EXPLORE AI ACADEMY

Maji Ndogo: From analysis to action

Clustering data to unveil Maji Ndogo's water crisis

Aziza Naledi
Online

Dear Team,

Our mission, as arduous as it is essential, requires us to delve deeper into our reservoir of data. To truly illuminate the road ahead, we must magnify our analysis, moving beyond isolated data points to discern larger patterns and trends.
In this next step, we will cluster our data, stepping back from the individual figures to gain a panoramic understanding. This bird's eye view will allow us to unearth broader narratives and hidden correlations concealed within our rich dataset.
Next, we must pay heed to the different forms of data in our possession. They are not mere numbers or dates; they are stories waiting to be deciphered. Their unique structure, though challenging, brims with valuable insights. As we process these, we unlock deeper layers of understanding.
Bear in mind that every piece of information you decipher, every category you determine, brings us one stride closer to our noble goal. It's through the intricate details and broader brushstrokes of data that we will uncover the solutions to Maji Ndogo's water crisis.
Your unwavering commitment to this mission emboldens me. Together, we continue marching forward, using data and dedication as our compass, towards a brighter, more secure future for Maji Ndogo.

Thank you for all your tireless efforts.

Warm regards,
Aziza Naledi

06:11

Chidi Kunto
Online

Hi Pres. Naledi,

I hope you're doing well. While diving into our recent survey data for the Maji Ndogo water project, our team stumbled upon some inconsistencies that caught our eye. It's nothing alarming, but we think it's worth a closer look.
Would you consider bringing in an independent auditor to double-check some of the records? I think it's a smart move to ensure everything is on the up-and-up. After all, we're all about accuracy and trust.

Feel free to reach out if you want to chat more about this or need more details.

Take care,
Chidi Kunto

06:45

Aziza Naledi
Online

Hi Chidi,

Thanks for catching that, and for being so attentive to detail. I'm right there with you on this - we want to be sure we're working with the best information possible.
I'll get an independent auditor on this ASAP. They'll touch base with you and the rest of the team to get things rolling. I've cc'ed everyone so that we're all on the same page.

Appreciate your diligence, Chidi. Let's keep up the great work. Maji Ndogo is counting on us.

All the best,
Aziza Naledi

07:04

Chidi Kunto
Online

Before we start, scan through the data dictionary, and perhaps query a couple of tables to get a feel for the database again.

08:31

Cleaning our data

Ok, bring up the `employee` table. It has info on all of our workers, but note that the email addresses have not been added. We will have to send them reports and figures, so let's update it. Luckily the emails for our department are easy: `first_name.last_name@ndogowater.gov`.

08:31

I am going to guide you through this one, so code along.

08:31

We can determine the email address for each employee by:
- selecting the `employee_name` column
- replacing the space with a full stop
- make it lowercase
- and stitch it all together

08:38

We have to update the database again with these email addresses, so before we do, let's use a SELECT query to get the format right, then use UPDATE and SET to make the changes.

08:41

Chidi Kunto
Online

First up, let's remove the space between the first and last names using `REPLACE()`. You can try this:

```
SELECT
    REPLACE(employee_name, ' ','.') -- Replace the space with a full stop
FROM
    employee
```

08:45

Then we can use `LOWER()` with the result we just got. Now the name part is correct.

```
SELECT
    LOWER(REPLACE(employee_name, ' ','.')) -- Make it all lower case
FROM
    employee
```

08:47

We then use `CONCAT()` to add the rest of the email address:

```
SELECT
    CONCAT(
    LOWER(REPLACE(employee_name, ' ', '.')), '@ndogowater.gov') AS new_email -- add it all together
FROM
    employee
```

08:49

Chidi Kunto
Online

Quick win! Since you have done this before, you can go ahead and UPDATE the `email` column this time with the email addresses. Just make sure to check if it worked!

```
UPDATE employee
SET email = CONCAT(LOWER(REPLACE(employee_name, ' ', '.')),
            '@ndogowater.gov')
```

08:50

I picked up another bit we have to clean up. Often when databases are created and updated, or information is collected from different sources, errors creep in. For example, if you look at the phone numbers in the `phone_number` column, the values are stored as strings.

08:53

The phone numbers should be 12 characters long, consisting of the plus sign, area code (99), and the phone number digits. However, when we use the **`LENGTH(column)`** function, it returns 13 characters, indicating there's an extra character.

```
SELECT
   LENGTH(phone_number)
FROM
   employee;
```

08:54

That's because there is a space at the end of the number! If you try to send an automated SMS to that number it will fail. This happens so often that they create a function, especially for trimming off the space, called `TRIM(column)`.

It removes any leading or trailing spaces from a string.

08:56

Chidi Kunto
Online

Use `TRIM()` to write a `SELECT` query again, make sure we get the string without the space, and then `UPDATE` the record like you just did for the emails. If you need more information about `TRIM()`, Google "TRIM documentation MySQL".

08:57

Honouring the workers

Before we dive into the analysis, let's get you warmed up a bit!
Let's have a look at where our employees live.

09:06

Use the `employee` table to count how many of our employees live in each town. Think carefully about what function we should use and how we should aggregate the data.

09:07

Chidi Kunto
Online

I get this:

town_name	num_employees
Ilanga	3
Rural	29
Lusaka	4
Zanzibar	4
Dahabu	6
Kintampo	1
Harare	5
Yaounde	1
...	...

09:08

Note how many of our workers are living in smaller communities in the rural parts of Maji Ndogo.

09:18

Pres. Naledi congratulated the team for completing the survey, but we would not have this data were it not for our field workers. So let's gather some data on their performance in this process, so we can thank those who really put all their effort in.

09:19

Pres. Naledi has asked we send out an email or message congratulating the top 3 field surveyors. So let's use the database to get the `employee_ids` and use those to get the names, email and phone numbers of the three field surveyors with the most location visits.

09:20

Chidi Kunto
Online

Let's first look at the number of records each employee collected. So find the correct table, figure out what function to use and how to group, order and limit the results to only see the top 3 `employee_ids` with the highest number of locations visited.

09:22

You should get a table like this but in a **different** order:

assigned_employee_id	number_of_visits
0	1099
1	3708
2	2033

09:29

Make a note of the top 3 `assigned_employee_id` and use them to create a query that looks up the employee's info. Since you're a pro at finding stuff in a database now, you can figure this one out. You should have a column of names, email addresses and phone numbers for our top dogs.

09:41

I'll send that off to Pres. Naledi. But this survey is not primarily about our employees, so let's get working on the main task! We'll start looking at some of the tables in the dataset at a larger scale, identify some trends, summarise important data, and draw insights.

09:49

Analysing locations

Looking at the `location` table, let's focus on the `province_name`, `town_name` and `location_type` to understand where the water sources are in Maji Ndogo.

09:52

Create a query that counts the number of records per town

09:56

For example, if we count the number of records for each **town**, I get:

records_per_town	town_name
23740	Rural
1650	Harare
1090	Amina
1070	Lusaka
990	Mrembo
930	Asmara
...	...

10:02

Now count the records per province.

10:06

My results:

records_per_province	province_name
9510	Kilimani
8940	Akatsi
8220	Sokoto
6950	Amanzi
6030	Hawassa
...	...

10:09

From this table, it's pretty clear that most of the water sources in the survey are situated in small rural communities, scattered across Maji Ndogo.

If we count the records for each province, most of them have a similar number of sources, so every province is well-represented in the survey.

10:12

Can you find a way to do the following:

1. Create a result set showing:
    - **`province_name`**
    - **`town_name`**
    - An aggregated count of records for each town (consider naming this **`records_per_town`**).
    - Ensure your data is grouped by both **`province_name`** and **`town_name`**.
2. Order your results primarily by **`province_name`**. Within each province, further sort the towns by their record counts in descending order.

10:23

Your table should look something like this:

province_name	town_name	records_per_town
Akatsi	Rural	6290
Akatsi	Lusaka	1070
Akatsi	Harare	800
Akatsi	Kintampo	780
Amanzi	Rural	3100
Amanzi	Asmara	930

10:27

These results show us that our field surveyors did an excellent job of documenting the status of our country's water crisis. Every province and town has many documented sources.

This makes me confident that the data we have is reliable enough to base our decisions on. This is an insight we can use to communicate data integrity, so let's make a note of that.

10:33

Finally, look at the number of records for each location type

10:35

Like this:

num_sources	location_type
15910	Urban
23740	Rural

10:37

We can see that there are more rural sources than urban, but it's really hard to understand those numbers. Percentages are more relatable. If we use SQL as a very overpowered calculator:

```
SELECT 23740 / (15910 + 23740) * 100
```

We can see that 60% of all water sources in the data set are in rural communities.

10:44

So again, what are some of the insights we gained from the `location` table?

1. Our entire country was properly canvassed, and our dataset represents the situation on the ground.
2. 60% of our water sources are in rural communities across Maji Ndogo. We need to keep this in mind when we make decisions.

10:55

Chidi Kunto
Online

Diving into the sources

Ok, `water_source` is a big table, with lots of stories to tell, so strap in!

11:02

Before I go and spoil it all, open up the table, look at the various columns, make some notes on what we can do with them, and go ahead and make some queries and explore the dataset. Perhaps you see something I don't.

11:04

The way I look at this table; we have access to different water source types and the number of people using each source.

These are the questions that I am curious about.
1. How many people did we survey in total?
2. How many wells, taps and rivers are there?
3. How many people share particular types of water sources on average?
4. How many people are getting water from each type of source?

11:10

I'll leave the first one to you. Try answering the rest on your own too.

11:15

For the second question, we want to count how many of each of the different water source types there are, and remember to sort them.

11:17

Chidi Kunto
Online

I get something like this:

type_of_water_source	number_of_sources
tap_in_home	7265
tap_in_home_broken	5856
...	...

11:18

Which of those sources stands out? It is pretty clear that although there was a drought, water is still abundant in Maji Ndogo. This isn't just an informative result, we will need these numbers to understand how much all of these repairs will cost. If we know how many taps we need to install, and we know how much it will cost to install them, we can calculate how much it will cost to solve the water crisis.

11:25

Ok next up, question 3: What is the average number of people that are served by each water source? Remember to make the numbers easy to read.

11:30

Chidi Kunto
Online

I got:

type_of_water_source	ave_people_per_source
tap_in_home	644
tap_in_home_broken	649
well	279
shared_tap	2071
river	699
...	...

11:32

These results are telling us that 644 people share a `tap_in_home` on average. Does that make sense?

11:40

No it doesn't, right?
Remember I told you a few important things that apply to `tap_in_home` and `broken_tap_in_home`? The surveyors combined the data of many households together and added this as a single tap record, but each household actually has its own tap. In addition to this, there is an average of 6 people living in a home. So 6 people actually share 1 tap (not 644).

11:42

Chidi Kunto
Online

It is always important to think about data. We tend to just analyse, and calculate at the start of our careers, but the value we bring as data practitioners is in understanding the meaning of results or numbers, and interpreting their meaning.

Imagine we were presenting this to the President and all of the Ministers, and one of them asks us: "Why does it say that 644 share a home tap?" and we had no answer.

It happened to me once... I don't miss those days.

11:45

This means that 1 `tap_in_home` actually represents 644 ÷ 6 = ± 100 taps.

11:47

Calculating the average number of people served by a single instance of each water source type helps us understand the typical capacity or load on a single water source. This can help us decide which sources should be repaired or upgraded, based on the average impact of each upgrade. For example, wells don't seem to be a problem, as fewer people are sharing them.

11:48

On the other hand, 2000 share a single public tap on average! We saw some of the queue times last time, and now we can see why. So looking at these results, we probably should focus on improving shared taps first.

11:49

Now let's calculate the total number of people served by each type of water source in total, to make it easier to interpret, order them so the most people served by a source is at the top.

11:50

This is what I got:

type_of_water_source	population_served
shared_tap	11945272
well	4841724
tap_in_home	4678880
tap_in_home_broken	3799720
river	2362544
...	...

11:53

It's a little hard to comprehend these numbers, but you can see that one of these is dominating. To make it a bit simpler to interpret, let's use percentages. First, we need the total number of citizens then use the result of that and divide each of the `SUM(number_of_people_served)` by that number, times 100, to get percentages.

11:56

Make a note of the number of people surveyed in the first question we answered. I get a total of about 27 million citizens!

11:57

Next, calculate the percentages using the total we just got.

12:05

Chidi Kunto
Online

Mine is:

type_of_water_source	percentage_people_per_source
shared_tap	43.2359
well	17.5246
tap_in_home	16.9352
tap_in_home_broken	13.7531
river	8.5512
...	...

12:07

Having percentages with a bunch of decimals really doesn't help get the point across, does it?

12:09

Let's round that off to 0 decimals, and order the results.

12:11

Yeah, this looks better, right?

type_of_water_source	percentage_people_per_source
shared_tap	43
well	18
tap_in_home	17
tap_in_home_broken	14
river	9
...	...

12:13

43% of our people are using shared taps in their communities, and on average, we saw earlier, that 2000 people share one `shared_tap`.

12:18

By adding `tap_in_home` and `tap_in_home_broken` together, we see that 31% of people have water infrastructure installed in their homes, but 45% (14/31) of these taps are not working! This isn't the tap itself that is broken, but rather the infrastructure like treatment plants, reservoirs, pipes, and pumps that serve these homes that are broken.
18% of people are using wells. But only 4916 out of 17383 are clean = 28% (from last week).

12:23

Start of a solution
At some point, we will have to fix or improve all of the infrastructure, so we should start thinking about how we can make a data-driven decision how to do it. I think a simple approach is to fix the things that affect most people first. So let's write a query that ranks each type of source based on how many people in total use it. `RANK()` should tell you we are going to need a window function to do this, so let's think through the problem.

12:26

We will need the following columns:

- Type of sources -- Easy
- Total people served grouped by the types -- We did that earlier, so that's easy too.
- A rank based on the total people served, grouped by the types -- A little harder.

12:30

Let's look at the results from the last query again:

type_of_water_source	population_served
shared_tap	11945272
well	4841724
tap_in_home	4678880
tap_in_home_broken	3799720
river	2362544
...	...

12:36

It should be clear for you to see what the rank is, but we want to let SQL do it.

12:37

But think about this: If someone has a tap in their home, they already have the best source available. Since we can't do anything more to improve this, we should remove `tap_in_home` from the ranking before we continue.

12:38

So use a window function on the total people served column, converting it into a rank.

12:41

I get something like this:

type_of_water_source	people_served	rank_by_population
shared_tap	111945272	1
well	4841724	2
...	...	...

12:44

Ok, so we should fix shared taps first, then wells, and so on. But the next question is, **which** shared taps or wells should be fixed first? We can use the same logic; the most used sources should really be fixed first.

12:53

So create a query to do this, and keep these requirements in mind:

1. The sources within each type should be assigned a rank.
2. Limit the results to only improvable sources.
3. Think about how to partition, filter and order the results set.
4. Order the results to see the top of the list.

12:57

This is what I got using `RANK`:

source_id	type_of_water_source	number_of_people_served	priority_rank
...	...	...	...
AmRu14978224	river	400	3364
HaDj16848224	river	400	3364
HaRu19509224	shared_tap	3998	1
AkRu05603224	shared_tap	3998	1
...	...	...	...

12:58

By using `RANK()` teams doing the repairs can use the value of rank to measure how many they have fixed, but what would be the benefits of using `DENSE_RANK()`?

Maybe it is easier to explain to the engineers this way, or the priority feels a bit more natural?

13:02

Chidi Kunto
Online

What about `ROW_NUMBER()`? Since each source now has a unique rank, teams don't have to think whether they should repair AkRu05603224, or AkRu04862224 first (both serve 3998 people), because `ROW_NUMBER()` doesn't consider records that are equal.

13:03

Try the different ranking functions in queries. Imagine yourself in an engineer's boots, and try to interpret the priority list. Thinking about the user of a table helps us to design the table better.
In that line of thought, would it make sense to give them a list of source_ids? How would they know where to go?

13:05

Chidi Kunto
Online

Analysing queues

Ok, this is the really big, and last table we'll look at this time. The analysis is going to be a bit tough, but the results will be worth it, so stretch out, grab a drink, and let's go!

13:11

A recap from last time:
The `visits` table documented all of the visits our field surveyors made to each location. For most sources, one visit was enough, but if there were queues, they visited the location a couple of times to get a good idea of the time it took for people to queue for water. So we have the time that they collected the data, how many times the site was visited, and how long people had to queue for water.

13:15

So, look at the information we have available, and think of what we could learn from it. Remember we can use some DateTime functions here to get some deeper insight into the water queueing situation in Maji Ndogo, like which day of the week it was, and what time.

13:17

Ok, these are some of the things I think are worth looking at:

1. How long did the survey take?
2. What is the average total queue time for water?
3. What is the average queue time on different days?
4. How can we communicate this information efficiently?

13:18

Chidi Kunto
Online

Try to answer some of these questions. Think of the data you will need to answer each question, and how to transform that data into the right form to answer each question. Then make some queries to try and answer them. You learned all of the skills you need, so give it a try.

13:22

HINT: I had to read up a bit on control flow, DateTime and window functions to do these, so you probably will have to as well.

13:27

Look at `visits`, especially the `time_of_record` column. It is an SQL DateTime datatype, so we can use all of the DateTime functions to aggregate data for each day and even per hour.

13:29

Question 1:

To calculate how long the survey took, we need to get the first and last dates (which functions can find the largest/smallest value), and subtract them. Remember with DateTime data, we can't just subtract the values. We have to use a function to get the difference in days.

13:36

When I do it, I get 924 days which is about 2 and a half years!

13:42

Chidi Kunto
Online

Just imagine all the visits, meeting all those people on the ground for two years! It is sometimes easy to see data as meaningless numbers and text, but remember that each person in that queue that day could have been someone who walked 10 kilometres, queued for 4-5 hours and then walked all the way back home! Often these are children who need to do this, so they have less time to attend school. It makes me sad and angry that we got to this point!

13:53

Anyway, Mambo yatakuwa sawa as my mother used to say, 'Things will be okay'. The important thing is we're doing our part to stop that kind of thing from happening.

13:57

Question 2:

Let's see how long people have to queue on average in Maji Ndogo. Keep in mind that many sources like `taps_in_home` have no queues. These are just recorded as 0 in the `time_in_queue` column, so when we calculate averages, we need to exclude those rows. Try using `NULLIF()` do to this.

13:59

You should get a queue time of about 123 min. So on average, people take two hours to fetch water if they don't have a tap in their homes.

14:02

That may sound reasonable, but some days might have more people who need water, and only have time to go and collect some on certain days.

14:15

Question 3:

So let's look at the queue times aggregated across the different days of the week.

14:17

`DAY()` gives you the day of the month. It we want to aggregate data for each day of the week, we need to use another DateTime function, `DAYNAME(column)`. As the name suggests, it returns the day from a timestamp as a string. Using that on the `time_of_record` column will result in a column with day names, Monday, Tuesday, etc., from the timestamp.

14:17

To do this, we need to calculate the average queue time, grouped by day of the week. Remember to revise DateTime functions, and also think about how to present the results clearly.

14:19

I got this:

day_of_week	avg_queue_time
Friday	120
Saturday	246
Sunday	82
Monday	137
Tuesday	108
Wednesday	97
Thursday	105

14:21

Wow, ok Saturdays have much longer queue times compared to the other days!

14:31

Question 4:

We can also look at what time during the day people collect water. Try to order the results in a meaningful way.

14:36

I get something like this:

hour_of_day	avg_queue_time
6	149
7	149
8	149
...	...

14:43

I don't know about you, but the hour number is difficult to interpret. A format like 06:00 will be easier to read, so let's use that.

14:50

I'll help you with this one. To format time into a specific display format, we can use `TIME_FORMAT(time, format)`. It takes a `time` data field and converts it into a format like `%H:00` which is easy to read. `HOUR(time_of_record)` gives us an integer value of the hour of the day, that won't work with `TIME_FORMAT()`, so we need to use `TIME(time_of_record)` instead.

14:54

Chidi Kunto
Online

So this line:

```
...
HOUR(time_of_record) AS hour_of_day
...
```

in the previous query should become:

```
...
TIME_FORMAT(TIME(time_of_record), '%H:00') AS hour_of_day
...
```

15:00

Try it, and now look at the format! This is much easier for someone to interpret and helps us to tell the story better.

15:05

I get something like this:

hour_of_day	avg_queue_time
06:00	149
07:00	149
08:00	149
...	...

15:08

Chidi Kunto
Online

Can you see that mornings and evenings are the busiest? It looks like people collect water before and after work. Wouldn't it be nice to break down the queue times for each hour of each day? In a spreadsheet, we can just create a pivot table.

Pivot tables are not widely used in SQL, despite being useful for interpreting results. So there are no built-in functions to do this for us. Sometimes the dataset is just so massive that it is the only option.

15:10

For rows, we will use the hour of the day in that nice format, and then make each column a different day!

15:25

To filter a row we use `WHERE`, but using `CASE()` in `SELECT` can filter columns. We can use a `CASE()` function for each day to separate the queue time column into a column for each day. Let's begin by only focusing on Sunday. So, when a row's `DAYNAME(time_of_record)` is `Sunday`, we make that value equal to `time_in_queue`, and `NULL` for any days.

15:37

If you run this query you will see what I mean.

```sql
SELECT
    TIME_FORMAT(TIME(time_of_record), '%H:00') AS hour_of_day,
    DAYNAME(time_of_record),
    CASE
        WHEN DAYNAME(time_of_record) = 'Sunday' THEN time_in_queue
        ELSE NULL
    END AS Sunday
FROM
    visits
WHERE
    time_in_queue != 0; -- this exludes other sources with 0 queue times.
```

15:44

Where the day name is Sunday, there are queue time values, and all other rows are null. So now if we aggregate that column, we will use all of the values that are not null.

15:44

By adding `AVG()` around the `CASE()` function, we calculate the average, but since all of the other days' values are 0, we get an average for Sunday only, rounded to 0 decimals. To aggregate by the hour, we can group the data by `hour_of_day`, and to make the table chronological, we also order by `hour_of_day`.

15:51

Chidi Kunto
Online

This is the form of the query we will use:

```sql
SELECT
    TIME_FORMAT(TIME(time_of_record), '%H:00') AS hour_of_day,

    -- Sunday
    ROUND(AVG(
        CASE
        WHEN DAYNAME(time_of_record) = 'Sunday' THEN time_in_queue
        ELSE NULL
    END
        ),0) AS Sunday,

    -- Monday
    ROUND(AVG(
        CASE
        WHEN DAYNAME(time_of_record) = 'Monday' THEN time_in_queue
        ELSE NULL
    END
        ),0) AS Monday

    -- Tuesday
    -- Wednesday

FROM
    visits
WHERE
    time_in_queue != 0 -- this excludes other sources with 0 queue times
GROUP BY
    hour_of_day
ORDER BY
    hour_of_day;
```

16:26

We create separate columns for each day with a CASE() function.

16:24

Ok, so here's your challenge: Fill out the query for the rest of the days, and run it. Make sure to specify the day in the CASE() function, and the alias.

16:28

You should have 8 columns with average queue times for each hour in each day!

hour_of_day	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
06:00	79	190	134	112	134	153	247
07:00	82	186	128	111	139	156	247
08:00	86	183	130	119	129	153	247
09:00	84	127	105	94	99	107	252
10:00	83	119	99	89	95	112	259
...	...	...	...	...	...	...	...

16:31

Now we can compare the queue times for each day, hour by hour!

16:33

Chidi Kunto
Online

See if you can spot these patterns:

1. Queues are very long on a Monday morning and Monday evening as people rush to get water.
2. Wednesday has the lowest queue times, but long queues on Wednesday evening.
3. People have to queue pretty much twice as long on Saturdays compared to the weekdays. It looks like people spend their Saturdays queueing for water, perhaps for the week's supply?
4. The shortest queues are on Sundays, and this is a cultural thing. The people of Maji Ndogo prioritise family and religion, so Sundays are spent with family and friends.

16:36

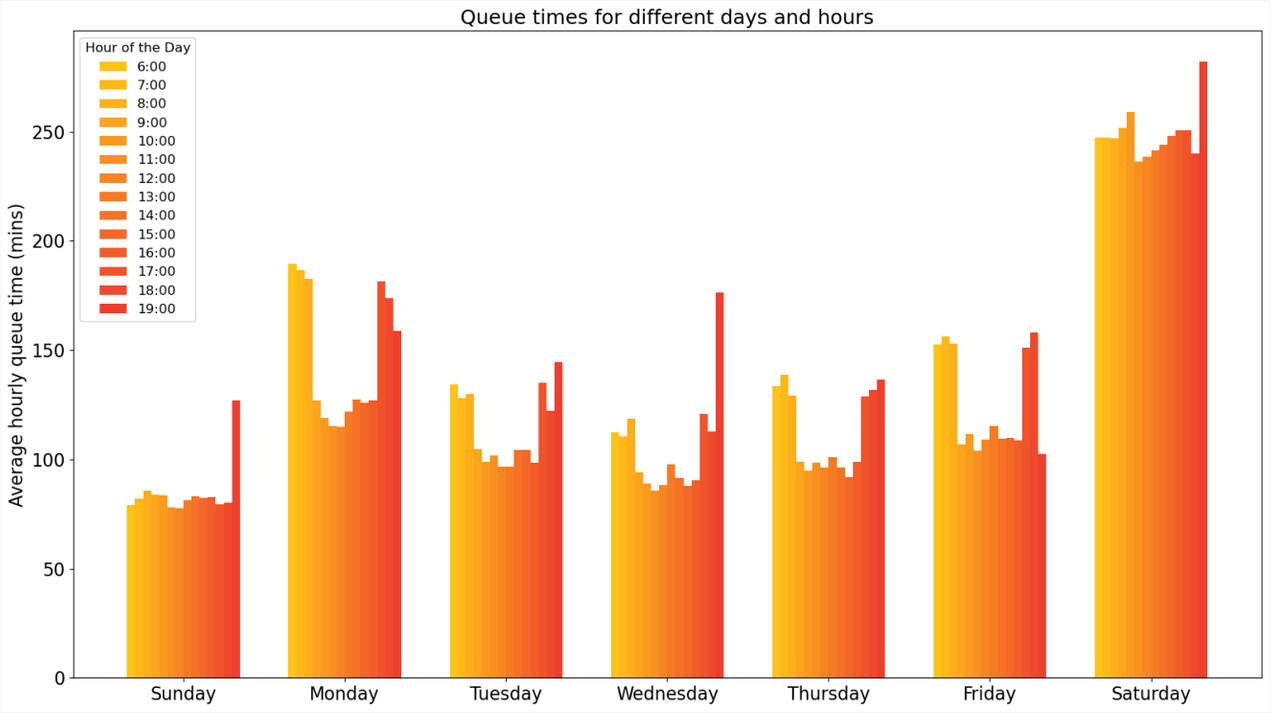
We built a pivot table in SQL! The thing I want you to remember today is: SQL is a set of tools we can apply. By understanding CASE, we could build a complex query that aggregates our data in a format that is very easy to understand.

16:37

To take it one step further, I made a graph! If you copy the pivot table into a spreadsheet, you can too.

16:37

The colors represent the hours of the day, and each bar is the average queue time, for that specific hour and day.



10:39

What do you think about this chart? As we consider presenting this to President Naledi, think about how we can use our findings to tell the story and bring focus to the patterns in the queue times.

16:41

Water Accessibility and infrastructure summary report

This survey aimed to identify the water sources people use and determine both the total and average number of users for each source. Additionally, it examined the duration citizens typically spend in queues to access water.
So let's create a short summary report we can send off to Pres. Naledi:

16:51

Insights

1. Most water sources are rural.
2. 43% of our people are using shared taps. 2000 people often share one tap.
3. 31% of our population has water infrastructure in their homes, but within that group, 45% face non-functional systems due to issues with pipes, pumps, and reservoirs.
4. 18% of our people are using wells of which, but within that, only 28% are clean..
5. Our citizens often face long wait times for water, averaging more than 120 minutes.
6. In terms of queues:

- Queues are very long on Saturdays.
- Queues are longer in the mornings and evenings.
- Wednesdays and Sundays have the shortest queues.

16:58

Start of our plan

We have started thinking about a plan:

1. We want to focus our efforts on improving the water sources that affect the most people.
- Most people will benefit if we improve the shared taps first.
- Wells are a good source of water, but many are contaminated. Fixing this will benefit a lot of people.
- Fixing existing infrastructure will help many people. If they have running water again, they won't have to queue, thereby shorting queue times for others. So we can solve two problems at once.
- Installing taps in homes will stretch our resources too thin, so for now, if the queue times are low, we won't improve that source.
2. Most water sources are in rural areas. We need to ensure our teams know this as this means they will have to make these repairs/upgrades in rural areas where road conditions, supplies, and labour are harder challenges to overcome.

17:04

Practical solutions

1. If communities are using **rivers**, we can dispatch trucks to those regions to provide water temporarily in the short term, while we send out crews to drill for wells, providing a more permanent solution.
2. If communities are using **wells**, we can install filters to purify the water. For wells with **biological** contamination, we can **install UV filters** that kill microorganisms, and for *polluted wells*, we can install **reverse osmosis** filters. In the long term, we need to figure out why these sources are polluted.
3. For **shared taps**, in the short term, we can send additional water tankers to the busiest taps, on the busiest days. We can use the queue time pivot table we made to send tankers at the busiest times. Meanwhile, we can start the work on **installing extra taps** where they are needed. According to UN standards, the maximum acceptable wait time for water is 30 minutes. With this in mind, our aim is to **install taps** to get **queue times below 30 min**.
4. **Shared taps** with **short queue times** (< 30 min) represent a logistical challenge to further reduce waiting times. The most effective solution, installing taps in homes, is resource-intensive and better suited as a long-term goal.
5. **Addressing broken infrastructure** offers a significant impact even with just a single intervention. It is expensive to fix, but so **many people** can **benefit** from repairing one facility. For example, fixing a reservoir or pipe that multiple taps are connected to. We will have to find the commonly affected areas though to see where the problem actually is.

17:10

Think these through a bit and in the meantime I'll send out some emails to get estimates of the cost to repair or improve each of these sources.

17:14