← Back **Week 6 Honors Quiz**
Graded Quiz • 10 min

Due Jun 18, 11:59 PM +07

Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 75% or higher

Go to next item

1. In 2019, a growing hotel chain sent out customer satisfaction surveys to all guests after their stay and recorded the responses in a table named **survey2019**. The survey includes a field for the location of the recent stay, but some customers neglect to include it in their responses, so sometimes the **location** field is **NULL**. (Note that some locations may close, and several new locations were opened in 2019.) The chain also has a table that holds average responses from previous years, for different demographics. Consider this query, which gets information from both tables for comparison.

1 / 1 point

```
SELECT this.location, AVG(this.room_service_rating) AS avg_2019,
    AVG(s.room_service_rating) AS avg_previous
  FROM survey2019 this
    JOIN survey_summary s ON (this.location <=> s.location)
  GROUP BY this.location;
```

Which best describes the result of this query?

- All rows from both tables will be used to calculate averages; there will be no rows with **NULL** in **this.location**
- [✓ Only rows with a match will be used to calculate averages; there will be one row with **NULL** in **this.location**, and it will have values for both **avg_2019** and **avg_previous**]
- All rows from both tables will be used to calculate averages; there will be one row with **NULL** in **this.location**, and it will have values for both **avg_2019** and **avg_previous**
- Only rows with a match will be used to calculate averages; there will be two rows with **NULL** in **this.location**, one with **NULL** for **avg_previous** (but a value for **avg_2019**) and one with **NULL** for **avg_2019** (but a value for **avg_previous**)
- Only rows with a match will be used to calculate averages; there will be no rows with **NULL** in **this.location**
- All rows from both tables will be used to calculate averages; there will be two rows with **NULL** in **this.location**, one with **NULL** for **avg_previous** (but a value for **avg_2019**) and one with **NULL** for **avg_2019** (but a value for **avg_previous**)

✓ **Correct**
Correct. Only rows with a match will be used, and the **NULL**-safe operator (**<=>**) will match all rows with **NULL** for **this.location** to those with **NULL** for **s.location**.

2. Following are two tables, and a query to join them.

1 / 1 point

uncles

name	age
John	38
Harry	63
Stiel	44

aunts

name	age
Ann	24
Mildred	64
Kayla	54

```
SELECT aunts.name AS aunt, uncles.name AS uncle
  FROM aunts JOIN uncles
    ON aunts.age > uncles.age;
```

How many rows will this query return?

5

✓ **Correct**

Correct. This returns all **aunt** and **uncle** pairings for which the aunt is older than the uncle. Ann is younger than all the uncles, so there will be no rows in the result set with **Ann** for **aunts.name**. Mildred is older than all three uncles, so there will be will be three rows with **Mildred** for **aunts.name**. Finally, Kayla is older than John and Stiel, so there will be two rows with **Kayla** for **aunts.name**.

3. How many rows will result if you cross join a table that has 10 rows with a table that has 60 rows?

1 / 1 point

600

✓ **Correct**

Correct. Each of the 10 rows in the first table will produce 60 rows when joined with the second table. That's 10 * 60 or 600 rows.

4. Following are the schema for the **fly.flights** and **fly.planes** tables on the VM.

1 / 1 point

fly.flights

name	type
year	smallint

month	tinyint
day	tinyint
dep_time	smallint
sched_dep_time	smallint
dep_delay	smallint
arr_time	smallint
sched_arr_time	smallint
arr_delay	smallint
carrier	string
flight	smallint
tailnum	string
origin	string
dest	string
air_time	smallint
distance	smallint

fly.planes

name	**type**
tailnum	string
year	int
type	string
manufacturer	string
model	string
engines	int
seats	int
engine	string

Which of the following are valid semi-joins? Check all that apply.

☐ SELECT origin, arr_delay, engines

FROM fly.flights f LEFT SEMI JOIN fly.planes p

ON f.tailnum = p.tailnum;

☐ SELECT origin, type, flight, arr_delay

FROM fly.flights f LEFT SEMI JOIN fly.planes p

ON f.tailnum = p.tailnum;

[✓ SELECT origin, flight, arr_delay

FROM fly.flights f LEFT SEMI JOIN fly.planes p

ON f.tailnum = p.tailnum WHERE arr_delay > 60;]

Correct
Correct. The join condition is the only place where a column from the right table (**planes**) is used; this is allowed.

☑ SELECT origin, flight, arr_delay

```
 FROM fly.flights f LEFT SEMI JOIN fly.planes p
   ON f.tailnum = p.tailnum AND engines > 1;
```

Correct
Correct. The only place where a column from the right table (**planes**) is used is in the join condition, which is allowed.