Integrated project: Maji Ndogo part 3 [MCQ] (Version : 0)

TEST

● **Correct Answer**

Answered in 18.616666666667 Minutes

Question 1/10

The following query results in 2,698 rows of data being retrieved, but the auditor_report table only has 1,620 rows. Analyse the query and select the reason why this discrepancy occurs. Hint: Think about the type of relationship between our tables.

```
SELECT
   auditorRep.location_id,
   visitsTbl.record_id,
   Empl_Table.employee_name,
   auditorRep.true_water_source_score AS auditor_score,
   wq.subjective_quality_score AS employee_score
FROM auditor_report AS auditorRep
JOIN visits AS visitsTbl
ON auditorRep.location_id = visitsTbl.location_id
JOIN water_quality AS wq
ON visitsTbl.record_id = wq.record_id
JOIN employee as Empl_Table
ON Empl_Table.assigned_employee_id = visitsTbl.assigned_employee_id;
```

- [✓] The visits table has multiple records for each **location_id**, which when joined with **auditor_report**, results in multiple records for each **location_id**.
- [ ] The **auditor_report** table has duplicate **location_id** values causing more rows to be retrieved than expected.
- [ ] The **water_quality** table has incorrect **record_id** values causing an incorrect join with the visits table.
- [ ] The **employee** table has duplicate **assigned_employee_id** values leading to duplicate rows in the result set.

Explanation:

The visits table has multiple records for each location_id, which when joined with auditor_report, results in multiple records for each location_id." is correct. This scenario leads to more rows being retrieved due to the one-to-many relationship between the auditor_report table and the visits table.

The statement "The auditor_report table has duplicate location_id values causing more rows to be retrieved than expected." is incorrect because the auditor_report table does not have duplicate location_id values.

The statement "The employee table has duplicate assigned_employee_id values leading to duplicate rows in the result set." is incorrect because there is no information provided about duplicate assigned_employee_id values in the employee table.

The statement "The water_quality table has incorrect record_id values causing an incorrect join with the visits table." is incorrect because there is no information provided about incorrect record_id values in the water_quality table.

Question 2/10

What is the function of **Incorrect_records** in the following query?

```
WITH Incorrect_records AS ( –– This CTE fetches all of the records with wrong scores
SELECT
    auditorRep.location_id,
    visitsTbl.record_id,
    Empl_Table.employee_name,
    auditorRep.true_water_source_score AS auditor_score,
    wq.subjective_quality_score AS employee_score
FROM auditor_report AS auditorRep
JOIN visits AS visitsTbl
ON auditorRep.location_id = visitsTbl.location_id
JOIN water_quality AS wq
ON visitsTbl.record_id = wq.record_id
JOIN employee as Empl_Table
ON Empl_Table.assigned_employee_id = visitsTbl.assigned_employee_id
WHERE visitsTbl.visit_count =1 AND auditorRep.true_water_source_score != wq.subjective_quality_score)

SELECT
    employee_name,
    count(employee_name)
```

```
FROM Incorrect_records
GROUP BY Employee_name;
```

- [ ] Incorrect_records acts as a persistent storage structure, saving the records with different scores between auditor and employee in an intermediate result in the database for future queries.
- [ ] Incorrect_records creates a new permanent table in the database, storing the records with different scores between auditor and employee for further analysis.
- [ ] Incorrect_records serves as a temporary result set to store aggregated data of records with different scores between auditor and employee for the main query.
- [✓] Incorrect_records filters and organises records with different scores between auditor and employee, preparing a tailored dataset for the main query.

Explanation:

The statement "Incorrect_records filters and organises records with discrepant scores between auditor and employee, preparing a tailored dataset for the main query." is correct. Incorrect_records identifies records where there's a discrepancy between the auditor and employee scores and organises this data for use in the main query.

The statement "Incorrect_records serves as a temporary result set to store aggregated data for the main query." is incorrect because Incorrect_records doesn't aggregate data; it selects and filters data based on specified conditions.

The statement "Incorrect_records acts as a persistent storage structure, saving the intermediate results in the database for future queries." is incorrect because CTEs do not persist data beyond the execution of the query they are defined in. They are temporary result sets that exist only during the execution of the query.

The statement "Incorrect_records creates a new permanent table in the database, storing the records with discrepant scores for further analysis."

is incorrect because CTEs do not create permanent tables in the database; they serve as temporary result sets for organising and filtering data within a query.

Question 3/10

In the suspect_list CTE, a subquery is used. What type of subquery is it, and what is its purpose in the query?

suspect_list AS (
SELECT employee_name, number_of_mistakes
FROM error_count
WHERE number_of_mistakes > (SELECT AVG(number_of_mistakes) FROM error_count))

- [ ] The subquery is a multi-row subquery used to fetch a list of employee_name with above-average number_of_mistakes.
- [ ] The subquery is a table subquery used to create a temporary table for data filtering.
- [✓] The subquery is a scalar subquery used to calculate the average number_of_mistakes for comparison.
- [ ] The subquery is a correlated subquery used to compare each number_of_mistakes to the average number_of_mistakes.

Explanation:

The subquery is a scalar subquery that calculates the average number_of_mistakes, which is then used for comparison in the WHERE clause of the suspect_list CTE.

The subquery is a correlated subquery … is incorrect. The subquery is not correlated as it doesn't reference any columns from the outer query.

The subquery is a multi-row subquery … is incorrect. The subquery is not a multi-row subquery and does not fetch a list of employee_name.

The subquery is a table subquery… is incorrect. The subquery is not a table subquery and does not create a temporary table for data filtering.

Question 4/10

A colleague proposed the following CTE as an alternative to the suspect_list we used previously, but it does not give the desired results. What will be the result of this subquery?

```
suspect_list AS (
  SELECT ec1.employee_name, ec1.number_of_mistakes
  FROM error_count ec1
  WHERE ec1.number_of_mistakes >= (
    SELECT AVG(ec2.number_of_mistakes)
    FROM error_count ec2
    WHERE ec2.employee_name = ec1.employee_name))
```

- ☐ The subquery is a scalar subquery that calculates a single average number of mistakes for all employees.
- ☒ The subquery is a multi-row subquery that calculates the average mistakes for every employee_name.
- ☐ The subquery is a table subquery designed to produce a virtual table capturing average mistakes for each employee_name.
- ☐ The subquery is a correlated subquery that returns all of the employees that made errors.

Explanation:

The subquery is a scalar subquery that calculates a single average number of mistakes for all employees … is incorrect because the subquery is a correlated query that returns a single value per row.

The subquery is a multi-row subquery that calculates the average mistakes for every employee_name … is incorrect because the subquery is a correlated query that returns a single value per row that returns a single value per row. Subqueries that are used with comparison operators must be scalar values.

The subquery is a table subquery designed to produce a virtual table capturing average mistakes for each employee_name … is incorrect because the subquery is a correlated query that returns a single value per row.

Question 5/10

How is the relationship between the employee table and the visits table represented in the ERD?

- ☐ employee has a 1-to-1 relationship with visits.
- ☐ employee has a many-to-many relationship with visits.
- ☐ There is no direct relationship between the employee table and the visits table.
- ✓ employee has a 1-to-many relationship with visits.

Explanation:

employee has a 1-to-1 relationship with visits is incorrect. The ERD shows the assigned_employee_id in the visits table as a foreign key, which means multiple visits can be assigned to one employee.

employee has a many-to-many relationship with visits is incorrect. The ERD does not show a junction table between these two tables, which is characteristic of many-to-many relationships.

There is no direct relationship between the employee table and the visits table is incorrect. The assigned_employee_id in the visits table is a direct reference to the employee table.

Question 6/10

Which table contains the location_id as its primary key?

- [ ] employee
- [ ] global_water_access
- [✓ location]
- [ ] data_dictionary

Explanation:

Employee data_dictionary and global_water_access are incorrect. Based on the ERD, only the location table has location_id as its primary key.

Question 7/10

How would you modify the Incorrect_records CTE to join the well_pollution data?

```
WITH Incorrect_records AS (
SELECT
    auditorRep.location_id,
    visitsTbl.record_id,
    Empl_Table.employee_name,
    auditorRep.true_water_source_score AS auditor_score,
    wq.subjective_quality_score AS employee_score,
    auditorRep.statements AS statements
FROM auditor_report AS auditorRep
JOIN visits AS visitsTbl
ON auditorRep.location_id = visitsTbl.location_id
JOIN water_quality AS wq
ON visitsTbl.record_id = wq.record_id
JOIN employee as Empl_Table
ON Empl_Table.assigned_employee_id = visitsTbl.assigned_employee_id
WHERE visitsTbl.visit_count =1 AND auditorRep.true_water_source_score != wq.subjective_quality_score);
```

- [ ] JOIN well_pollution ON auditorRep.location_id = well_pollution.location_id
- [ ] JOIN well_pollution ON visitsTbl.record_id = well_pollution.record_id

- [✓] JOIN well_pollution
ON visitsTbl.source_id =
well_pollution.source_id

- [ ] JOIN well_pollution
ON water_quality.subjective_quality_score
= well_pollution.subjective_quality_score

Explanation:

auditorRep.location_id = well_pollution.location_id … is incorrect. The location_id doesn't directly connect to the well_pollution table.

visitsTbl.record_id = well_pollution.record_id … is incorrect. The record_id in the visits table is a unique identifier for visits, not a reference to water sources. The record_id doesn't match anything in the well_pollution table.

water_quality.subjective_quality_score = well_pollution.subjective_quality_score … is incorrect. This suggests joining on a value attribute rather than a key, which can lead to incorrect relationships. It's essential to discern between actual keys and data attributes when designing joins.

Question 8/10

Which employee just avoided our classification of having an above-average number of mistakes? Hint: Use one of the queries we used to aggregate data from Incorrect_records.

- [ ] Farai Nia
- [ ] Enitan Zuri
- [✓] Rudo Imani
- [ ] Lalitha Kaburi

Explanation:

Lalitha Kaburi is one of the suspects, with 7 mistakes

Rudo Imani has 5 mistakes

Farai Nia has 4 mistakes

Enitan Zuri has 4 mistakes

Question 9/10

Which of the following “suspects” is connected to the following civilian statement:
“Suspicion coloured villagers' descriptions of an official's aloof demeanour and apparent laziness. The reference to cash transactions casts doubt on their motives.”

- ☐ Malachi Mavuso
- [✓ Lalitha Kaburi]
- ☐ Bello Azibo
- ☐ Zuriel Matembo

Explanation:

There are some statements that are similar, for example:
Suspicion and unease coloured the villagers' accounts of an official's haughty behaviour and potential corruption. The mention of cash changing hands added to their apprehension. That would trap students who manually search the list.

Question 10/10

Consider the provided SQL query. What does it do?

```
SELECT
auditorRep.location_id,
visitsTbl.record_id,
auditorRep.true_water_source_score AS auditor_score,
wq.subjective_quality_score AS employee_score,
wq.subjective_quality_score - auditorRep.true_water_source_score  AS score_diff
FROM auditor_report AS auditorRep
JOIN visits AS visitsTbl
ON auditorRep.location_id = visitsTbl.location_id
JOIN water_quality AS wq
```

```
ON visitsTbl.record_id = wq.record_id
WHERE (wq.subjective_quality_score - auditorRep.true_water_source_score) > 9;
```

- ■ The query retrieves the auditor records where employees assigned very high scores to very poor water sources.
- ☐ The query retrieves the auditor records where the auditor found all of the records with incorrect scores.
- ☐ The query retrieves the location_id, record_id, and water scores, and calculates a difference in scores between the employee's scores and the auditor's scores.
- ✗ The query retrieves the location_id, record_id, and water scores by JOINING the water_quality and visits table, and then calculates a difference in scores between the employee's scores and the auditor's scores.

Explanation:

The query retrieves records of locations and their associated visits, focusing on instances where the difference between the employee's subjective water quality score and the auditor's true water source score is greater than 9.

The query retrieves the auditor records where the auditor found all of the records with incorrect scores. The query will only give the incorrect records, where the difference in score > 9. If the scores only differ by 9 or below, it won't be in the results set.

The query retrieves the location_id, record_id, and water scores, and calculates a difference … is incorrect. While it explains the technical details of the query, it does not recognise the filter.

The query retrieves the location_id, record_id, and water scores by JOINING … is incorrect. While it explains in detail that a JOIN occurs, and a calculation of values, it does not mention that a comparison is made to identify very incorrect scores.