Integrated Project: Maji Ndogo Part 2

Clustering data to unveil Maji Ndogo's water crisis

Cleaning our data

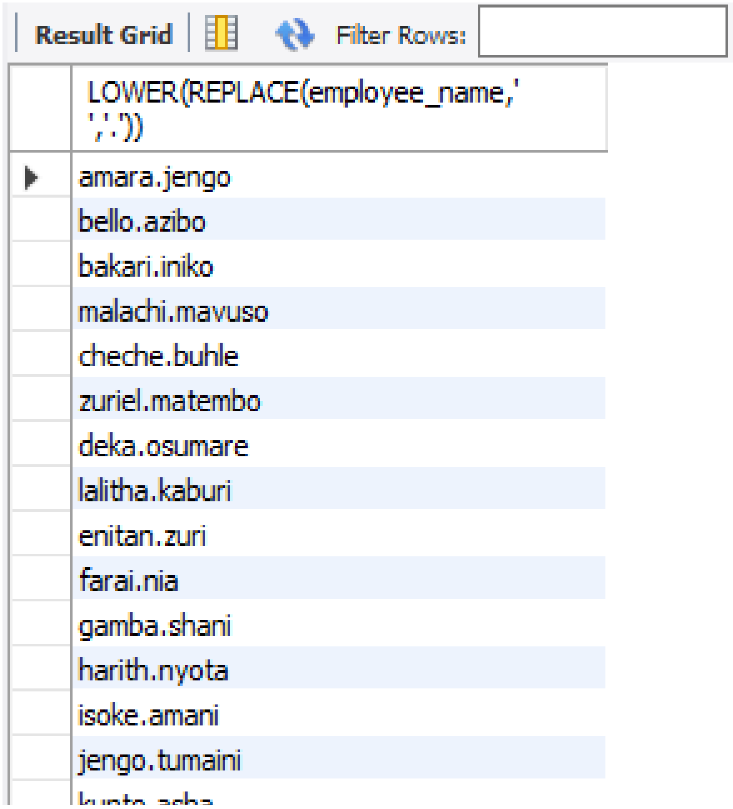
Ok, bring up the employee table. It has info on all of our workers, but note that the email addresses have not been added. We will have to send them reports and figures, so let's update it. Luckily the emails for our department are easy: first_name.last_name@ndogowater.gov.

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

- Replace the space with a full stop AND Make it all lowercase PAGE (6)

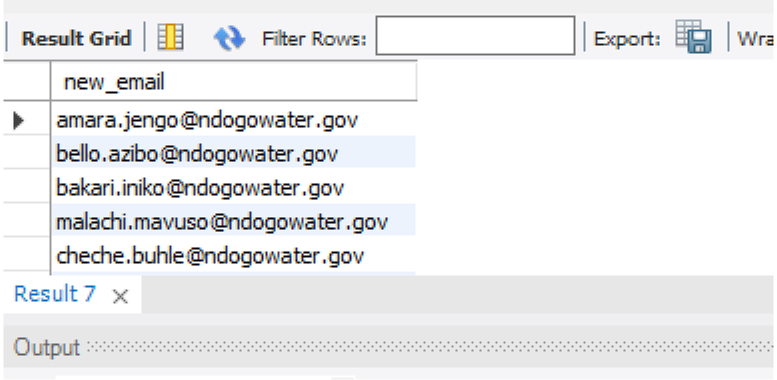
```
SELECT

      LOWER(REPLACE(employee_name,' ', '.'))

FROM   employee;
```



```
LOWER(REPLACE(employee_name,'
',' . '))
amara.jengo
bello.azibo
bakari.iniko
malachi.mavuso
cheche.buhle
zuriel.matembo
deka.osumare
lalitha.kaburi
enitan.zuri
farai.nia
gamba.shani
harith.nyota
isoke.amani
jengo.tumaini
kunto.asha
```

- Replace the space with a full stop AND Make it all lower case,
- and use CONCAT() '@ndogowater.gov') AS new_email –– add it all together   PAGE (6)

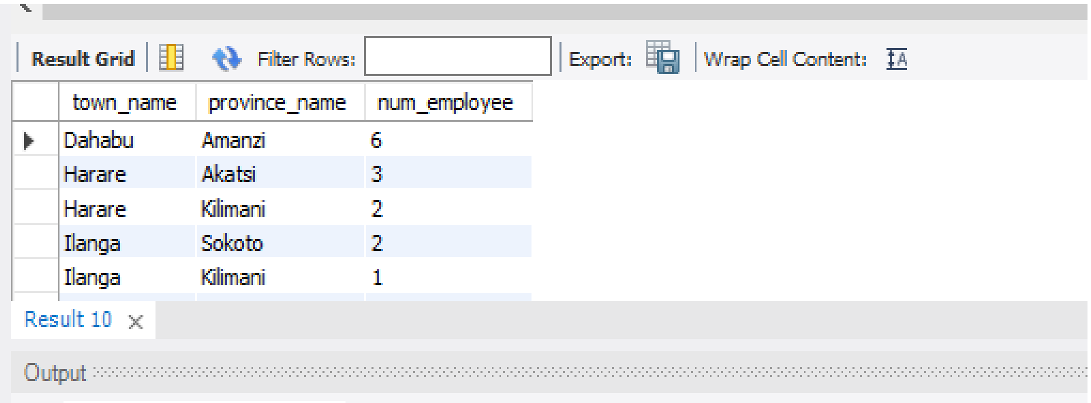
```
SELECT

    CONCAT(

        LOWER(REPLACE(employee_name,' ',  '.')), '@ndogowater.gov') AS new_email

FROM employee;
```



new_email
amara.jengo@ndogowater.gov
bello.azibo@ndogowater.gov
bakari.iniko@ndogowater.gov
malachi.mavuso@ndogowater.gov
cheche.buhle@ndogowater.gov

- 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.PAGE (9)

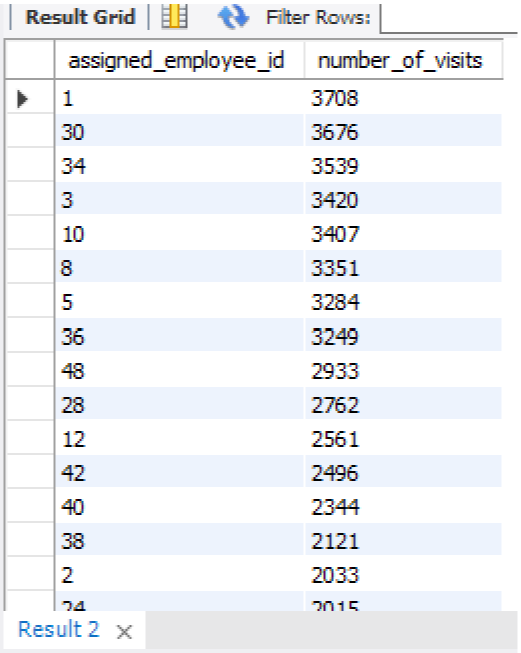
```
SELECT DISTINCT
    province_name,
    town_name,
    COUNT(town_name)  AS num_employees
 FROM employee
GROUP BY
 1,2
ORDER BY
   1,3  DESC;
```



town_name	province_name	num_employee
Dahabu	Amanzi	6
Harare	Akatsi	3
Harare	Kilimani	2
Ilanga	Sokoto	2
Ilanga	Kilimani	1

- 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 PAGE (10)

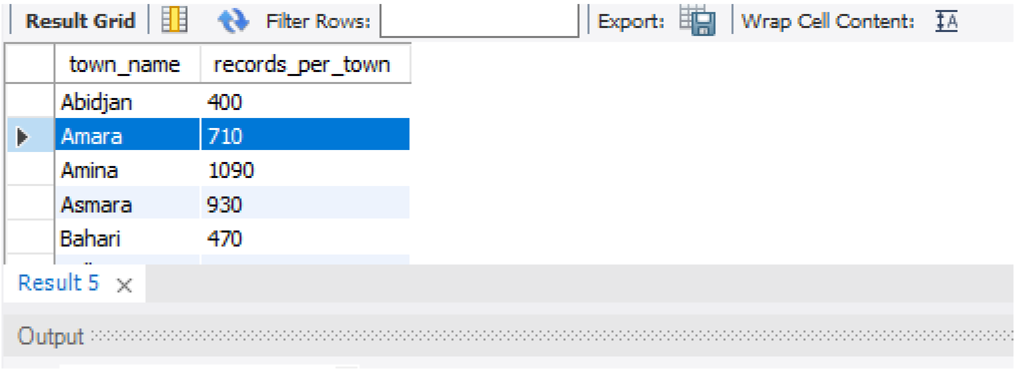
```
SELECT

    assigned_employee_id,

    COUNT(visit_count) AS number_of_visits

FROM

visits

GROUP BY

      assigned_employee_id

ORDER BY

     number_of_visits DESC
```



assigned_employee_id	number_of_visits
1	3708
30	3676
34	3539
3	3420
10	3407
8	3351
5	3284
36	3249
48	2933
28	2762
12	2561
42	2496
40	2344
38	2121
2	2033
24	2015

- Create a query that counts the number of records per town PAGE (11)

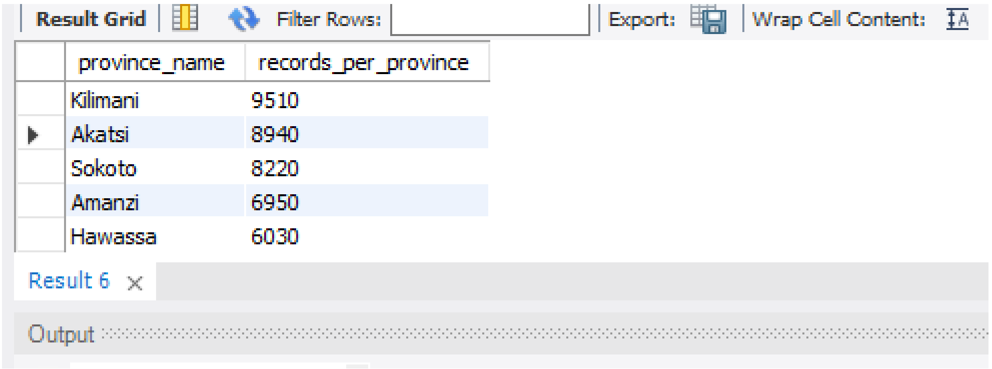
```
SELECT DISTINCT

   town_name,

   COUNT(town_name) OVER (PARTITION BY town_name) AS records_per_town

FROM

  Location;
```



town_name	records_per_town
Abidjan	400
Amara	710
Amina	1090
Asmara	930
Bahari	470

- Now count the records per province. PAGE (12)

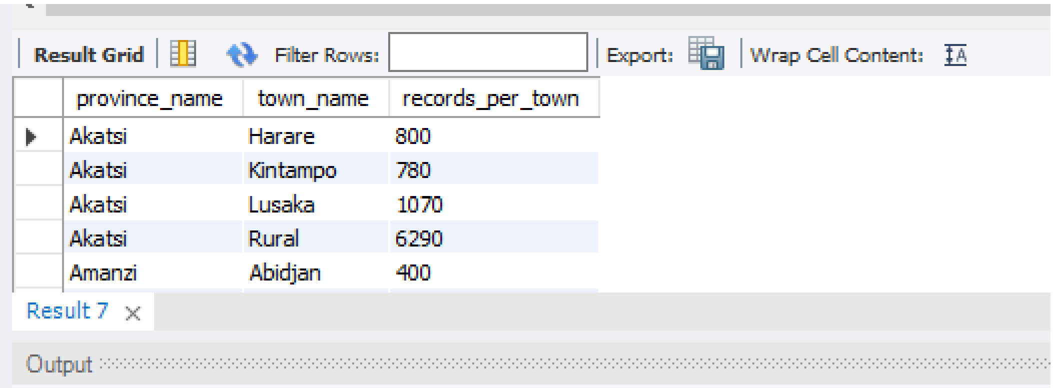
```
SELECT DISTINCT

   province_name,

   COUNT(province_name) OVER (PARTITION BY province_name) AS
records_per_province

FROM

  location;
```



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

- 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.

```
SELECT DISTINCT

    province_name,

    town_name,

    COUNT(town_name) AS records_per_town

FROM

  location

GROUP BY

    province_name,

    town_name

ORDER BY province_name;
```

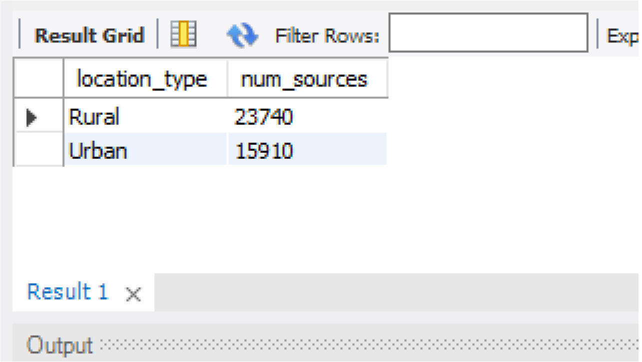


province_name	town_name	records_per_town
Akatsi	Harare	800
Akatsi	Kintampo	780
Akatsi	Lusaka	1070
Akatsi	Rural	6290
Amanzi	Abidjan	400

- Finally, look at the number of records for each location type   PAGE (13)

```
SELECT DISTINCT

   location_type,

   COUNT(location_type) OVER (PARTITION BY location_type) AS num_sources
```

```
FROM

 location;
```

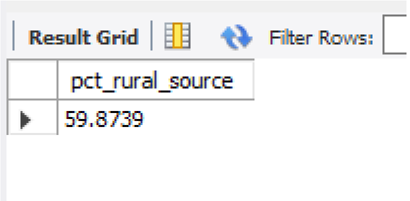


location_type	num_sources
Rural	23740
Urban	15910

- 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 AS pct_rural_source
```

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



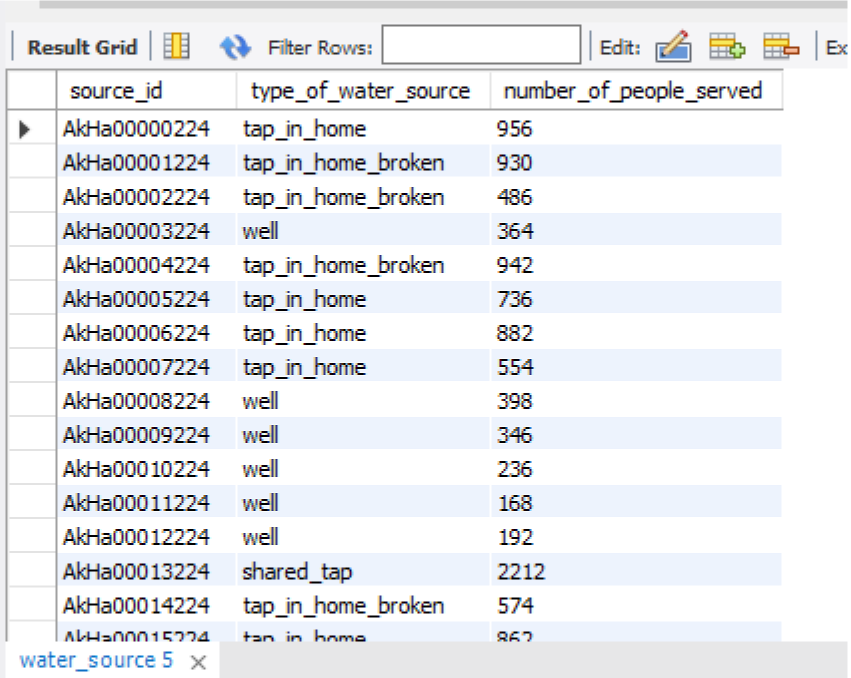
pct_rural_source
59.8739

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

```
SELECT  * FROM

  water_source
```

LIMIT 50;



source_id	type_of_water_source	number_of_people_served
AkHa00000224	tap_in_home	956
AkHa00001224	tap_in_home_broken	930
AkHa00002224	tap_in_home_broken	486
AkHa00003224	well	364
AkHa00004224	tap_in_home_broken	942
AkHa00005224	tap_in_home	736
AkHa00006224	tap_in_home	882
AkHa00007224	tap_in_home	554
AkHa00008224	well	398
AkHa00009224	well	346
AkHa00010224	well	236
AkHa00011224	well	168
AkHa00012224	well	192
AkHa00013224	shared_tap	2212
AkHa00014224	tap_in_home_broken	574
AkHa00015224	tap_in_home	862

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?

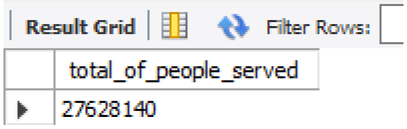
1. How many people did we survey in total?

SELECT

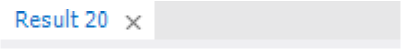
SUM(number_of_people_served) AS SURVEY_IN_TOTAL -- > ***EQUAL = '27628140'***

FROM

water_source;



total_of_people_served
27628140



2. How many wells, taps and rivers are there? PAGE(15)

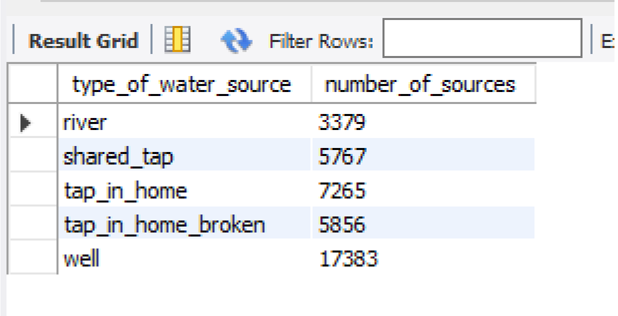
SELECT DISTINCT

type_of_water_source,

COUNT(type_of_water_source) OVER (PARTITION BY type_of_water_source) AS number_of_sources

FROM

water_source;



type_of_water_source	number_of_sources
river	3379
shared_tap	5767
tap_in_home	7265
tap_in_home_broken	5856
well	17383

3. How many people share particular types of water sources on average? PAGE(16)

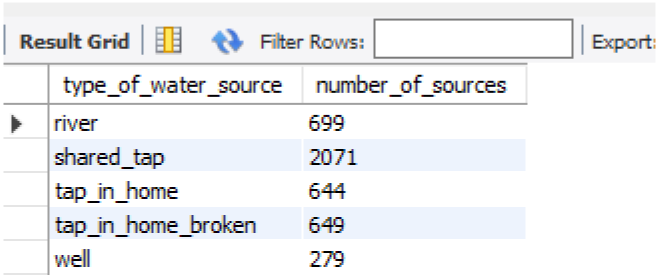
SELECT DISTINCT

type_of_water_source,

ROUND(AVG(number_of_people_served) OVER (PARTITION BY type_of_water_source)) AS number_of_sources

FROM

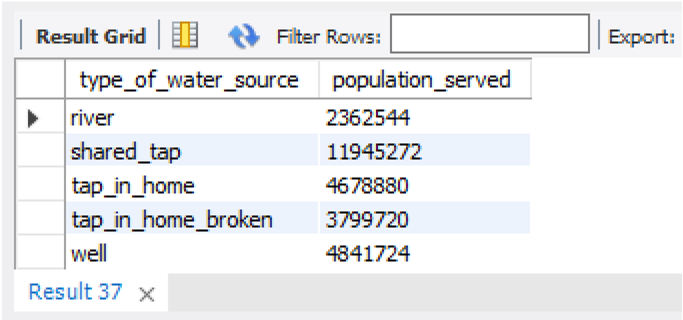
water_source;



type_of_water_source	number_of_sources
river	699
shared_tap	2071
tap_in_home	644
tap_in_home_broken	649
well	279

- 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.   PAGE(19)

```
SELECT DISTINCT

    type_of_water_source,

    ROUND(SUM(number_of_people_served) OVER (PARTITION BY
type_of_water_source)) AS population_served

FROM

   water_source;
```



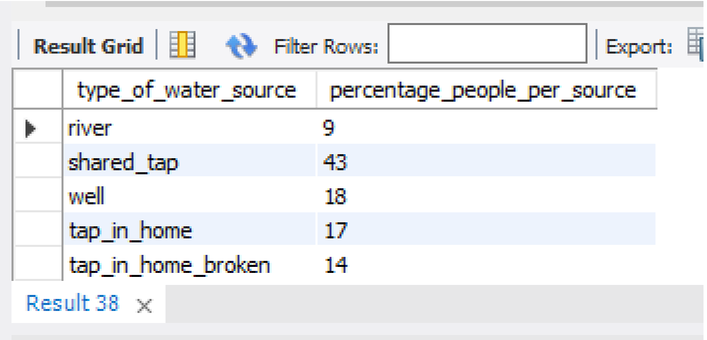
type_of_water_source	population_served
river	2362544
shared_tap	11945272
tap_in_home	4678880
tap_in_home_broken	3799720
well	4841724

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

Let's round that off to 0 decimals, and order the results.   PAGE(21)

```
SELECT DISTINCT

    type_of_water_source,

    FORMAT((SUM(number_of_people_served)/27628140)*100,0) AS
percentage_people_per_source

FROM
```

```
  water_source
GROUP BY
    type_of_water_source
ORDER BY
     percentage_people_per_source DESC;
```

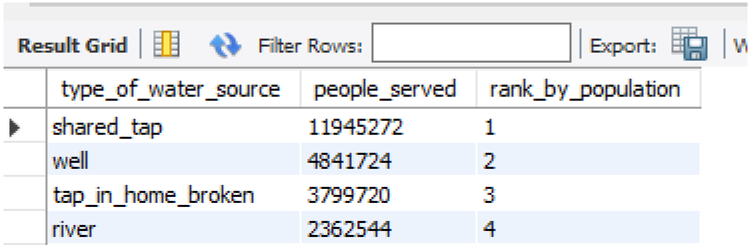


type_of_water_source	percentage_people_per_source
river	9
shared_tap	43
well	18
tap_in_home	17
tap_in_home_broken	14

So use a window function on the total people served column, converting it into a rank. PAGE(23)

- 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.

```
SELECT DISTINCT
   type_of_water_source,
   SUM(number_of_people_served) AS people_served,
   RANK() OVER (ORDER BY SUM(number_of_people_served) DESC ) AS
rank_by_population
FROM
  water_source
WHERE
   type_of_water_source != 'tap_in_home'
GROUP BY
    type_of_water_source
LIMIT 50;
```

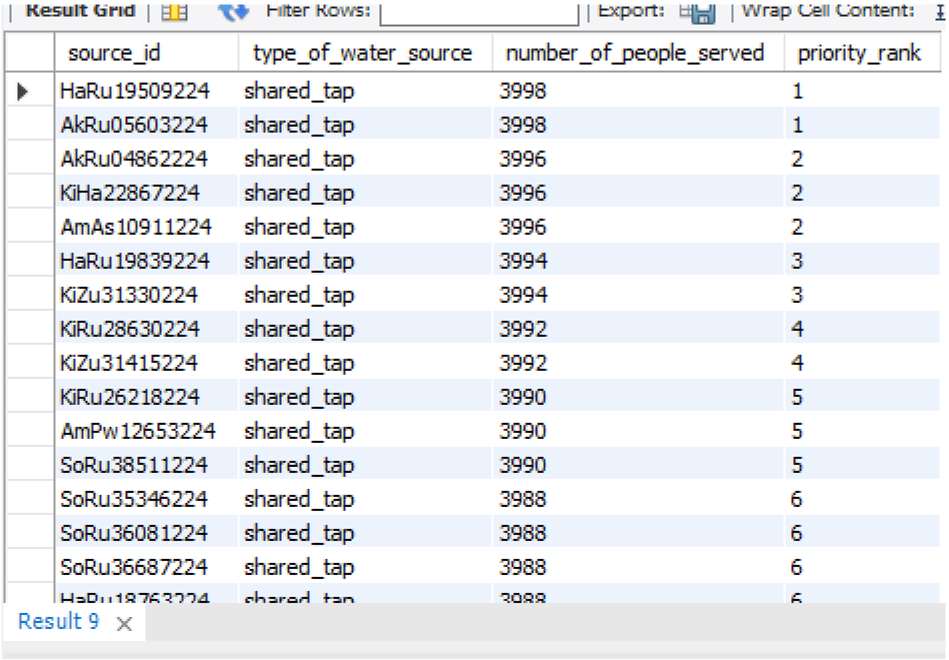


type_of_water_source	people_served	rank_by_population
shared_tap	11945272	1
well	4841724	2
tap_in_home_broken	3799720	3
river	2362544	4

So create a query to do this, and keep these requirements in mind: PAGE(24)

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.

```
SELECT DISTINCT

    source_id,

    type_of_water_source,

    number_of_people_served,

    DENSE_RANK() OVER (ORDER BY number_of_people_served DESC ) AS priority_rank

FROM

  water_source

WHERE

   type_of_water_source != 'tap_in_home'

LIMIT 50;
```



source_id	type_of_water_source	number_of_people_served	priority_rank
HaRu19509224	shared_tap	3998	1
AkRu05603224	shared_tap	3998	1
AkRu04862224	shared_tap	3996	2
KiHa22867224	shared_tap	3996	2
AmAs10911224	shared_tap	3996	2
HaRu19839224	shared_tap	3994	3
KiZu31330224	shared_tap	3994	3
KiRu28630224	shared_tap	3992	4
KiZu31415224	shared_tap	3992	4
KiRu26218224	shared_tap	3990	5
AmPw12653224	shared_tap	3990	5
SoRu38511224	shared_tap	3990	5
SoRu35346224	shared_tap	3988	6
SoRu36081224	shared_tap	3988	6
SoRu36687224	shared_tap	3988	6
HaRu18763224	shared_tap	3988	6

Analysing queues

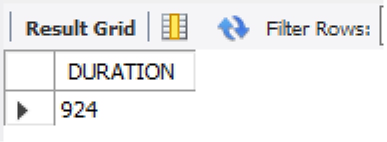
Ok, these are some of the things I think are worth looking at:

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?

1. Question 1: PAGE(27)

- 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.

```
SELECT

  TIMESTAMPDIFF(DAY,MIN(time_of_record),MAX(time_of_record)) AS DURATION

FROM

  visits;
```



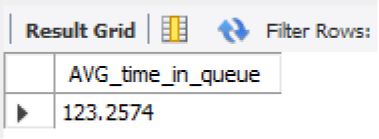
DURATION
924

2. Question 2: PAGE(28)

- 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() to do this.

```
SELECT
   AVG(NULLIF(time_in_queue,0)) AS AVG_time_in_queue
FROM
  visits;
```

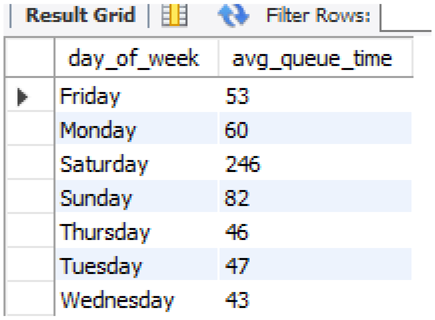
□ You should get a queue time of about 123 min. So on average,
□ People take two hours to fetch water if they don't have a tap in their homes.



AVG_time_in_queue
123.2574

So let's look at the queue times aggregated across the different days of the week. PAGE(29)

```
SELECT DISTINCT
   DAYNAME(time_of_record) AS day_of_week,
   ROUND(AVG(time_in_queue)) AS avg_queue_time
FROM
  visits
GROUP BY day_of_week
ORDER BY day_of_week;
```



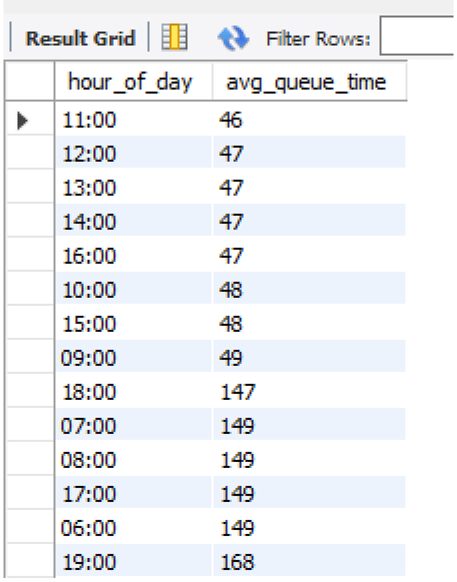
day_of_week	avg_queue_time
Friday	53
Monday	60
Saturday	246
Sunday	82
Thursday	46
Tuesday	47
Wednesday	43

4. Question 4: PAGE(31)

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

```
SELECT

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

  ROUND(AVG(time_in_queue)) AS avg_queue_time

FROM

   visits

GROUP BY

   hour_of_day

ORDER BY

    avg_queue_time;
```

- 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.

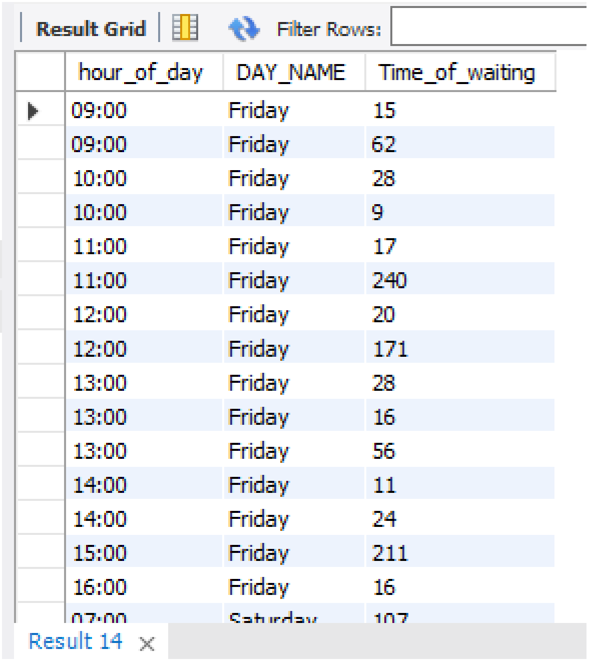


hour_of_day	avg_queue_time
11:00	46
12:00	47
13:00	47
14:00	47
16:00	47
10:00	48
15:00	48
09:00	49
18:00	147
07:00	149
08:00	149
17:00	149
06:00	149
19:00	168

```
SELECT
    TIME_FORMAT(TIME(time_of_record), '%H:00') AS hour_of_day,
    DAYNAME(time_of_record) AS DAY_NAME,
CASE
         WHEN DAYNAME(time_of_record) = 'Sunday'
        THEN time_in_queue
                 WHEN DAYNAME(time_of_record) = 'Monday'
        THEN time_in_queue
                 WHEN DAYNAME(time_of_record) = 'Saturday'
        THEN time_in_queue
                 WHEN DAYNAME(time_of_record) = 'Tuesday'
        THEN time_in_queue
                 WHEN DAYNAME(time_of_record) = 'Wednesday'
        THEN time_in_queue
                 WHEN DAYNAME(time_of_record) = 'Thursday'
        THEN time_in_queue
                 WHEN DAYNAME(time_of_record) = 'Friday'
        THEN time_in_queue
    ELSE NULL
END AS Time_of_waiting
FROM
    mdd_water_services.visits
WHERE
    time_in_queue != 0
```

- This excludes other sources with 0 queue times.

```
LIMIT 50;
```

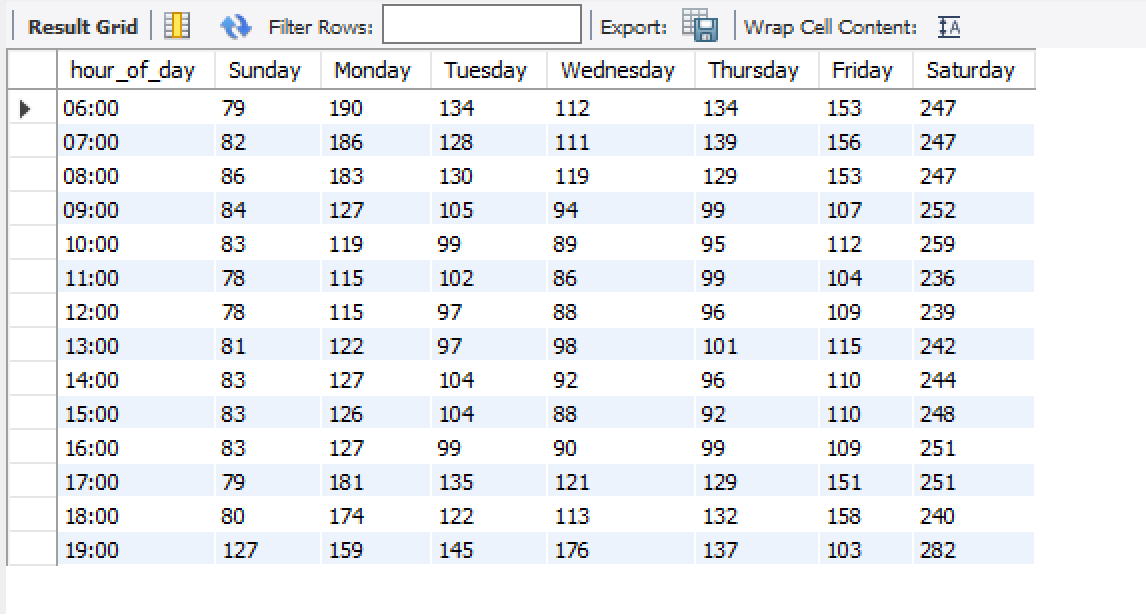


hour_of_day	DAY_NAME	Time_of_waiting
09:00	Friday	15
09:00	Friday	62
10:00	Friday	28
10:00	Friday	9
11:00	Friday	17
11:00	Friday	240
12:00	Friday	20
12:00	Friday	171
13:00	Friday	28
13:00	Friday	16
13:00	Friday	56
14:00	Friday	11
14:00	Friday	24
15:00	Friday	211
16:00	Friday	16
07:00	Saturday	107

- **Creating a pivot table of time waiting in each day PAGE(35)**

```
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
ROUND(AVG(
        CASE
    WHEN DAYNAME(time_of_record) = 'Tuesday'
       THEN time_in_queue
    ELSE NULL
        END),0) AS Tuesday,
-- Wednesday
ROUND(AVG(
        CASE
    WHEN DAYNAME(time_of_record) = 'Wednesday'
       THEN time_in_queue
    ELSE NULL
        END),0) AS Wednesday,
-- Thursday
ROUND(AVG(
        CASE
    WHEN DAYNAME(time_of_record) = 'Thursday'
       THEN time_in_queue
    ELSE NULL
        END),0) AS Thursday,
-- Friday
ROUND(AVG(
        CASE
    WHEN DAYNAME(time_of_record) = 'Friday'
       THEN time_in_queue
    ELSE NULL
        END),0) AS Friday,
-- Saturday
ROUND(AVG(
        CASE
    WHEN DAYNAME(time_of_record) = 'Saturday'
       THEN time_in_queue
    ELSE NULL
        END),0) AS Saturday
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;
```



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
11:00	78	115	102	86	99	104	236
12:00	78	115	97	88	96	109	239
13:00	81	122	97	98	101	115	242
14:00	83	127	104	92	96	110	244
15:00	83	126	104	88	92	110	248
16:00	83	127	99	90	99	109	251
17:00	79	181	135	121	129	151	251
18:00	80	174	122	113	132	158	240
19:00	127	159	145	176	137	103	282