周测 • 30 MIN

Week 6 Core Quiz

提交您的作业
截止时间 Feb 2, 11:59 PM PST 长期工作 328 hours

再试

收到成绩
通过条件 80% 或更高

成绩
80%

查看成绩

您可以多次提交您的最高分数

恭喜！您通过了！
通过条件 80% 或更高

返回学习

成绩
80%

Week 6 Core Quiz

最新提交作业的评分
80%

1. The questions in this quiz intentionally use tables that are not in the VM. You should be able to answer the questions without running any queries.

1/1 分

- (selected) I acknowledge that I do not need to run any queries for the following questions. I will not be able to run them because the tables do not exist on the VM.

Correct
Proceed!

2. Which of these queries produces the same result set as the following query?

1/1 分

SELECT * FROM table1

UNION

SELECT * FROM table2;

- SELECT * FROM table1 UNION ALL SELECT * FROM table2
- (selected) SELECT * FROM table1 UNION DISTINCT SELECT * FROM table2

Correct
Correct. The default for **UNION** is identical to **UNION DISTINCT**.

3. Choose the best query to run in Impala to return the distinct union of the columns zip_plus_4 (type STRING, has values like '94305-0507') in the california_emp table and zip (type INT, has values like 94110) in the california_offices table.

1/1 分

- SELECT CAST(zip_plus_4 AS INT) AS zipcode FROM california_emp UNION DISTINCT SELECT zip AS zipcode FROM california_offices;
- SELECT zip_plus_4 AS zipcode FROM california_emp UNION DISTINCT SELECT zip AS zipcode FROM california_offices;
- (selected) SELECT zip_plus_4 AS zipcode FROM california_emp UNION DISTINCT SELECT CAST(zip AS STRING) AS zipcode FROM california_offices;
- SELECT zip_plus_4 FROM california_emp UNION DISTINCT SELECT CAST(zip AS STRING) FROM california_offices;
- SELECT zip_plus_4 FROM california_emp UNION DISTINCT SELECT zip FROM california_offices;

Correct
Correct. This query uses explicit casting and column aliases to give the columns the same name (**zipcode**) and data type (**STRING**). Converting the integer values in the **zip** column to string values avoids any values being returned as **NULL**.

4. The zip column (type INT) in the california_offices table has values from 90001 to 99899. The zip column (also type INT) in the oregon_offices table has values from 97000 to 97430. Which value is guaranteed to be in the top row of the result set when you run the following query with Impala?

1/1 分

SELECT zip FROM california_offices

UNION ALL

SELECT zip FROM oregon_offices

ORDER BY country DESC;

- 90001
- 99899
- 97000
- 97430
- (selected) No particular value is guaranteed to be in the top row

Correct
Correct. With Impala, the **ORDER BY** clause at the end of this query arranges the rows from the **oregon_offices** table in descending order by **zip**, but it has no effect on the arrangement of rows from the **california_offices** table. Furthermore, when the **UNION** operator combines the rows from the two tables, there is no guarantee that it will preserve row ordering. Therefore, there is no way to know for sure which value will be in the first row of this result set.

5. The california_offices table has 65 rows, and the oregon_offices table has 5 rows. How many rows does the following query return when you run it with Impala?

1/1 分

SELECT zip FROM california_offices

UNION ALL

SELECT zip FROM oregon_offices

LIMIT 2;

67

Correct
Correct. With Impala, the **LIMIT 2** at the end of this query limits the number of rows returned from the **oregon_offices** table (the table on the right side of the **UNION ALL**). It has no effect on the number of rows returned from the **california_offices** table (the table on the left side of the **UNION ALL**). The **california_offices** table has 65 rows, so this query returns 65 + 2 = 67 rows.

6. The california_offices and california_emp tables each have a column named office_id. All other columns have unique names between the two tables. Which of the following are valid join queries that Impala would run successfully on the VM, if these tables existed on the VM? Check all that apply.

0/1 分

- [ ] SELECT name, office_id, city, salary
  FROM california_offices AS o
  JOIN california_emp AS e ON o.office_id = e.office_id;
- [ ] SELECT name, california_emp.office_id, city, salary
  FROM california_emp
  JOIN california_offices
  ON california_emp.office_id = california_offices.office_id;
- [x] SELECT name, e.office_id AS office_id, city, salary
  FROM california_offices
  JOIN california_emp ON o.office_id = e.office_id;

  This should not be selected
  Incorrect. You might want to review the "HQL Join Syntax" video.
- [x] SELECT name, e.office_id AS office_id, city, salary
  FROM california_offices AS o
  JOIN california_emp AS e ON o.office_id = e.office_id;

Correct
Correct. Any ambiguous column references are disambiguated by prepending the table alias.

7. Which of the following are valid join queries for Impala? Check all that apply.

0/1 分

- [ ] SELECT o.office_id as office, AVG(salary) AS avg_salary
  FROM california_emp e
  JOIN california_offices o ON o.office_id = e.office_id
  GROUP BY office
  ORDER BY avg_salary;
- [ ] SELECT o.office_id AS office, COUNT(*) AS number_of_employees
  FROM california_emp e
  JOIN california_offices o ON o.office_id = e.office_id
  GROUP BY office;
- [x] SELECT name, o.office_id AS office, city
  FROM california_emp e
  JOIN california_offices o ON o.office_id = e.office_id
  ORDER BY office DESC, name DESC;

Correct
Correct. It is valid to use this column alias office instead of the column reference **o.office_id** in the **ORDER BY** clause.

- [ ] SELECT name, o.office_id AS office
  FROM california_emp e
  JOIN california_offices o ON o.office_id = e.office_id
  WHERE office = 'CA0201';

You didn't select all the correct answers

8. The california_emp table includes one row with name='Sandy Tilbrook', with office_id='CA0201'. There is no row in california_offices with office_id='CA0201'. However, there is a office_id='CA0210' in california_offices with city='Redding', but no rows in california_emp have office_id='CA0210'. (There are no other rows with city='Redding'.) Choose the response that best describes how these rows will be included in the result set of this query:

1/1 分

SELECT name, city, salary

FROM california_emp e

INNER JOIN california_offices o ON e.office_id = o.office_id;

- A row with **name='Sandy Tilbrook'** will be included, and a row with **city='Redding'** will be included
- (selected) No row with **name='Sandy Tilbrook'** will be included, and no row with **city='Redding'** will be included
- A row with **city='Redding'** will be included, but no row with **name='Sandy Tilbrook'** will be included
- A row with **name='Sandy Tilbrook'** will be included, but no row with **city='Redding'** will be included

Correct
Correct. In an inner join, only rows that have a match will be included.

9. Which FROM clauses could you use to return data about all the employees in california_emp, even the remote workers who are not assigned to an office (office_id='0211') or those erroneously assigned to a non-existent office? Select all that apply.

1/1 分

- [x] FROM *california_emp* e LEFT OUTER JOIN *california_offices* o ON e.office_id=o.office_id
  Correct
  Correct. The left table in this case is **california_emp**, so the left outer join includes all rows from **california_emp**, even if there is no match in **california_offices**.
- [ ] FROM *california_offices* o LEFT OUTER JOIN *california_emp* e ON e.office_id=o.office_id
- [x] FROM *california_offices* o RIGHT OUTER JOIN *california_emp* e ON e.office_id=o.office_id
  Correct
  Correct. The right table in this case is **california_emp**, so the right outer join includes all rows from **california_emp**, even if there is no match in **california_offices**.
- [ ] FROM *california_emp* e RIGHT OUTER JOIN *california_offices* o ON e.office_id=o.office_id

10. Which of the following queries returns only the employees whose office IDs do not match any office IDs found in the offices table?

1/1 分

- SELECT emp1_id, name
  FROM *california_emp* e
  LEFT OUTER JOIN *california_offices* o ON e.office_id = o.office_id
  WHERE office_id IS NULL;
- (selected) SELECT emp1_id, name
  FROM *california_emp* e
  LEFT OUTER JOIN *california_offices* o ON e.office_id = o.office_id
  WHERE o.office_id IS NULL;
- SELECT emp1_id, name
  FROM *california_emp* e
  LEFT OUTER JOIN *california_offices* o ON e.office_id = o.office_id
  WHERE e.office_id IS NULL;
- SELECT emp1_id, name
  FROM *california_offices* o
  LEFT OUTER JOIN *california_emp* e ON e.office_id = o.office_id
  WHERE e.office_id IS NULL;
- SELECT emp1_id, name
  FROM *california_offices* o
  LEFT OUTER JOIN *california_emp* e ON e.office_id = o.office_id
  WHERE o.office_id IS NULL;

Correct
Correct. The left outer join includes all rows from the *california_emp* table, leaving **e.office_id NULL** if there is no match; then the **WHERE** clause returns only those cases for which that column is **NULL**. Any column with a match would have a non-**NULL** value for **e.office_id**.