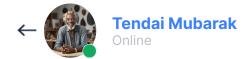


Maji Ndogo: From analysis to action

# Weaving the data threads of Maji Ndogo's narrative















Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## **Integrating the report**

Adding the auditor report to our database.



#### **Linking records**

Joining employee data to the report.



## Gathering evidence

Building a complex query seeking truth.

# Subject: Results of audit on Maji Ndogo water project

Dear President Naledi,

I hope this email finds you in the best of health and spirits. As you know, my team and I were tasked with conducting an independent audit of the Maji Ndogo water project, specifically the database recording water sources in our country, following the inconsistencies identified by Chidi Kunto and his team.

For clarity, in this specific audit, our objective was to assess the integrity, and accuracy of the data stored in the database. Auditing a database involves verifying that the data it contains is both accurate and has not been tampered with, thereby ensuring that the information can be relied upon for decision-making and governance.

I am pleased to report that the audit is now complete. After a rigorous examination of the database's records, as well as the procedures in place for data entry and modification, we can confirm that the vast majority of the data aligns with the principles of good governance and data-driven decision-making that you have so vigorously championed.

However, we did find some data that was tampered with, which requires your immediate attention. I have attached the records I have re-examined for your review.

Thank you for entrusting us with this crucial task. Your commitment to accountability and transparency is truly commendable.

Sincerely, Tendai Mubarak Chief Auditor



Auditor\_report.csv







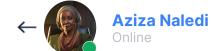




















Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



#### **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

## Re: Results of audit on Maji Ndogo water project

Dear Tendai,

Thank you for the meticulous work you and your team have put into auditing the Maji Ndogo water project. The depth of your analysis reflects your commitment to excellence and accountability.

I am heartened to learn that our operations are largely in line with our principles of governance. At the same time, your findings are valuable as they highlight areas we can further improve.

Rest assured, I will be convening our data team to address any issues and take any steps necessary to ensure the integrity of our data.

Once again, I commend you and your team for your hard work and dedication. Maji Ndogo is indeed counting on all of us to deliver on our promises. Thank you for playing your part.

All the best,

Aziza Naledi

























Setting the stage for our data exploration journey.



# Generating an ERD

Understanding the database structure.



#### Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



10:30

Hey! It's been a while! I hope you are ready to get stuck in. We got back the auditor's report, so we need to compare their results to ours.

10:33

To integrate the auditor's report, we will need to access many of the tables in the database, so it is important to understand the database structure. To do this we really need to understand the relationships first, so we know where to pull information from. Can you please get the ERD for the md\_water\_services database? Spend a few minutes looking at the diagram.

10:55

Note that the visits table is the central table. location\_id, source\_id and assigned\_employee\_id are primary keys in their respective tables, but are all foreign keys in visits. These are mostly dsone-to-many relationships. Let's think some of these through so we are sure this is correct.

10:59

The visits table logs all the times we've been to different places, and we can see that some locations have been visited multiple times. On the other hand, the location table has all the specifics about each place we've been but it only includes each location once. So, it's a one-to-many relationship: for each unique location in the location table, there might be many corresponding records in the visits table detailing all the different times we went there.



























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



#### **Integrating the report**

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



#### **Gathering evidence**

Building a complex query seeking truth.



11:03

But if we look at the ERD, it shows a many-to-one relationship. This does not agree with our thinking. Errors like these can cause problems, so let's fix that.

11:04

We should normally be careful making a change like this. First you should check that record\_id is unique for both tables, and are indeed one-to-one.

11:04

I did that, so all we have to do is right-click on the relationship line and select Edit relationship, then at the bottom, select the Foreign key tab, and change the Cardinality to one-to-one. Now the relationship line should indicate a one-to-one relation.

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

# **Integrating the Auditor's report**

Ok, I sent you a .csv file of the auditor's results. You should get it loaded into SQL. You will have to think back to the start of our journey on how to do that.

11:13

11:10

To make sure we have the same names, use this query to create the table first, then Import the CSV file:

```
DROP TABLE IF EXISTS `auditor_report`;

CREATE TABLE `auditor_report` (
  `location_id` VARCHAR(32),
  `type_of_water_source` VARCHAR(64),
  `true_water_source_score` int DEFAULT NULL,
  `statements` VARCHAR(255)
);
```

11:14

Got it? Good. Looking at the table and the data dictionary I sent you, you can see the type of info we have.

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

Wow! First off, it looks like we have 1620 records, or sites that they re-visited. I see a location\_id, type of water source at that location, and the quality score of the water source, that is now independently measured. Our auditor also investigated each site a bit by speaking to a few locals. Their statements are also captured in his results.

11:22

We need to tackle a couple of questions here.

- 1. Is there a difference in the scores?
- 2. If so, are there patterns?

11:28

For the first question, we will have to compare the quality scores in the water\_quality table to the auditor's scores. The auditor\_report table used location\_id, but the quality scores table only has a record\_id we can use. The visits table links location\_id and record\_id, so we can link the auditor\_report table and water\_quality using the visits table.

11:37

So first, grab the location\_id and true\_water\_source\_score columns from auditor\_report.

























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



# **Integrating the report**

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



location_id	true_water_source_score
AkHa00008	3
AkHa00053	9

11:46

Now, we join the visits table to the auditor\_report table. Make sure to grab subjective\_quality\_score, record\_id and location\_id.

11:53

## I get:

audit_location	true_water_source_score	visit_location	record_id
SoRu34980	1	SoRu34980	5185
AkRu08112	3	AkRu08112	59367
AkLu02044	0	AkLu02044	37379
			•••

























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



# Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

Note that I specified from which table each selected column is from in this query:

```
SELECT
```

```
auditor_report.location_id AS audit_location,
   auditor_report.true_water_source_score,
   visits.location_id AS visit_location,
   visits.record_id

FROM
   auditor_report

JOIN
   visits
   ON auditor_report.location_id = visits.location_id;
```

12:02

Now that we have the record\_id for each location, our next step is to retrieve the corresponding scores from the water\_quality table. We are particularly interested in the subjective\_quality\_score. To do this, we'll JOIN the visits table and the water\_quality table, using the record\_id as the connecting key.

12:04

## Whoo, first try:

audit_location	true_water_source_score	visit_location	record_id	subjective_quality_score
SoRu34980	1	SoRu34980	5185	1
AkRu08112	3	AkRu08112	59367	3
AkLu02044	0	AkLu02044	37379	0

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



#### **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

It doesn't matter if your columns are in a different format, because we are about to clean this up a bit. Since it is a duplicate, we can drop one of the location\_id columns. Let's leave record\_id and rename the scores to surveyor\_score and auditor\_score to make it clear which scores we're looking at in the results set.

12:17

## I got this:

location_id	record_id	auditor_score	surveyor_score
SoRu34980	5185	1	1
AkRu08112	59367	3	3
AkLu02044	37379	0	0
AkHa00421	51627	3	3
	•••		

12:18

Since were joining 1620 rows of data, we want to keep track of the number of rows we get each time we run a query. We can either set the maximum number of rows we want from "Limit to 1000 rows" to a larger number like 10000, or we can force SQL to give us all of the results, using LIMIT 10000.

12:19

Ok, let's analyse! A good starting point is to check if the auditor's and exployees' scores agree. There are many ways to do it. We can have a WHERE clause and check if surveyor\_score = auditor\_score, or we can subtract the two scores and check if the result is 0.

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

You try and figure this one out. If you run into an error, remember that you can't use aliases in WHERE, so you have to use the same name as in the SELECT part, like this: auditor\_report.true\_water\_source\_score

12:28

You got 2505 rows right? Some of the locations were visited multiple times, so these records are duplicated here. To fix it, we set visits.visit\_count = 1 in the WHERE clause. Make sure you reference the alias you used for visits in the join.

12:30

With the duplicates removed I now get 1518. What does this mean considering the auditor visited 1620 sites?

12:31

I think that is an excellent result. 1518/1620 = 94% of the records the auditor checked were correct!!

12:43

But that means that 102 records are incorrect. So let's look at those. You can do it by adding one character in the last query!

























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



location_id	record_id	auditor_score	surveyor_score
AkRu05215	21160	3	10
KiRu29290	7938	3	10
KiHa22748	43140	9	10
SoRu37841	18495	6	10
KiRu27884	33931	1	10
KiZu31170	17950	9	10
	•••		

12:52

Since we used some of this data in our previous analyses, we need to make sure those results are still valid, now we know some of them are incorrect. We didn't use the scores that much, but we relied a lot on the type\_of\_water\_source, so let's check if there are any errors there.

12:59

So, to do this, we need to grab the type\_of\_water\_source column from the water\_source table and call it survey\_source, using the source\_id column to JOIN. Also select the type\_of\_water\_source from the auditor\_report table, and call it auditor\_source.

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



# Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



location_id	auditor_source	survey_source	record_id	auditor_score	surveyor_score
AkRu05215	well	well	21160	3	10
KiRu29290	shared_tap	shared_tap	7938	3	10
KiHa22748	tap_in_home_broken	tap_in_home_broken	43140	9	10
SoRu37841	shared_tap	shared_tap	18495	6	10
KiRu27884	well	well	33931	1	10

13:03

So what I can see is that the types of sources look the same! So even though the scores are wrong, the integrity of the type\_of\_water\_source data we analysed last time is not affected.

13:14

Once you're done, remove the columns and JOIN statement for water\_sources again.

13:15











13















Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



#### Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



Next up, let's look at where these errors may have come from. At some of the locations, employees assigned scores incorrectly, and those records ended up in this results set.

13:21

I think there are two reasons this can happen.

- 1. These workers are all humans and make mistakes so this is expected.
- 2. Unfortunately, the alternative is that someone assigned scores incorrectly on purpose!

13:25

In either case, the employees are the source of the errors, so let's JOIN the assigned\_employee\_id for all the people on our list from the visits table to our query. Remember, our query shows the shows the 102 incorrect records, so when we join the employee data, we can see which employees made these incorrect records.

13:26











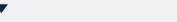
14













Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



**Chidi Kunto** 

location_id	record_id	assigned_employee_id	auditor_score	surveyor_score
AkRu05215	21160	34	3	10
KiRu29290	7938	1	3	10
KiHa22748	43140	1	9	10
SoRu37841	18495	34	6	10
KiRu27884	33931	1	1	10
KiZu31170	17950	5	9	10
	•••			

13:31

So now we can link the incorrect records to the employees who recorded them. The ID's don't help us to identify them. We have employees' names stored along with their IDs, so let's fetch their names from the employees table instead of the ID's.

13:41

# I get this table:

location_id	record_id	employee_name	auditor_score	surveyor_score
AkRu05215	21160	Rudo Imani	3	10
KiRu29290	7938	Bello Azibo	3	10
KiHa22748	43140	Bello Azibo	9	10
SoRu37841	18495	Rudo Imani	6	10
	•••	•••		

13:52











15















Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



#### Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

Well this query is massive and complex, so maybe it is a good idea to save this as a CTE, so when we do more analysis, we can just call that CTE like it was a table. Call it something like Incorrect\_records. Once you are done, check if this query SELECT \* FROM Incorrect\_records, gets the same table back.

13:57

Now that we defined Incorrect\_records, we can query it like any other table.

14:00

Let's first get a unique list of employees from this table. Think back to the start of your SQL journey to answer this one. I got 17 employees.

14:02

Next, let's try to calculate how many mistakes each employee made. So basically we want to count how many times their name is in Incorrect\_records list, and then group them by name, right?

14:07

# Something like this:

employee_name	number_of_mistakes
Rudo Imani	5
Bello Azibo	26
Zuriel Matembo	17
Yewande Ebele	3
•••	

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



# Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

Do you see a pattern there? Order the list if you can't see it yet.

14:15

It looks like some of our surveyors are making a lot of "mistakes" while many of the other surveyors are only making a few. I don't like where this is going!

























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.



Ok, so thinking about this a bit. How would we go about finding out if any of our employees are corrupt?

14:21

Let's say all employees make mistakes, if someone is corrupt, they will be making a lot of "mistakes", more than average, for example. But someone could just be clumsy, so we should try to get more evidence...

14:23

Our auditor did say some of the things he heard on the streets were quite shady, and he recorded this in the statements column. Considering both of these sources should give us a pretty reliable answer.

























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



#### **Integrating the report**

Adding the auditor report to our database.



#### **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

So let's try to find all of the employees who have an above-average number of mistakes. Let's break it down into steps first:

- 1. We have to first calculate the number of times someone's name comes up. (we just did that in the previous query). Let's call it error\_count.
- 2. Then, we need to calculate the average number of mistakes employees made. We can do that by taking the average of the previous query's results. Something like this:

```
SELECT
```

```
AVG(number_of_mistakes)
FROM
```

error\_count;

Let's call that result avg\_error\_count\_per\_empl, which would be a scalar value.

3. Finaly we have to compare each employee's error\_count with avg\_error\_count\_per\_empl. We will call this results set our suspect\_list. Remember that we can't use an aggregate result in WHERE, so we have to use avg\_error\_count\_per\_empl as a subquery.

#### **SELECT**

```
employee_name,
   number_of_mistakes
FROM
   error_count
WHERE
   number_of_mistakes > (avg_error_count_per_empl);
```

14:26

1. Let's start by cleaning up our code a bit. First, Incorrect\_records is a result we'll be using for the rest of the analysis, but it makes the query a bit less readable. So, let's convert it to a VIEW. We can then use it as if it was a table. It will make our code much simpler to read, but, it comes at a cost. We can add comments to CTEs in our code, so if we return to that query a year later, we can read those comments and quickly understand what Incorrect\_records represents. If we save it as a VIEW, it is not as obvious. So we should add comments in places where we use Incorrect\_records.



















14:28





#### Introduction

Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## **Integrating the report**

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

So, replace WITH with CREATE VIEW like this, and note that I added the statements column to this table in line 8 too:

```
CREATE VIEW Incorrect_records AS (
SELECT
    auditor_report.location_id,
   visits.record_id,
    employee.employee_name,
    auditor_report.true_water_source_score AS auditor_score,
   wq.subjective_quality_score AS surveyor_score,
   auditor_report.statements AS statements
FROM
    auditor_report
JOIN
    visits
   ON auditor_report.location_id = visits.location_id
JOIN
   water_quality AS wg
   ON visits.record_id = wq.record_id
JOIN
    employee
   ON employee.assigned_employee_id = visits.assigned_employee_id
```

Now, calling SELECT \* FROM Incorrect\_records gives us the same result as the CTE did.

AND auditor\_report.true\_water\_source\_score != wq.subjective\_quality\_score);

14:30











WHERE

visits.visit count =1







Like this:

-- Query

SELECT \* FROM error\_count;









#### Introduction

Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

Next, we convert the query error\_count, we made earlier, into a CTE. Test it to make sure it gives the same result again, using SELECT \* FROM Incorrect\_records. On large gueries like this, it is better to build the guery, and test each step, because fixing errors becomes harder as the query grows.

14:31

```
WITH error_count AS ( -- This CTE calculates the number of mistakes each employee made
    SELECT
        employee_name,
        COUNT(employee_name) AS number_of_mistakes
    FROM
        Incorrect_records
            /_{\bigstar} Incorrect_records is a view that joins the audit report to the database
            for records where the auditor and
            employees scores are different*/
    GROUP BY
        employee_name)
```

14:33

2. Now calculate the average of the number\_of\_mistakes in error\_count. You should get a single value.



























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



#### **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

3. To find the employees who made more mistakes than the average person, we need the employee's names, the number of mistakes each one made, and filter the employees with an above-average number of mistakes.

**HINT:** Use SELECT AVG(mistake\_count) FROM error\_count as a custom filter in the WHERE part of our query.

14:37

## There they are:

employee_name	number_of_mistakes
Zuriel Matembo	17
Malachi Mavuso	21
Lalitha Kaburi	7
Bello Azibo	26

These are the employees who made more mistakes, on average, than their peers, so let's have a closer look at them.

14:39

We should look at the Incorrect\_records table again, and isolate all of the records these four employees gathered. We should also look at the statements for these records to look for patterns.

14:40

First, convert the suspect\_list to a CTE, so we can use it to filter the records from these four employees. Make sure you get the names of the four "suspects", and their mistake count as a result, using SELECT employee\_name FROM suspect\_list.



























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



#### **Integrating the report**

Adding the auditor report to our database.



#### **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

You should get a column of names back. So let's just recap here...

- 1. We use Incorrect\_records to find all of the records where the auditor and employee scores don't match.
- 2. We then used error\_count to aggregate the data, and got the number of mistakes each employee made.
- 3. Finally, suspect\_list retrieves the data of employees who make an above-average number of mistakes.

Now we can filter that Incorrect\_records view to identify all of the records associated with the four employees we identified.

14:43

Firstly, let's add the statements column to the Incorrect\_records view. Then pull up all of the records where the employee\_name is in the suspect list. HINT: Use SELECT employee\_name FROM suspect\_list as a subquery in WHERE.

14:44

# I got something like this:

employee_name	location_id	statements
Bello Azibo	Kils23853	Villagers' wary accounts of an official's arrogance
Bello Azibo	KiHa22748	A young girl's hopeful eyes
Bello Azibo	KiRu27884	A traditional healer's empathy t
Zuriel Matembo	KiZu31170	A community leader stood with
Bello Azibo	AkRu06495	A healthcare worker in



























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



# Integrating the report

Adding the auditor report to our database.



## Linking records

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

```
Using this kind of query:
```

```
-- This query filters all of the records where the "corrupt" employees gathered data.
SELECT
. . .
FROM
    Incorrect_records
WHERE
    employee_name IN (SELECT employee_name FROM suspect_list);
```

























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

This guery is complex, right! But, if we document it well, it is simpler to understand. Oh, and you don't want to see what this guery looks like using only subqueries!

```
WITH error_count AS ( -- This CTE calculates the number of mistakes each employee made
    SELECT
        employee_name,
       COUNT(employee_name) AS number_of_mistakes
    FROM
       Incorrect_records
            /* Incorrect_records is a view that joins the audit report to the database
            for records where the auditor and
            employees scores are different*/
    GROUP BY
        employee_name),
suspect_list AS (-- This CTE SELECTS the employees with above-average mistakes
    SELECT
        employee_name.
       number of mistakes
   FROM
        error_count
   WHERE
       number_of_mistakes > (SELECT AVG(number_of_mistakes) FROM error_count))
-- This query filters all of the records where the "corrupt" employees gathered data.
SELECT
    employee_name,
   location_id,
    statements
FROM
    Incorrect_records
WHERE
    employee_name in (SELECT employee_name FROM suspect_list);
```





























Setting the stage for our data exploration journey.



# **Generating an ERD**

Understanding the database structure.



## Integrating the report

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

If you have a look, you will notice some alarming statements about these four officials (look at these records: AkRu04508, AkRu07310, KiRu29639, AmAm09607, for example. See how the word "cash" is used a lot in these statements.

14:49

Filter the records that refer to "cash".

14:50

Let's just do one more check to make sure...

14:57

Check if there are any employees in the Incorrect\_records table with statements mentioning "cash" that are not in our suspect list. This should be as simple as adding one word.

14:58

I get an empty result, so no one, except the four suspects, has these allegations of bribery.

























Setting the stage for our data exploration journey.



## **Generating an ERD**

Understanding the database structure.



#### **Integrating the report**

Adding the auditor report to our database.



## **Linking records**

Joining employee data to the report.



## **Gathering evidence**

Building a complex query seeking truth.

So we can sum up the evidence we have for Zuriel Matembo, Malachi Mavuso, Bello Azibo and Lalitha Kaburi:

- 1. They all made more mistakes than their peers on average.
- 2. They all have incriminating statements made against them, and only them.

Keep in mind, that this is not decisive proof, but it is concerning enough that we should flag it. Pres. Naledi has worked hard to stamp out corruption, so she would urge us to report this.

15:06

I am a bit shocked to be honest! After all our teams set out to do, it is hard for me to uncover this. I'll let Pres. Naledi know what we found out.









