# Integrated project: Maji Ndogo part 2 [MCQ] (Version : 0)

TEST

## Correct Answer

Answered in 4.1166666666667 Minutes

## Question 1/10

Which SQL query will produce the date format "DD Month YYYY" from the time_of_record column in the visits table, as a single column? Note: Monthname() acts in a similar way to DAYNAME().

- [ ] SELECT CONCAT(day(time_of_record), " ", month(time_of_record), " ", year(time_of_record)) FROM visits;
- [x] SELECT CONCAT(day(time_of_record), " ", monthname(time_of_record), " ", year(time_of_record)) FROM visits;
- [ ] SELECT day(time_of_record), monthname(time_of_record), year(time_of_record) FROM visits;
- [ ] SELECT CONCAT(monthname(time_of_record), " ", day(time_of_record), ", ", year(time_of_record)) FROM visits;

**Explanation:**

SELECT CONCAT(day(time_of_record), " ", month(time_of_record)... is incorrect because it will produce a string with the month as a number: for example: “21 01 2021”

SELECT day(time_of_record), monthname(time_of_record), ... is incorrect because the result is that the date is spread across three columns.

SELECT CONCAT(monthname(time_of_record), " ".... is incorrect because the format would be Month

DD, YYYY.

## Question 2/10

You are working with an SQL query designed to calculate the Annual Rate of Change (ARC) for basic rural water services:

```
SELECT
name,
wat_bas_r - LAG(wat_bas_r) OVER (PARTITION BY(a) ORDER BY(b))
FROM
global_water_access
ORDER BY
name;
```

To accomplish this task, what should you use for placeholders (a) and (b)?

- [ ] Use **wat_bas_r** for grouping and **year** for sorting.
- [ ] Use **name** for grouping and **wat_bas_r** for sorting.
- [x] Use **name** for grouping and **year** for sorting.
- [ ] Use **year** for grouping and **name** for sorting.

**Explanation:**

Grouping by year and sorting by name would partition the data by each year, making it ineffective for monitoring changes for a specific location over multiple years.

Grouping by wat_bas_r and sorting by year wouldn't be logical for calculating ARC. This setup groups data by the same access values, overlooking the essential dimensions of 'name' and 'year'.

Grouping by name and sorting by wat_bas_r is inappropriate for calculating the Annual Rate of Change as the order is determined by the rate itself, not the time period.

## Question 3/10

What are the names of the two worst-performing employees who visited the fewest sites, and how many sites did the worst-performing employee visit? Modify your queries from the “Honouring the workers” section.

- [ ] Pili Zola, Bello Azibo (3708)
- [ ] 20, 22 (15)
- [x] Lesedi Kofi, Kunto Asha (15)
- [ ] Kunto Asha, Lesedi Kofi (15)
- [ ] Wambui Jabali, Lesedi Kofi, Kunto Asha (15)

**Explanation:**

This is the query we want to modify:

```
SELECT assigned_employee_id,
COUNT(record_id) AS number_of_visits
FROM visits
GROUP BY assigned_employee_id
ORDER BY number_of_visits DESC
LIMIT 3;
```

We have to reverse the ORDER BY clause by removing DESC, and LIMIT to 2.

## Question 4/10

What does the following query do?

```
SELECT
    location_id,
    time_in_queue,
    AVG(time_in_queue) OVER (PARTITION BY location_id ORDER BY visit_count) AS total_avg_queue_time
FROM
    visits
WHERE
visit_count > 1 -- Only shared taps were visited > 1
ORDER BY
    location_id, time_of_record;
```

- [ ] It computes an average queue time for each shared tap, that is updated for each visit, and the results set is ordered by visit_count.
- [ ] It calculates the average queue time for shared taps.
- [ ] It calculates the average queue times for any water sources and increase each visit.
- [x] It computes an average queue time for shared taps visited more than once, which is updated each time a source is visited.

**Explanation:**

It computes an average queue time for shared taps, that is updated each time a source is visited. This option accurately captures what the query does. It computes a running average of queue times for shared taps (due to visit_count > 1), partitioned by location_id, and this average is updated for each visit based on the ORDER BY visit_count.

*It calculates the average queue time for shared taps.* This option suggests that a single average queue time is calculated for each location. This exposes a misunderstanding that window functions produce a single aggregated result per partition, rather than a running aggregate for each row within each partition.

*It calculates the average queue times for any water sources and increase each visit.* This option might lead one to think that the query calculates a new average for each individual visit with visit_count > 1. The misconception here is failing to recognise that the query calculates a running average for each shared tap, partitioned by location_id.

*It computes an average queue time for each shared tap, that is updated for each visit, and the results set is ordered by visit_count.* This option correctly identifies the function of the window function, but the results set is not ordered by visit_count but by location_id, time_of_record. The ORDER BY in the window function orders the partitions so that the running average is calculated.

## Question 5/10

One of our employees, Farai Nia, lives at 33 Angelique Kidjo Avenue. What would be the result if we TRIM() her address?

TRIM('33 Angelique Kidjo Avenue  ')

- [ ] ‘33 angelique kidjo avenue’
- [x] ‘33 Angelique Kidjo Avenue’
- [ ] ‘33 AngeliqueKidjoAvenue’
- [ ] ‘33AngeliqueKidjoAvenue’

**Explanation:**

The TRIM() function removes the leading and trailing spaces from the string '33 Angelique Kidjo Avenue '. The spaces within the address are untouched, making the correct output ‘33 Angelique Kidjo Avenue’.

‘33 AngeliqueKidjoAvenue’ – This option suggests that the TRIM() function also removes spaces within the string, which is incorrect. The TRIM() function only removes the leading and trailing spaces specified.

‘33AngeliqueKidjoAvenue’ – This option further suggests that TRIM() would remove all spaces, including those between the number "33" and the word "Angelique" and within the name itself. This is not what TRIM() does; it only removes leading and trailing spaces.

‘33 angelique kidjo avenue’ – This option is incorrect because the capitalisation does not match the original string. The TRIM() function does not alter the case of the letters within the string; it only removes leading and trailing characters specified in its arguments.

## Question 6/10

How many employees live in Dahabu? Rely on one of the queries we used in the project to answer this.

- [x] 6
- [ ] 5
- [ ] 29
- [ ] 4

**Explanation:**

29 employees live in rural areas, not Dahabu

4 employees live in the towns of Lusaka and Zanzibar, not Dahabu

5 People live in Harare, not Dahabu

## Question 7/10

How many employees live in Harare, Kilimani? Modify one of your queries from the project to answer this question.

- [ ] 9
- [x] 2
- [ ] 5
- [ ] 3

**Explanation:**

5 is incorrect because there are two Harare towns – 2 employees live in Harare, Kilimani, and 3 live in Harare, Akatsi.

3 is incorrect because 3 employees live in Harare, Akatsi, not the one in Kilimani.

9 is incorrect because it is Rural, Kilimani.

## Question 8/10

How many people share a well on average? Round your answer to 0 decimals.

- [x] 279
- [ ] 6
- [ ] 2071
- [ ] 699

**Explanation:**

699 people share river water sources.

2071 people share shared taps.

6 people on average share a broken tap.

## Question 9/10

Consider the query we used to calculate the total number of people served:

```
SELECT
SUM(number_of_people_served) AS population_served
FROM
water_source
ORDER BY
population_served
```

Which of the following lines of code will calculate the total number of people using some sort of tap?

- [ ] HAVING type_of_water_source LIKE "tap%"
- [x] WHERE type_of_water_source LIKE "%tap%"
- [ ] HAVING type_of_water_source LIKE "%tap%"

- [ ] WHERE type_of_water_source IN ("tap")

**Explanation:**

HAVING type_of_water_source LIKE "%tap%"
RESULT: Error Code: 1054. Unknown column 'type_of_water_source' in 'having clause'

HAVING type_of_water_source LIKE "tap%"
RESULT: Error Code: 1054. Unknown column 'type_of_water_source' in 'having clause'

WHERE type_of_water_source IN ("tap")
RESULT: NULL

## Question 10/10

Use the pivot table we created to answer the following question. What are the average queue times for the following times?

Saturday from 12:00 to 13:00
Tuesday from 18:00 to 19:00
Sunday from 09:00 to 10:00

- [ ] Saturday: 242, Tuesday: 145, Sunday: 79
- [x] Saturday: 239, Tuesday: 122, Sunday: 84
- [ ] Saturday: 242, Tuesday: 145, Sunday: 83
- [ ] Saturday: 248, Tuesday: 145, Sunday: 82

**Explanation:**

Saturday: 242, Tuesday: 145, Sunday: 83 are incorrect because the queue times for Saturday and Tuesday and Sunday are taken from 13:00, 18:00 and 10:00 respectively. This tests if students understand the correct time stamps to reference.

Saturday: 248, Tuesday: 145, Sunday: 82 are incorrect because while Tuesday's time is correct, the queue time for Saturday is from 15:00 and for Sunday is from 07:00. This tests for random selections of data that are close to the correct answer.

Saturday: 242, Tuesday: 145, Sunday: 79 are incorrect because while the time for Tuesday is correct, the queue time for Saturday is from 13:00 and for Sunday is from 06:00. This tests if students pick times adjacent to the correct ones.