply a convenient shorthand for us.
Your output table should have a single column named 'manager'.

```
create table t as
    select * from employee cross join manages
    where eid = manager;

--the returned column name is manager
select t1.manager from t t1 cross join t t2
    where t1.eid = t2.eid and t1.junior !=
t2.junior;
```

History Submit

✔ Your submission is correct!

✔ Test Case Passed

Expected Result is Hidden

Actual

manager
1
1
3
3

Rich Sales ✔

sales(eid, day, amount)
employee(eid, name, salary, dept)
manages(manager, junior)
department(did, name, division)
employee[dept] ⊆ department[did]
manages[manager] ⊆ employee[eid]
manages[junior] ⊆ employee[eid]
sales[eid] ⊆ employee[eid]

Report the name and sales of everyone with a salary over 50, using the keywords `NATURAL JOIN` (plus a `WHERE` clause, of course):

```
SELECT * FROM Table1 NATURAL JOIN Table2
```

Your output table should have three columns: 'name', 'day', and 'amount', in that order.

```
1 select name, day, amount
2 from employee natural join sales
3 where salary > 50;
```

History Submit

✔ Your submission is correct!

✔ Test Case Passed

Expected Result is Hidden

Actual

name	day	amount
Larry Ellison	2011-11-01	155
Larry Ellison	2011-11-02	19
Larry Ellison	2011-11-03	10
Larry Ellison	2012-11-04	100
Larry Ellison	2012-12-01	25

Larry Ellison	2012-12-04	29
Larry Ellison	2009-12-05	20
Larry Ellison	2009-12-07	120
Mark Zuckerberg	2011-11-06	60
Mark Zuckerberg	2012-11-07	18
Mark Zuckerberg	2009-12-04	2
Mark Zuckerberg	2009-12-05	95
Larry Page	2012-11-01	98
Larry Page	2012-11-02	92
Larry Page	2009-12-31	91
Larry Page	2009-12-21	98

Department Salaries I

sales(eid, day, amount)
employee(eid, name, salary, dept)
manages(manager, junior)
department(did, name, division)
employee[dept] ⊆ department[did]
manages[manager] ⊆ employee[eid]
manages[junior] ⊆ employee[eid]
sales[eid] ⊆ employee[eid]

Report the salaries of everyone in every department, using the keywords `INNER JOIN ... ON` (equivalent to theta join):

```
SELECT * FROM Table1 INNER JOIN Table2 ON Condition
```

Your output table should have three columns: 'name' (name of employee), 'department' (name of employee's department) and 'salary' (salary of an employee in that department).

```
1 select employee.name as name, department.name as
  department, salary from
```

```
2  employee inner join department
3  on dept = did;
```

History Submit

✔ Your submission is correct!

✔ Test Case Passed

Expected Result is Hidden

Actual

name	department	salary
Bill Gates	Widgets	59
Marissa Mayer	Widgets	82
Sheryl Sandberg	Widgets	17
Larry Ellison	Electronics	55
Tim Cook	Widgets	48
Mark Zuckerberg	Housewares	95
Jeff Weiner	Electronics	28
Larry Page	Electronics	145

Now let's think about outer joins. First, we'll do some tracing questions to confirm that you understand what they do.

Nulls and joins 1 ✔

Suppose table R contains this:

```
 a | b
---+---
 1 | 2
 8 | 7
 5 |
   | 6
```

and table S contains this:

```
 a | b
---+---
 3 | 4
 8 | 7
 5 |
   | 6
```

What is the output of this query?

```
SELECT *
FROM R Natural JOIN S;
```

- [ ]
```
 a | b
---+---
 1 | 2
 5 |
 8 | 7
   | 6
```

- [x]
```
a | b
---+---
 8 | 7
```

- [ ]
```
 a | b
---+---
 3 | 4
 5 |
 8 | 7
   | 6
```

- [ ]
```
 a | b
---+---
 1 | 2
 5 |
 3 | 4
 5 |
 8 | 7
   | 6
   | 6
```

- [ ] **None of the above**

History Submit

✔ Your solution is complete.

Nulls and joins 2

Suppose table R contains this:

```
 a | b
---+---
 1 | 2
 8 | 7
 5 |
   | 6
```

and table S contains this:

```
 a | b
---+---
 3 | 4
 8 | 7
 5 |
   | 6
```

What is the output of this query?

```
SELECT *
FROM R NATURAL LEFT JOIN S;
```

- [x]
```
 a | b
---+---
 1 | 2
 5 |
 8 | 7
   | 6
```

- [ ]
```
a | b
---+---
 8 | 7
```

- [ ]
```
 a | b
---+---
 3 | 4
 5 |
 8 | 7
   | 6
```

- [ ]
```
 a | b
---+---
 1 | 2
 5 |
 3 | 4
 5 |
 8 | 7
   | 6
   | 6
```

☐ **None of the above**

History Submit

✔ Your solution is complete.

Nulls and joins 3 ✔

Suppose table R contains this:

```
 a | b
---+---
 1 | 2
 8 | 7
 5 |
   | 6
```

and table S contains this:

```
 a | b
---+---
 3 | 4
 8 | 7
 5 |
   | 6
```

What is the output of this query?

```
SELECT *
FROM R NATURAL  RIGHTJOIN S;
```

☐
```
 a | b
---+---
 1 | 2
 5 |
 8 | 7
   | 6
```

☐
```
a | b
---+---
 8 | 7
```

☑
```
 a | b
---+---
 3 | 4
 5 |
```

```
   8 | 7
     | 6
```

☐
```
  a | b
 ---+---
  1 | 2
  5 |
  3 | 4
  5 |
  8 | 7
    | 6
    | 6
```

☐ **None of the above**

History Submit

✔ Your solution is complete.

Nulls and joins 4 ✔

Suppose table R contains this:

```
 a | b
---+---
 1 | 2
 8 | 7
 5 |
   | 6
```

and table S contains this:

```
 a | b
---+---
 3 | 4
 8 | 7
 5 |
   | 6
```

What is the output of this query?

```
SELECT *
FROM R NATURAL FULL JOIN S;
```

☐
```
  a | b
 ---+---
  1 | 2
```

```
 5 |
 8 | 7
   | 6
```

- [ ]
```
a | b
---+---
 8 | 7
```

- [ ]
```
 a | b
---+---
 3 | 4
 5 |
 8 | 7
   | 6
```

- [x]
```
 a | b
---+---
 1 | 2
 5 |
 3 | 4
 5 |
 8 | 7
   | 6
   | 6
```

- [ ] **None of the above**

History Submit

✔ Your solution is complete.

Department Salaries II ✔

sales(eid, day, amount)
employee(eid, name, salary, dept)
manages(manager, junior)
department(did, name, division)
employee[dept] ⊆ department[did]
manages[manager] ⊆ employee[eid]
manages[junior] ⊆ employee[eid]
sales[eid] ⊆ employee[eid]

And now let's write some queries that require outer joins.

As in a previous question, report the salaries of everyone in every department except this time, each employee with no department should

appear in a single tuple in the output, with their department being NULL (this is displayed in PCRS as simply "`null`")

As before, your output table should have three columns: 'name' (name of employee), 'department' (name of employee's department) and 'salary' (salary of an employee in that department).

```
1 select employee.name as name, department.name as department, salary from
2 employee left join department
3 on dept = did;
```

History Submit

✔ Your submission is correct!

✔ Test Case Passed

Expected Result is Hidden

Actual

name	department	salary
Bill Gates	Widgets	59
Marissa Mayer	Widgets	82
Sheryl Sandberg	Widgets	17
Larry Ellison	Electronics	55
Susan Wojcicki	null	300
Tim Cook	Widgets	48
Mark Zuckerberg	Housewares	95
Jeff Weiner	Electronics	28
Larry Page	Electronics	145
Dick Costolo	null	108

Department Salaries III ✔

sales(eid, day, amount)
employee(eid, name, salary, dept)
manages(manager, junior)
department(did, name, division)
employee[dept] ⊆ department[did]
manages[manager] ⊆ employee[eid]
manages[junior] ⊆ employee[eid]
sales[eid] ⊆ employee[eid]

Report the salaries of every person by department, as in the earlier questions, except that:

- Any employee belonging to no department is reported in the output with a NULL department.
- Any department with no employees is reported in the output with a NULL salary and employee name.

As before, your output table should have three columns, 'name', 'department', and 'salary', where any of these may be NULL in an output tuple.

```
1 select employee.name as name, department.name as
  department, salary from
2 employee full join department
3 on dept = did;
```

History Submit

✔ Your submission is correct!

✔ Test Case Passed

Expected Result is Hidden

Actual

name	department	salary
Bill Gates	Widgets	59
Marissa Mayer	Widgets	82
Sheryl Sandberg	Widgets	17
Larry Ellison	Electronics	55

Susan Wojcicki	null	300
Tim Cook	Widgets	48
Mark Zuckerberg	Housewares	95
Jeff Weiner	Electronics	28
Larry Page	Electronics	145
Dick Costolo	null	108
null	Videos	null
null	Minions	null

Using PostgreSQL on the CS Teaching Lab Machines

Now it's time to begin using PostgreSQL to run your queries. Here's what to do:

1. Learn how to use PostgreSQL on the CS Teaching Labs by completing Parts 1 through 4 of the PostgreSQL tutorial on Quercus (see the page labelled PostgreSQL Instructions).
2. Re-aquaint yourself with the schema for our database of countries and languages from last week by trying these psql commands:
    - "\d" to describe the tables that are available.
    - "\d country" to describe the table called "country".
    - "\?" to get a look at some of the other commands available inside psql. You can use "q" to quit the list.

    You don't have to understand all of what you see here, but remember that these psql commands exist. They will be helpful later.
3. Run SQL queries **on the CS Teaching labs** to find out the answer to the final questions below. Give each answer (a number, not the query that generated it) below.

English countries

In how many countries is English spoken?

60

History Submit

✔ Your solution is complete.

Mozambique languages ✔

How many languages are spoken in Mozambique?

10

History Submit

✔ Your solution is complete.