← Back **Week 6 Core Quiz**
Graded Quiz • 30 min

Due Jun 18, 11:59 PM +07

Congratulations! You passed!

Grade received 100% **Latest Submission Grade** 100% **To pass** 80% or higher

Go to next item

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 point

● 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 point

SELECT * FROM table1

UNION

SELECT * FROM table2;

● SELECT * FROM table1 UNION DISTINCT SELECT * FROM table2

○ SELECT * FROM table1 UNION ALL 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 **'94306-0001'**) in the **california_emp** table and **zip** (type **INT**, has values like **94105**) in the **california_offices** table.

1 / 1 point

○ 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 FROM california_emp UNION DISTINCT SELECT zip FROM california_offices;

○ SELECT zip_plus_4 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 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;

> ✓ **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 **95899**. The **zip** column (also type **INT**) in the **oregon_offices** table has values from **97030** to **97440**. Which value is guaranteed to be in the top row of the result set when you run the following query with Impala?

1 / 1 point

SELECT zip FROM california_offices

UNION ALL

SELECT zip FROM oregon_offices

ORDER BY country DESC;

○ 90001

○ 95899

○ 97030

○ 97440

[✓ 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 point

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

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.

1 / 1 point

[✓]
```
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.

[✓]
```
SELECT name, california_emp.office_id, city, salary
  FROM california_emp
  JOIN california_offices
    ON california_emp.office_id = california_offices.office_id;
```

Correct
Correct. This query would be improved by using table aliases instead of the lengthy table names, but it is valid.

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

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

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

1 / 1 point

[ ]
```
SELECT name, o.office_id AS office
  FROM california_emp e
  JOIN california_offices o ON o.office_id = e.office_id
  WHERE office = 'CA009';
```

[✓]
```
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;

Correct
Correct. Impala allows use of aliases from the **SELECT** list in the **GROUP BY** clause.

- [x] 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;

Correct
Correct. Aliases set in the **SELECT** clause are allowed in the **ORDER BY** clause, and in Impala, they can also be used in the **GROUP BY** clause.

- [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.

8. The **california_emp** table includes one row with **name='Sandy Tilbrook'**, with **office_id='CA086'**. There is no row in **california_offices** with **office_id='CA086'**. However, there is a **office_id='CA070'** in **california_offices** with **city='Redding'**, but no rows in **california_emp** have **office_id='CA070'**. (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 point

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
- ○ A row with **name='Sandy Tilbrook'** with be included, but no row with **city='Redding'** will be included
- [✓ 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

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=NULL**) or those erroneously assigned to a non-existent office? Select all that apply.

1 / 1 point

- ☐ FROM california_emp e RIGHT OUTER JOIN california_offices o ON e.office_id=o.office_id
- ☐ FROM california_offices o LEFT OUTER JOIN california_emp e ON e.office_id=o.office_id
- ☑ 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 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**.

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 point

○ SELECT empl_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 empl_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;

○ SELECT empl_id, name

FROM california_emp e

LEFT OUTER JOIN california_offices o ON e.office_id = o.office_id

WHERE office_id IS NULL;

● SELECT empl_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 empl_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;

✓ **Correct**

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