CPT103 Coursework Report

Must Read

Your report and database design must be your own work. You should not copy any code from others or let anyone develop this PMMS. ***Plagiarism and collusion lead to a zero mark for this coursework***.

All tables must be in 3NF and the ER diagram should not contain any M: N relationships. Please strictly follow the structure of this template. Remember to double-check grammar and wording errors so that the report could be understood easily. Failing to do so will result in mark deductions. ***Any language other than English will be ignored when marking the report***.

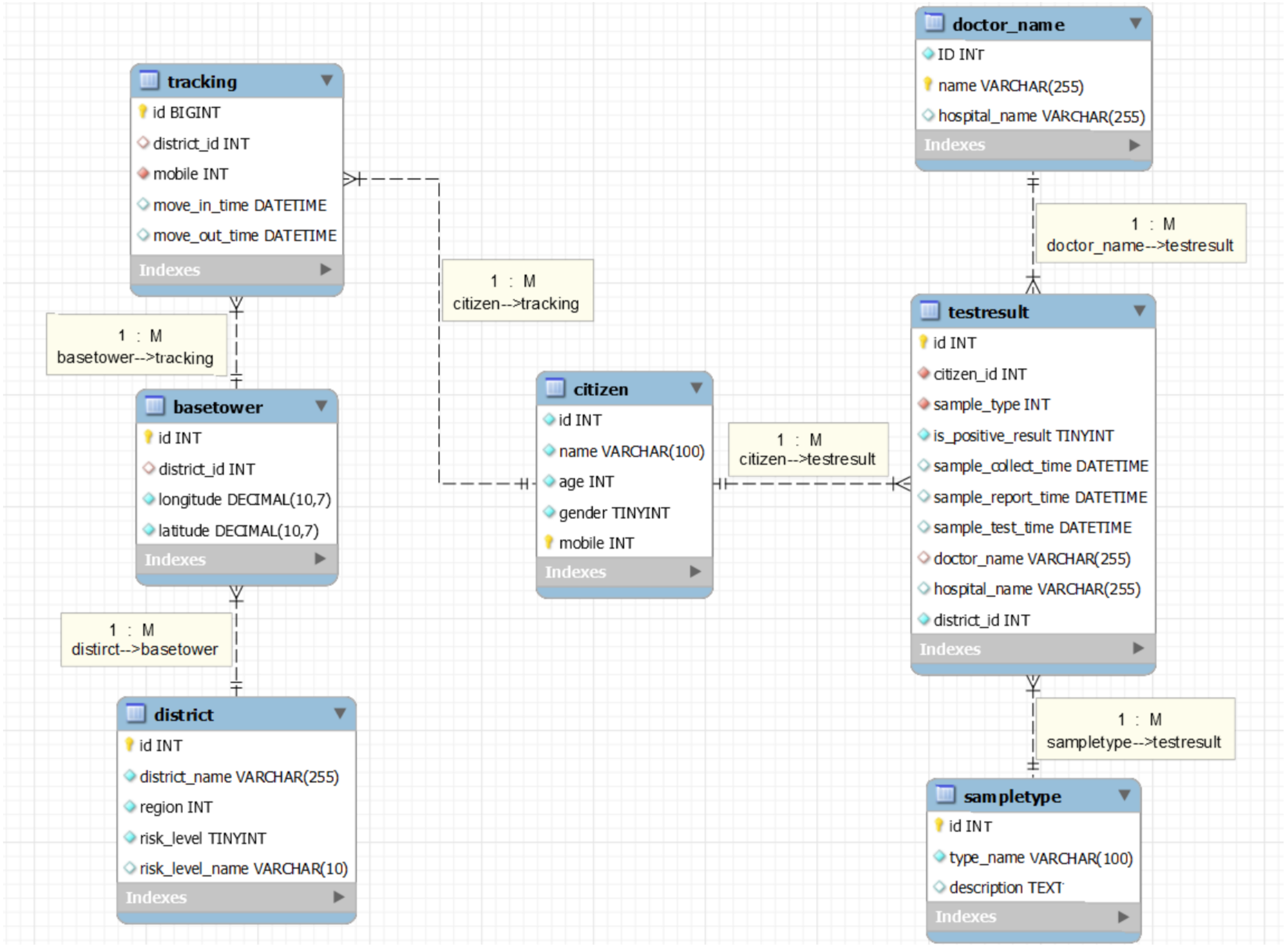
Your ID: 2039153

Name: Changqing.Lin

Email Address: Changqing.Lin20@student.xjtlu.edu.cn

Your ER Diagram here.

Make sure this ER diagram fits one single page and is clear to read. Do not split the diagram into several pages.



Database Design Details

Tables

In this section, you are required to explain all of the tables in your ER diagram. An example is given below. Please make sure you follow the template and everything is clearly explained.

------ The Beginning of the Example ------

[This is only an example. You don't have to follow the same way I explained this table, but your explanation should be clear and detailed]

Table name: Staff

Table design general explanation:

The staff table is used to store the information of all staff members in the company. The primary key is staff_id as it is unique for every staff member. *[Add more explanations if you made some special considerations. Try to support your assumptions with proofs in real life]*

Attributes:

Column Definition	Domain	Explanation
staff_age int	18+	The age of staff members. The value should be larger than 18 because of the government law. This domain is checked by the domain constraint called XXXX.
address varchar(255)	Room *XX*, Building *YY*, *ZZ* Road, *WW* district	The address of the office, the data cannot be checked by the database directly. As a result, manual checking is required when entering data.
Staff_name varchar(100)	All valid names are acceptable. For example, 'Jun Qi'.	The name of staff members.
Branch_no char(4)	Branch numbers start with B followed by 3 digits. For example, 'b003'	The branch number where this staff works in
Staff_email varchar(255)	Valid email addresses XXX@YYY.ZZZ	The correctness of email addresses cannot be checked directly by the database, so manual checking is required. This column is not UNIQUE. It is an intentional design because the company sometimes offer shared email addresses for one office to improve communications.
Staff_id int primary key	2003001	The staff id must be 7 numbers long and follow the format XXXYYYY, where XXX refers to the branch of that staff and YYYY is the individual unique number of that staff.

Foreign keys:

The column Branch_no r sting branches of the company. It corresponds to the XXX relationship in the ER diagram.

------ The End of the Example ------

Table name: Citizen

Table design explanation:

The citizen table is used to store the information of all citizen in The Lukewarm Kingdom. The primary key is mobile as it is unique for every citizen.

Column Definition	Domain	Explanation
Id int unique key	0+	Every citizen has only one id number for themselves, this id number begin from 0.
name varchar(100)	All kinds of name are acceptable, like "AA, AB" are all fine.	The name of citizen.
age int	0+	The age of citizen must be equal or greater than 0.
gender tinyint	0 or 1	0=female, 1=male
mobile int(11) Primary key	XXXXXX	Each citizens has a unique mobile number.

Table name: sampletype

Table design explanation:

The sampletype table is used to store the information of all kinds of virus that named by the Word Health Organization. The primary key is id as it is unique for every sample type of virus.

Column Definition	Domain	Explanation
id int primary key	0+	The id number of types of viruses.
typename varchar(100)	All kinds of virus named by the Word Health Organization.	The name of virus.
description text	Including the name of virus, the statement of virus, and recommendation to people who get sick.	The detail information to remind people.

Table name: doctor_name

Table design explanation:

The doctor_name table is used to store the information of all doctor's signature and which hospital they are working for in The Lukewarm Kingdom. In order to avoid the data read errors, supposed that there is no doctor with the same name, therefore each doctor's signature on the test result will not repetition. The primary key doctor_name is unique for every doctor in The Lukewarm Kingdom.

Column Definition	Domain	Explanation
ID INT unique key	0+	The id number is unique for each doctor
name varchar(255) primary key	All kinds of name are acceptable, like "ZA, ZB" are all fine.	The signature of each doctor.

hospital_name varchar(100)	XXhospital	There are 12 hospital in The Lukewarm Kingdom, name as 1sthospital, 2ndhospotal…

Table name: testresult

Table design explanation:

The testresult table is used to collect each test result from citizens, for each citizen could take the same or different test many times, therefore should be separate with the information of citizen. The primary key is id as it is unique attributes for every test result in The Lukewarm Kingdom among different hospital.

Column Definition	Domain	Explanation
id int primary key	0+	Using to identify each test result from others
citizen_id int	0+	Every citizen has only one id number for themselves, this id number begin from 0.
sample_type int	0+	The id number of types of viruses.
is_positive_result tinyint	0 or 1	0=negative, 1=positive
sample_collect_time datetime	YYYY-MM-DD HH:MM:SS	The time citizen take test.
sample_test_time datetime	YYYY-MM-DD HH:MM:SS	The time hospital starts to test.
sample_report_time datetime	YYYY-MM-DD HH:MM:SS	The time test result published.
doctor_name varchar(100)	All kinds of name are acceptable, like “ZA, ZB” are all fine.	The name of doctors.
district_id INT		

Foreign keys and reasons:

```
 CONSTRAINT FOREIGN KEY (citizen_id) REFERENCES citizen (id),
CONSTRAINT FOREIGN KEY (sample_type) REFERENCES sampletype (id)
ON UPDATE CASCADE
ON DELETE CASCADE
CONSTRAINT FOREIGN KEY (doctor_name) REFERENCES doctor_name (`name`)
ON UPDATE SET NULL
ON DELETE SET NULL
```

1. For the first foreign key “testresult_ibfk_1”, the table testresult has M:1 relationship with table citizen, because each may take the same or different test many times, therefore should be separate with the detail information of each citizen, and use the foreign key to connect from the table citizen.

The mode of ON UPDATE CASCADE and the mode ON DELETE CASCADE is design for the condition that if the citizen immigrant to the other country or pass away, delete the data of those people who will never live in The Lukewarm Kingdom has the benefits of reduce the load of database.

2. For the second foreign key “testresult_ibfk_2”, the table testresult has M:1 relationship with table sampletype. According to the description from coursework, the test result for each citizens might have different types of virus, and each sample type of virus is unique and has its own description, there might be more than one type of virus in The Lukewarm Kingdom.

The mode of ON UPDATE CASCADE is design for the convenience to insert new types of virus into database or change the mistakes immediately for example, the real name of Covid-19 was identified as SARS by mistake. And the mode ON DELETE is to design for delete the data of those virus will no longer to be a threat to human kinds to reduce the load of database.

3. For the third foreign key “testresult_ibfk_3”, the table testresult has M:1 relationship with table doctor_name. Because a test result may come from different hospital, each doctor can only work in one hospital, and every test result would be signed by one doctor.

The mode of ON UPDATE SET NULL and ON DELETE SET NULL is design for if the doctor leaves his position for any reasons, the data of test result for citizens should be kept and still effective.

<u>Table name</u>: district

Table design explanation:

The district is used to illustrate each district’s information. The primary key is id as it is unique for every district. Supposed that each region of The Lukewarm Kingdom have 5 district, and these district could be classify by direction and name north, west, middle, east, and south.

Column Definition	Domain	Explanation
id int primary key	From 0 to 25	Define The Lukewarm Kingdom have 25 district and each district has its own number.(More information in the following case 2 GPS format)
district_name VARCHAR(255)	X-x The GPS format:“region-district”(Defined in Case 2)	Each region has 5 distirct, and each district name by north,west,middle,east,south. Totally 25 Districts has its own name, the format to exact district is name by”region-distirct” (Detail information is in the following case 2 GPS format)
region int	From 1 to 5	Five region in The Lukewarm Kingdom, 1=North, 2=West, 3=Middle, 4=East, 5=South.
risk_level tinyint	From 1 to 3	There are 3 kinds of risk level, 1=low, 2=mid, 3=high
risk_level_name	low/mid/high	The name of risk level

<u>Table name</u>: basetower

Table design explanation:

The table basetower is used to store the information of all base stations. Supposed that The Lukewarm Kingdom is a developing country, each district only has one base station, and the signal from the base station can cover the entire area without overlapping. Supposed that The Lukewarm Kingdom located in (0. 00000°-9.00000°N) and (0. 00000°-9.00000°E)(Detail information is in the following case 2 GPS format). The primary key is id as it is unique for every base station. There are totally 25 base station in The Lukewarm Kingdom(According to the GPS format in case 2).

Column Definition	Domain	Explanation

id int primary	From 0 to 25	The id number of each base stations.
district_id int	From 0 to 25	Define The Lukewarm Kingdom have 25 district and each district has its own number.(More information in the following case 2 GPS format)
longitude DECIMAL(10,7)	ddd.ddddd °	The longitude of The Lukewarm Kingdom (0. 00000°-9.00000°E)
latitude DECIMAL(10,7)	ddd.ddddd °	The latitude of The Lukewarm Kingdom (0. 00000° -9.00000°S)

Foreign keys and reasons:

CONSTRAINT FOREIGN KEY (district_id) REFERENCES district (id)

ON UPDATE CASCADE

ON DELETE SET NULL

As for The Lukewarm Kingdom is a developing country, maybe the government will decide to make several district more than one base stations. For example, 5G base station, with smaller coverage area but higher efficiency in data transmission. Therefore, in the future maybe one district will have more than one base station. Thus, for the foreign key "basetower_ibfk_1", the table basetower may have M:1 relationship with table district in the future.

The mode of ON UPDATE CASCADE is design for convenience. For example, if the government may change existing districts' name, adding new districts or cancel existing districts, the data changing from district will directly update the data in base stations that in table basetower. And the mode of ON DELETE SET NULL is design to keep data in base station safe. For example, the government might cancel existing districts in the future, the data changing from district will not affect the data in base stations that in table basetower.

Table name: tracking

Table design explanation:

The tracking table is used to collect the information of each citizens' activity in different districts. For each citizen may move into different district in any time, therefore, the information for every tracking record is independent from citizens' mobile number. According to the requirement of courses work, for those citizens who needn't to do the virus test could only record mobile number, the district they are, the time they move into the district and the time they leave the district. **The primary key is id as it is the unique identification for every tracking information.**

Column Definition	Domain	Explanation
id bigint primary key	0+	The id number of each tracking record for everyone who live in The Lukewarm Kingdom, therefore this is a large amount of data should use the form of BIGINT.
district_id int	From 0 to 25	Define The Lukewarm Kingdom have 25 district and each district has its own number.(More information in the following case 2 GPS format)

mobile VARCHAR(6)	XXXXXX	Each citizens has a unique mobile number.
move_in_time datetime	YYYY-MM-DD HH:MM:SS	The time people move into any districts.
move_out_time datetime	YYYY-MM-DD HH:MM:SS	The time people leave any districts.

Foreign keys and reasons:

```
  CONSTRAINT FOREIGN KEY(district_id) REFERENCES basetower (district_id),
  ON UPDATE CASCADE
  ON DELETE SET NULL
CONSTRAINT FOREIGN KEY(mobile) REFERENCES citizen(mobile)
  ON UPDATE CASCADE
  ON DELETE CASCADE
```

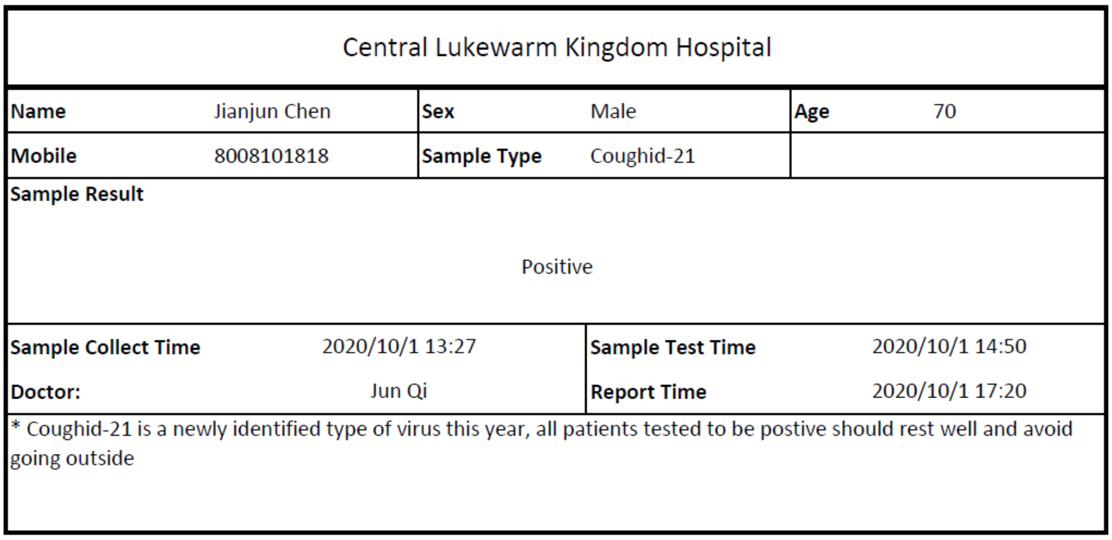
1. For the first foreign key “tracking_ibfk_1”, the table tracking has a M:1 relationship with the table basetower. As for The Lukewarm Kingdom is a developing country, maybe the government will decide to cancel existing base stations or build more than one base stations in a district. For example, 5G base station, with smaller coverage area than 4G base station, but possess higher efficiency in data transmission. Therefore, in the future maybe one district will have more than one base station.

The mode of ON UPDATE SET NULL and ON DELETE CASCADEare design for the efficiency of data update or migration. For example, if the government may changing existing base stations or adding new base stations, the data changing from tracking of previous base station will efficiently become the new one. The mode of ON DELETE SET NULL is design for save data completely. For example, if the government may cancel existing base stations, the data changing from base station will not affect the data in table tracking.

2. For the second foreign key “tracking_ibfk_2”, the table tracking has a M:1 relationship with the table citizen. Each citizen only has one mobile number, but could have thousands of tracking record.

The mode of ON UPDATE CASCADE is design for the efficiency of data update or migration. For example, if a citizen loses his mobile phone for any reasons, he has to buy a new one including the SIM card, so that might change the mobile number. Therefore, the mode of ON UPDATE CASCADE could transfer the information of that citizen efficient. And the mode of ON DELETE CASCADE is design for the situation if the citizen passes away, delete the data of those people who will never live in The Lukewarm Kingdom has the benefits of reduce the load of database.

Viral Test Report – Normalisation Process



Central Lukewarm Kingdom Hospital

Name	Jianjun Chen	**Sex**	Male	**Age**	70
Mobile	8008101818	**Sample Type**	Coughid-21		

Sample Result

Positive

Sample Collect Time	2020/10/1 13:27	**Sample Test Time**	2020/10/1 14:50
Doctor:	Jun Qi	**Report Time**	2020/10/1 17:20

* Coughid-21 is a newly identified type of virus this year, all patients tested to be postive should rest well and avoid going outside

Please write down the detailed normalisation process for the viral test report. Firstly, identify all attributes you can find in the viral test report as well as in the specifications (you need to add your own attributes if your database design has them). Then, at each normalisation stage, list all functional dependencies and normalise them to the higher normal form. 3NF is required for the final tables and must match your ER diagram.

Stage 1

Attributes:

According to the requirement of the coursework, the database should recorded the following detail information from citizens' test report: Name, Sex, Age, Mobile, Sample, Sample Result, Sample Collect Time, Sample Test Time, Report Time, Doctor's signature, Description of the Sample Type of virus, and the name of hospital.

Therefore, there are following attributes are needed:

```
CREATE TABLE `TestedCitizen`(
`id` INT AUTO_INCREMENT UNIQUE KEY,
`name` VARCHAR(100) NOT NULL,
`age` INT NOT NULL,
`gender` TINYINT NOT NULL,
`mobile` VARCHAR(6) PRIMARY KEY NOT NULL,
`sample_type` INT NOT NULL,
`type_name` VARCHAR(100) NOT NULL,
`description` TEXT,
`is_positive_result` TINYINT NOT NULL,
`sample_collect_time` DATETIME,
`sample_report_time` DATETIME,
`sample_test_time` DATETIME,
`doctor_name` VARCHAR() NOT NULL,
`hospital_name` VARCHAR(255) NOT NULL,
)
```

FDs (Indicate partial or transitive dependencies):

1. Primary Key: `mobile`

I choose `mobile` as Primary Key is because every citizen has an unique mobile phone number.

Therefore:

mobile --> id

mobile --> name

mobile --> age

mobile --> gender

mobile --> sample_type

mobile --> type_name

mobile --> description

mobile --> is_positive_result

mobile --> sample_collect_time

mobile --> sample_report_time

mobile --> sample_test_time

mobile --> doctor_name

mobile --> hospital_name

For each mobile number in table `TestedCitizen` is associated with exactly one value of other attributes.

2. According to the description from coursework, the test result for each citizens might have different types of virus, and each sample type of virus has its unique number and description.

Therefore:

sample_type --> type_name

type_name --> sample_type

sample_type --> description

description --> sample_type

type_name --> description

description --> type_name

At the same time:

mobile --> sample_type

mobile --> type_name

mobile --> description

Therefore, here existing transitive dependencies

mobile --> type_name--> sample_type--> description

mobile --> sample_type--> type_name--> description

mobile --> description--> sample_type--> type_name

So, I have to divide them into two tables.

3. Assumed that a citizen could do tests many times, but each test result could only test one type of virus and only belong to one citizen.

Therefore, assumed that "id" could identify each test report:

id-->Is_positive_result
id --> sample_collect_time
id--> sample_report_time
id--> sample_test_time
id--> doctor_name
id --> hospital_name

At the same time: each "id" of tested citizen could have only one test result for one test.
TestedCitizen.id --> TestResult.id

Therefore, here existing transitive dependencies:
TestedCitizen.id --> TestResult.id-->Is_positive_result
TestedCitizen.id --> TestResult.id--> sample_collect_time
TestedCitizen.id --> TestResult.id--> sample_report_time
TestedCitizen.id --> TestResult.id--> sample_test_time
TestedCitizen.id --> TestResult.id--> doctor_name
TestedCitizen.id --> TestResult.id--> hospital_name

So, I have to divide them into two tables.

4. Assumed that a doctor can only working in one hospital.

doctor_name --> hospital_name

At the same time:
mobile --> doctor_name
mobile--> hospital_name

Therefore, here existing transitive dependencies

mobile --> doctor_name --> hospital_name

So, I have to divide them into two tables.

Normalised tables and which normal form they are currently in:

```
CREATE TABLE `citizen`(
   `id` INT PRIMARY KEY AUTO_INCREMENT,
   `name` VARCHAR(100) NOT NULL,
   `age` INT NOT NULL,
   `gender` TINYINT NOT NULL,
   `mobile` VARCHAR(6) NOT NULL
   )
```

Table `citizen` is in the 3rd normal form.

```
CREATE TABLE `sampletype`(
   `id` INT PRIMARY KEY AUTO_INCREMENT,
   `type_name` VARCHAR(100) NOT NULL,
   `description` TEXT
)
```

Table `sampletype` is in the 3rd normal form.

```
CREATE TABLE `doctor_name`(
   `ID` INT AUTO_INCREMENT UNIQUE KEY NOT NULL,
   `name` VARCHAR(255) PRIMARY KEY,
   `hospital_name` VARCHAR(255)
)
```

Table `doctor_name` is in the 3rd normal form.

```
CREATE TABLE `testresult`(
   `id` INT PRIMARY KEY AUTO_INCREMENT,
   `citizen_id` INT NOT NULL,
   `sample_type` INT NOT NULL,
   `is_positive_result` TINYINT NOT NULL,
   `sample_collect_time` DATETIME,
   `sample_report_time` DATETIME,
   `sample_test_time` DATETIME,
   `doctor_name` VARCHAR(255) NOT NULL,
   `hospital_name` VARCHAR(255),
   `district_id` INT NOT NULL,
   CONSTRAINT FOREIGN KEY (citizen_id) REFERENCES citizen (id),
   CONSTRAINT FOREIGN KEY (sample_type) REFERENCES sampletype (id)
   ON UPDATE CASCADE
   ON DELETE CASCADE
   CONSTRAINT FOREIGN KEY (doctor_name) REFERENCES doctor_name (`name`)
   ON UPDATE SET NULL
   ON DELETE SET NULL
)
```

Table `testresult` is in the 3rd normal form.

Use Cases

Remember to put all of your SQL statements into the SQL script file, including all SELECT statements and INSERT statements used to insert test data.

Important Use Cases

This section lists some very important use cases of the PMMS. Your database design is expected to satisfy all of these use cases. ***Keep in mind that all use cases below should be achieved with a single SELECT statement (unless specified otherwise, sub-queries in a query is not counted as another query).*** Do not ignore the "explanation" or "proof" parts of this section, as they constitute the majority of your marks. If the SQL keywords/functions you learned cannot achieve these tasks, you are allowed to self-study some other keywords and use them. The example below is very simple and requires a short paragraph of explanation. But your answers should be more detailed.

------ The Beginning of the Example ------

Use case 0: Write a query to list all staff members in 'B007'. In the result, list staff names.

Your SQL statement:

```
SELECT staff_name FROM staff NATURAL JOIN branch where branch.branch_no = 'B007'
```

Test data and explanation:

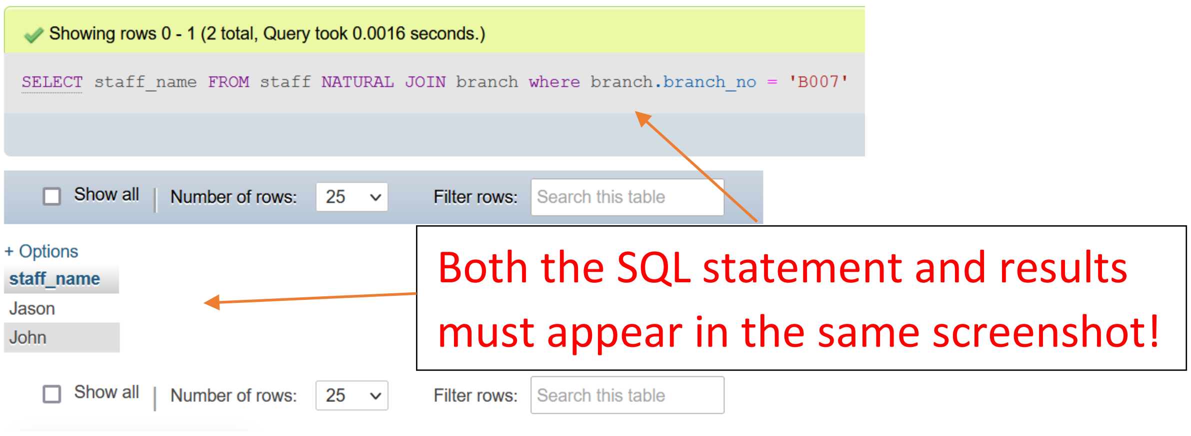
The following staff member information is added to the staff table (some attributes are hidden as they are not related to this task)

Staff_name	Branch_no
'Jason'	'B007'
'Anna'	'B002'
'John'	'B007'

The corresponding branch numbers ‘B007’ and ‘B002’ have already been added to the branch table.

This test data set contains staff members that are in ‘B007’ and not in ‘B007’. The expected result of the query should only contain staff in ‘B007’. Staff in other branches should be properly filtered out. For this test to work, all existing data in staff and branch need to be deleted first.

The result of the SELECT statement (screenshot):



Showing rows 0 - 1 (2 total, Query took 0.0016 seconds.)

```
SELECT staff_name FROM staff NATURAL JOIN branch where branch.branch_no = 'B007'
```

staff_name
Jason
John

Show all
Number of rows: 25
Filter rows: Search this table
+ Options
Both the SQL statement and results must appear in the same screenshot!

------ The End of the Example ------

Use case 1: A person can potentially get infected if he was in the same district with someone. The government requires that, if someone is tested to be positive, all people in the same district as him in the past 48 hours (before the positive report is published) need to take viral tests. Assume that a person called Mark was tested to be positive at 19:30 on 09-Oct-2021. Mark’s telephone number is 233636. Please write a query that can get the phone numbers of all citizens who will potentially get infected because of him.

Your SQL statement:

```
SELECT mobile FROM tracking
        WHERE    move_in_time <= '2021-10-09 19:30:00'
        AND    (    ISNULL(move_out_time) = 1 OR (move_out_time > '2021-10-07 19:30:00')   )
        AND district_id =  (SELECT district_id
        FROM tracking
        WHERE mobile = 233636)FROM tracking
        WHERE mobile = 233636)
```

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

Assumed that Mark got the positive result in District 1, 2021-10-09 19:30.

There are two types of peoples’ mobile number need to be collected:

1. People come into District 1 before 19:30 on 09-Oct-2021 and did not went out.
2. People come into District 1 before 19:30 on 09-Oct-2021 and the time they move out this area after 2021-10-07 19:30:00.

The following information is added to the `tracking` table:

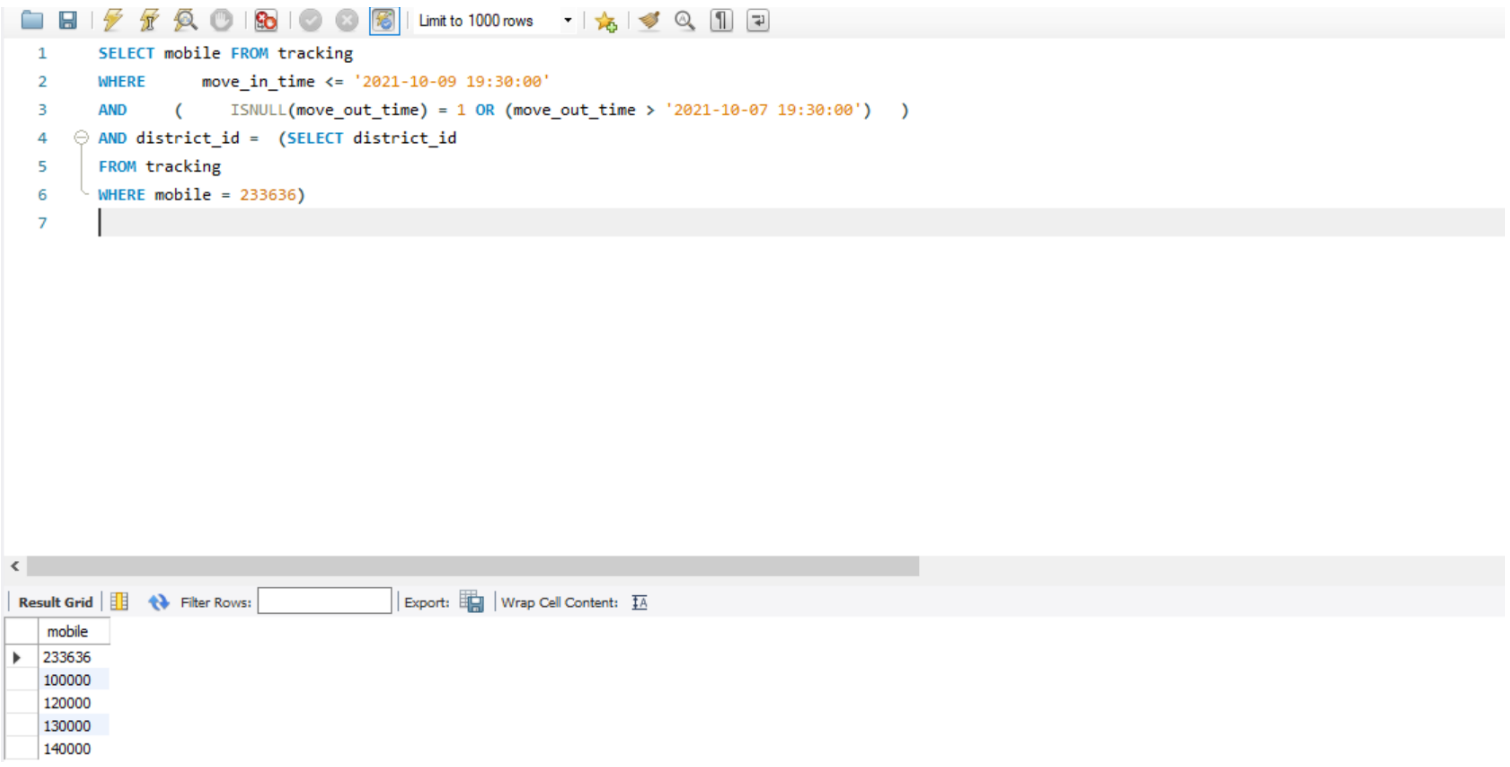
id	district_id	mobile	Move_in_time	Move_out_time
1	1	233636	2021-10-09 19:30:00	2021-10-10 10:00:00
2	1	100000	2021-10-06 14:00:00	NULL
3	2	110000	2021-10-08 15:00:00	2021-10-09 18:59:59
4	1	120000	2021-10-07 19:30:00	2021-10-09 19:29:59
5	1	130000	2021-10-05 06:00:00	2021-10-07 19:30:01
6	1	140000	2021-10-05 06:00:00	2021-10-09 19:30:01

People recorded in different district “1” and “2” have already been added to the `tracking` table.

This test data set contains peoples’ recorded in district 1**(id= 1,2,4,5,6)** and not in district 1**(id= 3)**, contains people come into district”1” before 19:30 on 09-Oct-2021 and did not went out **(id= 2)**, and People come into District 1 before 19:30 on 09-Oct-2021**(id= 3,4,5)** and the time they move out this area after 2021-10-07 19:30:00**(id= 6)**.

The expected result of the query should only contain the mobile number that people had stayed in district 1 during the period of time between 2021-10-07 19:30:00 and 2021-10-09 19:30:00.

The result of the SELECT statement (screenshot):



```
SELECT mobile FROM tracking
WHERE     move_in_time <= '2021-10-09 19:30:00'
AND    (     ISNULL(move_out_time) = 1 OR (move_out_time > '2021-10-07 19:30:00')    )
AND district_id =  (SELECT district_id
FROM tracking
WHERE mobile = 233636)
```

```
mobile
233636
100000
120000
130000
140000
```

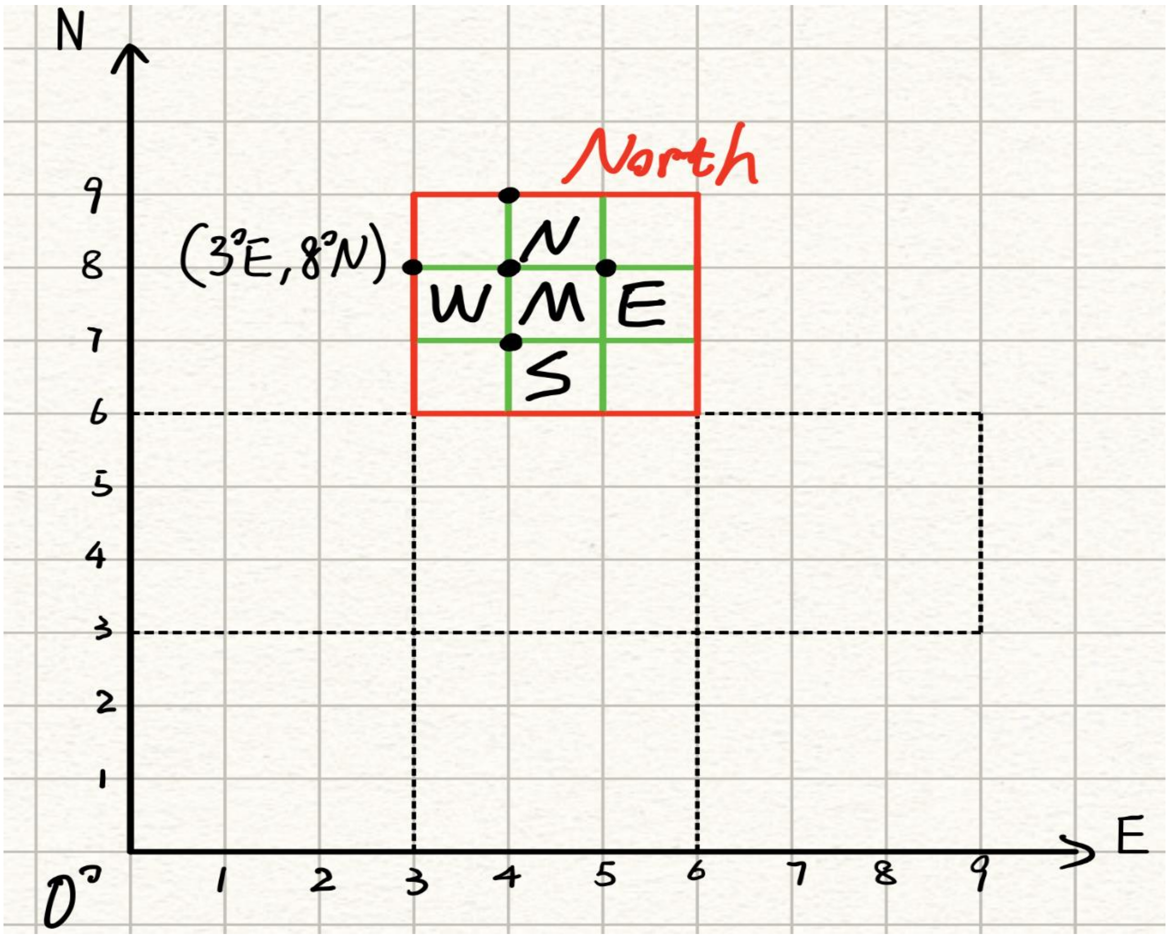
Use case 2: Please first clearly describe the format of GPS locations. The format must be a valid format that is used in real life. Then mimic what happens to your database when a user moves into the range of a base station and then moves out one hour later by listing all SQL statements involved in the process.

The GPS format and where did you learn it from (show the website link or the screenshot of the book):

GPS FORMAT:

Firstly, supposed that the location of The Lukewarm Kingdom could be expansion as a regular floor plan form with several squares. And define The Lukewarm Kingdom possess **5 region**, North-area, West-area, Mid-area, East-area, and South-area. And each area has **5 district**, North-district, West-district, Mid-district, East-district, and South-district. So, the name of district's format could be recognized as "**region-district**". For example, "N-w" means in the "west" district of "North" region.

Secondly, I define The Lukewarm Kingdom located on Earth with the domain of longitude from 0° to 9°E and latitude from 0° to 9°S, each district occupy a (1°,1°) square and each district has **one basetower**. Therefore, each region possesses 5 squares and could draw The Lukewarm Kingdom in the rectangular



coordinate system as following picture:

In this picture, the red square represent the region "North", green square represent the location of "district" "N"="North", "W"="West", "M"="Middle", "E"="East", "S"="South"; And the point • is the location of basetower. For example, the coordinate of (3°E, 8°N) represent the basetower in the **west district** of the **region "North"**.

Therefore, the format of GPS location is based on the rectangular coordinate system and each locations could be described as (longitude, latitude). And the value of longitude and latitude had been INSERT INTO table(`basetower`) `longitude` (DECIMAL(10,7) NOT NULL,`latitude` DECIMAL(10,7) NOT NULL)

The test data:

Because there is a foreign key between table `citizen` and table `tracking`, I have to insert this data into table citizen first.

name	age	gender	mobile
BB	30	female	111111

Your SQL statement(s) for travel record insertion:

INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`)

VALUES (1, 111111, NOW())

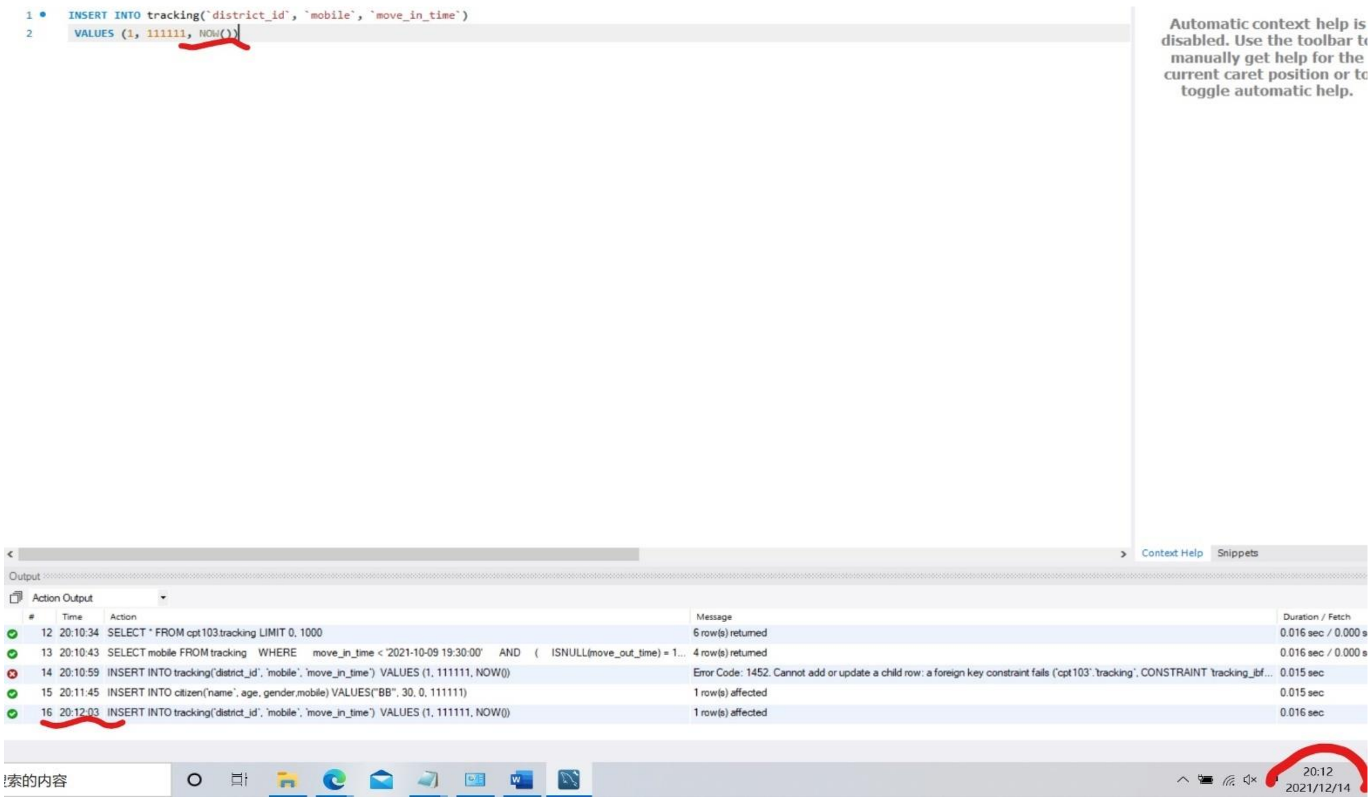
UPDATE tracking

SET `move_out_time` = NOW()

WHERE mobile = 111111 AND ISNULL(move_out_time)

//[Explain: in this case I wrote SET `move_out_time` = "2021-12-14 21:13:03" instead of "NOW()" to simulate the situation that citizen "BB" leave district 1 an hour later at "2021-12-14 21:13:03"]

The result of the SELECT statements (screenshot):

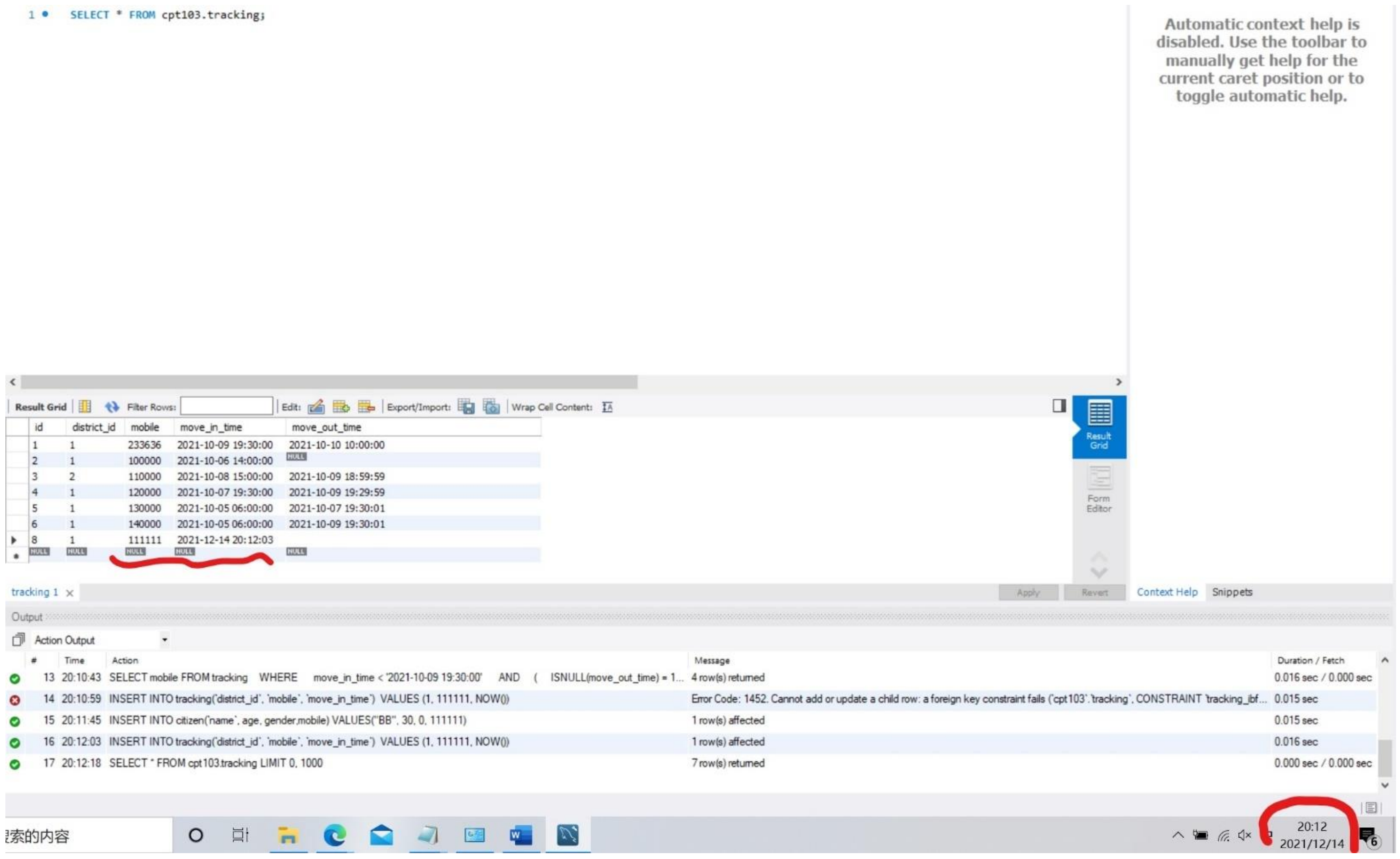


```
INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`)
 VALUES (1, 111111, NOW())
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

```
Output
Action Output
 #   Time      Action                                                                                              Message                                                                                                      Duration / Fetch
 12  20:10:34  SELECT * FROM cpt103.tracking LIMIT 0, 1000                                                         6 row(s) returned                                                                                            0.016 sec / 0.000 s
 13  20:10:43  SELECT mobile FROM tracking   WHERE   move_in_time < '2021-10-09 19:30:00'   AND   (   ISNULL(move_out_time) = 1...  4 row(s) returned                                                                       0.016 sec / 0.000 s
 14  20:10:59  INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`) VALUES (1, 111111, NOW())              Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`cpt103`.`tracking`, CONSTRAINT `tracking_ibf...  0.015 sec
 15  20:11:45  INSERT INTO citizen(`name`, age, gender,mobile) VALUES("BB", 30, 0, 111111)                          1 row(s) affected                                                                                            0.015 sec
 16  20:12:03  INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`) VALUES (1, 111111, NOW())              1 row(s) affected                                                                                            0.016 sec
```

In order to get the condition closer to reality, I choose the time that I test this query(2021-12-14 20:12:03) to simulate citizen “BB” move into district 1, and the result represent “BB” had moved into district 1 is in the following picture:



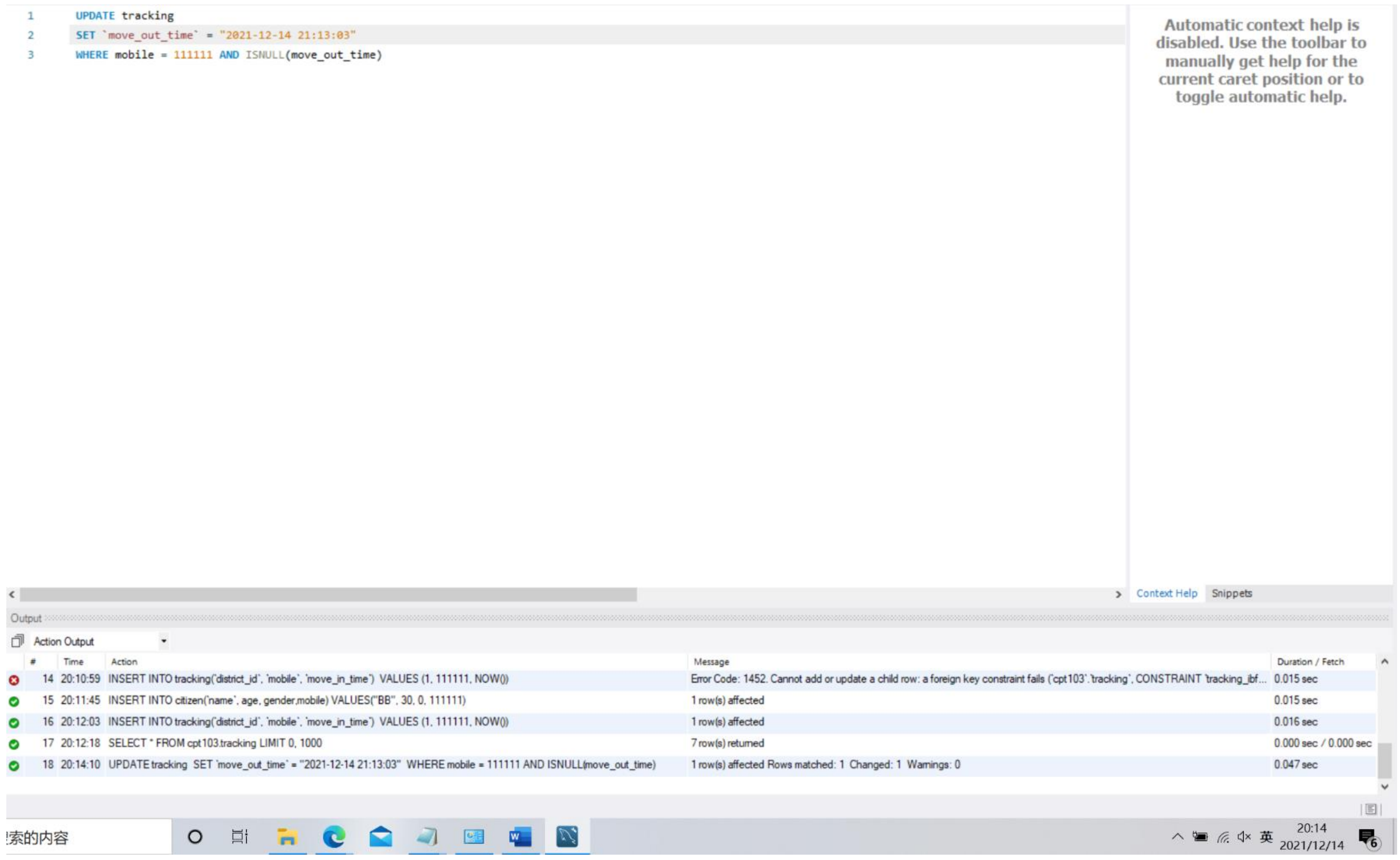
```sql
SELECT * FROM cpt103.tracking;
```

id	district_id	mobile	move_in_time	move_out_time
1	1	233636	2021-10-09 19:30:00	2021-10-10 10:00:00
2	1	100000	2021-10-06 14:00:00	NULL
3	2	110000	2021-10-08 15:00:00	2021-10-09 18:59:59
4	1	120000	2021-10-07 19:30:00	2021-10-09 19:29:59
5	1	130000	2021-10-05 06:00:00	2021-10-07 19:30:01
6	1	140000	2021-10-05 06:00:00	2021-10-09 19:30:01
8	1	111111	2021-12-14 20:12:03	
NULL	NULL	NULL	NULL	NULL

```
#   Time      Action                                                                                      Message                                                                                                      Duration / Fetch
13  20:10:43  SELECT mobile FROM tracking   WHERE   move_in_time < '2021-10-09 19:30:00'   AND   (   ISNULL(move_out_time) = 1...   4 row(s) returned                                                          0.016 sec / 0.000 sec
14  20:10:59  INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`) VALUES (1, 111111, NOW())     Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`cpt103`.`tracking`, CONSTRAINT `tracking_ibf...   0.015 sec
15  20:11:45  INSERT INTO citizen(`name`, age, gender,mobile) VALUES("BB", 30, 0, 111111)                  1 row(s) affected                                                                                            0.015 sec
16  20:12:03  INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`) VALUES (1, 111111, NOW())     1 row(s) affected                                                                                            0.016 sec
17  20:12:18  SELECT * FROM cpt103.tracking LIMIT 0, 1000                                                  7 row(s) returned                                                                                            0.000 sec / 0.000 sec
```

20:12
2021/12/14

Next, supposed that “BB” leave district 1 an hour later at 2021-12-14 21:13:03, here is the result:

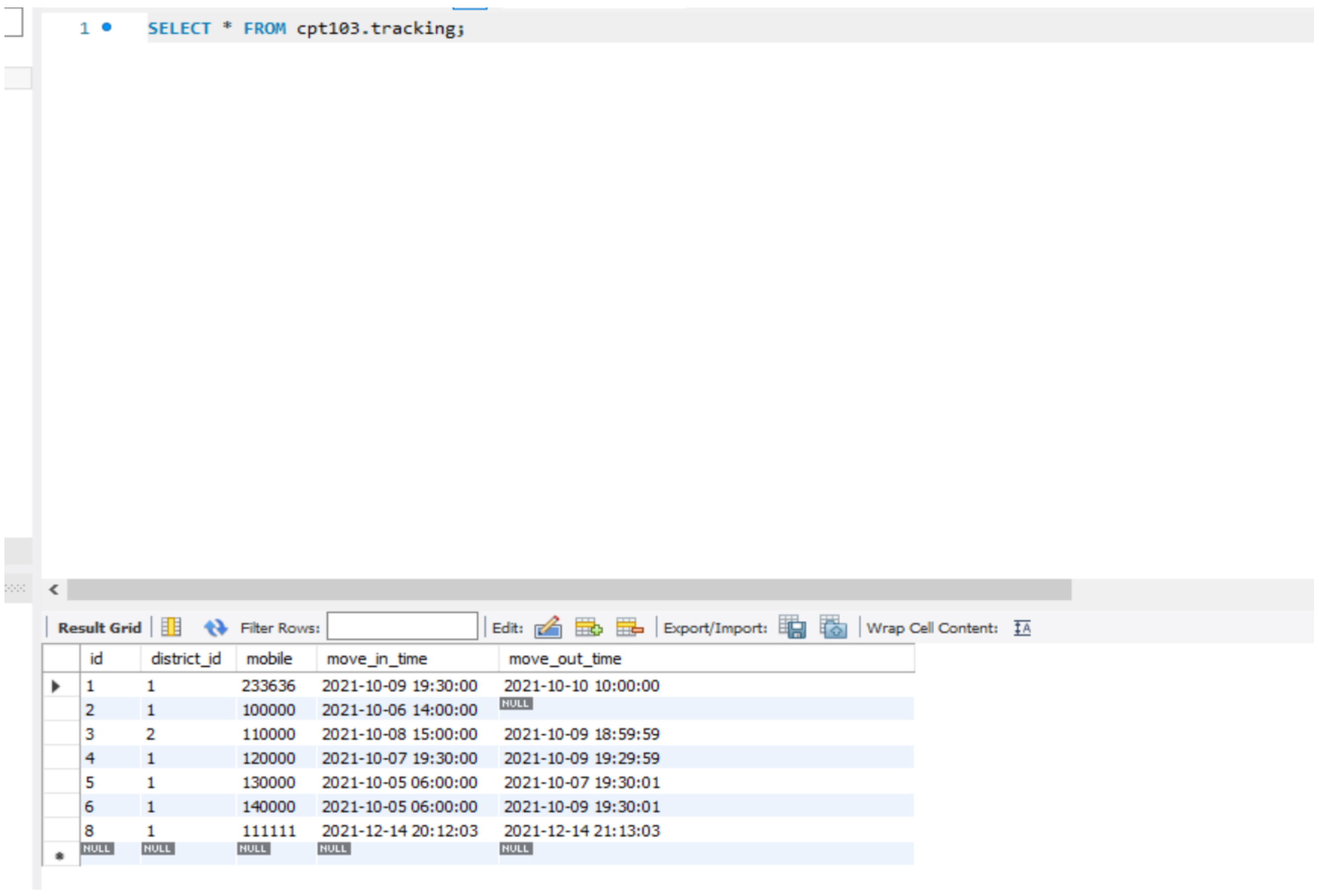


```sql
UPDATE tracking
SET `move_out_time` = "2021-12-14 21:13:03"
WHERE mobile = 111111 AND ISNULL(move_out_time)
```

```
#   Time      Action                                                                                      Message                                                                                                      Duration / Fetch
14  20:10:59  INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`) VALUES (1, 111111, NOW())     Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails (`cpt103`.`tracking`, CONSTRAINT `tracking_ibf...   0.015 sec
15  20:11:45  INSERT INTO citizen(`name`, age, gender,mobile) VALUES("BB", 30, 0, 111111)                  1 row(s) affected                                                                                            0.015 sec
16  20:12:03  INSERT INTO tracking(`district_id`, `mobile`, `move_in_time`) VALUES (1, 111111, NOW())     1 row(s) affected                                                                                            0.016 sec
17  20:12:18  SELECT * FROM cpt103.tracking LIMIT 0, 1000                                                  7 row(s) returned                                                                                            0.000 sec / 0.000 sec
18  20:14:10  UPDATE tracking  SET `move_out_time` = "2021-12-14 21:13:03"  WHERE mobile = 111111 AND ISNULL(move_out_time)   1 row(s) affected Rows matched: 1  Changed: 1  Warnings: 0                       0.047 sec
```

20:14
2021/12/14

And the record in table tracking:



```
SELECT * FROM cpt103.tracking;
```

	id	district_id	mobile	move_in_time	move_out_time
▶	1	1	233636	2021-10-09 19:30:00	2021-10-10 10:00:00
	2	1	100000	2021-10-06 14:00:00	NULL
	3	2	110000	2021-10-08 15:00:00	2021-10-09 18:59:59
	4	1	120000	2021-10-07 19:30:00	2021-10-09 19:29:59
	5	1	130000	2021-10-05 06:00:00	2021-10-07 19:30:01
	6	1	140000	2021-10-05 06:00:00	2021-10-09 19:30:01
	8	1	111111	2021-12-14 20:12:03	2021-12-14 21:13:03
●	NULL	NULL	NULL	NULL	NULL

Use case 3: The Lukewarm Kingdom wants to find out the hospitals that can do viral tests efficiently. The report generation time is calculated using (report time - sample test time). Please write a query to find out which hospital has the least average report generation time.

Your SQL statement:

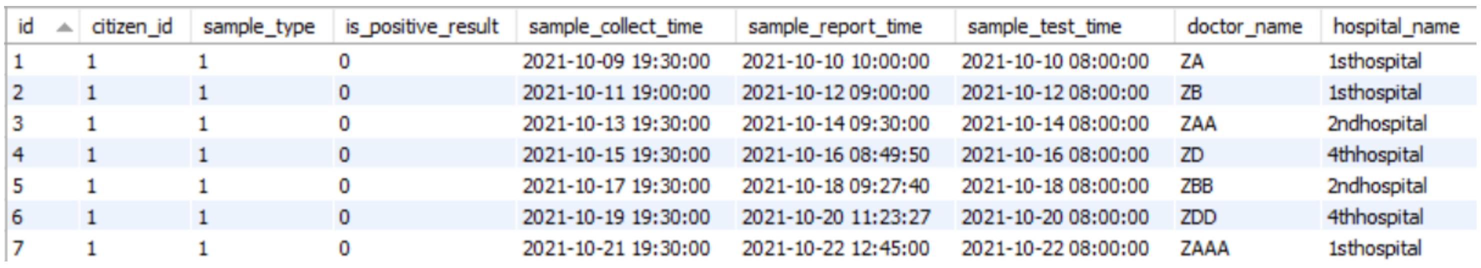
SELECT AVG(testresult.sample_report_time-testresult.sample_test_time) avg_time, doctor_name.`hospital_name` hospital_name

FROM testresult, doctor_name

WHERE testresult.doctor_name = doctor_name.`name`

GROUP BY doctor_name.`hospital_name`

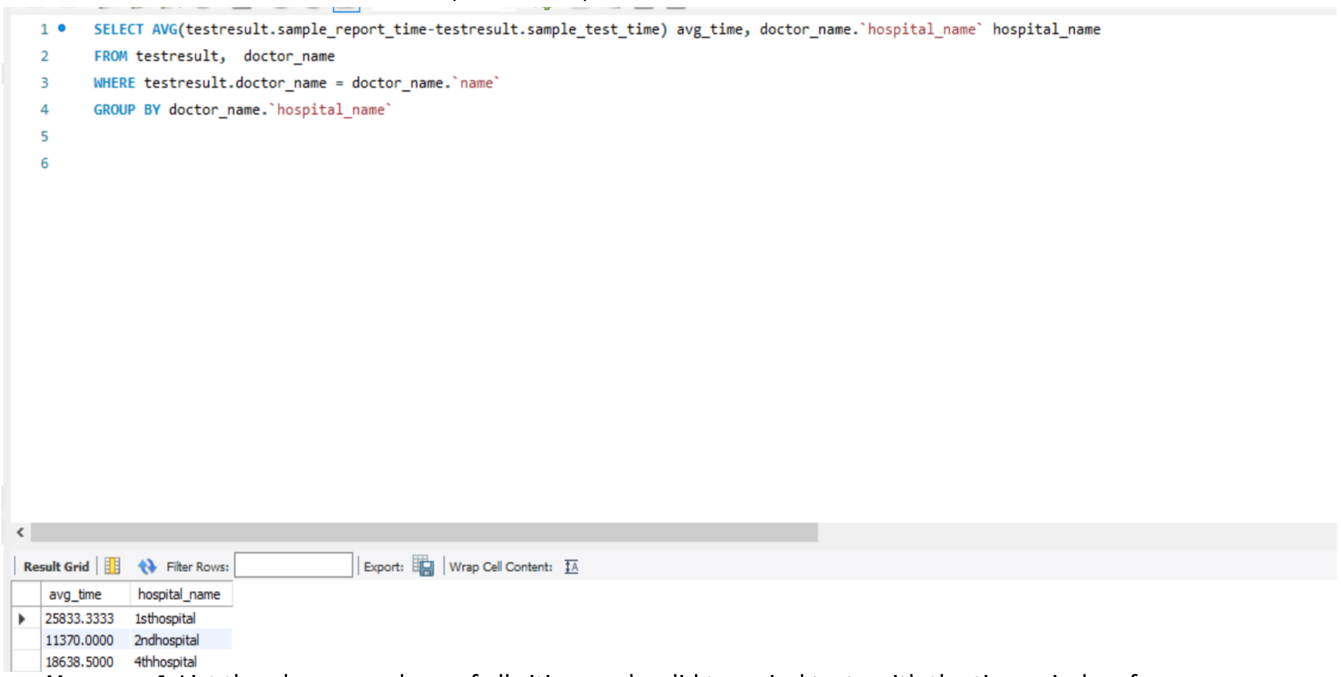
Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):



id	citizen_id	sample_type	is_positive_result	sample_collect_time	sample_report_time	sample_test_time	doctor_name	hospital_name
1	1	1	0	2021-10-09 19:30:00	2021-10-10 10:00:00	2021-10-10 08:00:00	ZA	1sthospital
2	1	1	0	2021-10-11 19:00:00	2021-10-12 09:00:00	2021-10-12 08:00:00	ZB	1sthospital
3	1	1	0	2021-10-13 19:30:00	2021-10-14 09:30:00	2021-10-14 08:00:00	ZAA	2ndhospital
4	1	1	0	2021-10-15 19:30:00	2021-10-16 08:49:50	2021-10-16 08:00:00	ZD	4thhospital
5	1	1	0	2021-10-17 19:30:00	2021-10-18 09:27:40	2021-10-18 08:00:00	ZBB	2ndhospital
6	1	1	0	2021-10-19 19:30:00	2021-10-20 11:23:27	2021-10-20 08:00:00	ZDD	4thhospital
7	1	1	0	2021-10-21 19:30:00	2021-10-22 12:45:00	2021-10-22 08:00:00	ZAAA	1sthospital

As it could be seen in this figure, here is 7 test result of citizen Mark (citizen_id=1), he did tests in different hospital (1sthospital, 2ndhospital, 4thhospital), and he did test in each hospital by different doctors. I choose these data to test is to prove that the result of query to count average time for each hospital will not affect by different doctors.

The result of the SELECT statement (screenshot):



```sql
SELECT AVG(testresult.sample_report_time-testresult.sample_test_time) avg_time, doctor_name.`hospital_name` hospital_name
FROM testresult,  doctor_name
WHERE testresult.doctor_name = doctor_name.`name`
GROUP BY doctor_name.`hospital_name`
```

```
avg_time     hospital_name
25833.3333   1sthospital
11370.0000   2ndhospital
18638.5000   4thhospital
```

Use case 4: List the phone numbers of all citizens who did two viral tests with the time window from 2021-10-03 00:00 to 2021-10-05 00:00. The two viral tests must have a gap time of at least 24 hours (at least 24 hours apart).

Your SQL statement: N/A

The result of the SELECT statement (screenshot): N/A

Use case 5: List the high-risk, mid-risk and low-risk districts using one query. High-risk districts should be listed first, followed by mid-risk districts and then low-risk districts. Example:

district_name	**risk_level**
Centre Lukewarm Hillside	high
Lenny town	high
Glow Sand district	mid
Raspberry town	low
Bunny Tail district	low

Your SQL statement:

SELECT district_name, risk_level_name

FROM district

ORDER BY risk_level DESC

Test Data:

district_name	region	risk_level	risk_level_name
N-w	1	3	high
E-w	4	2	mid
M-w	3	3	high
S-w	5	1	low
W-n	2	2	mid

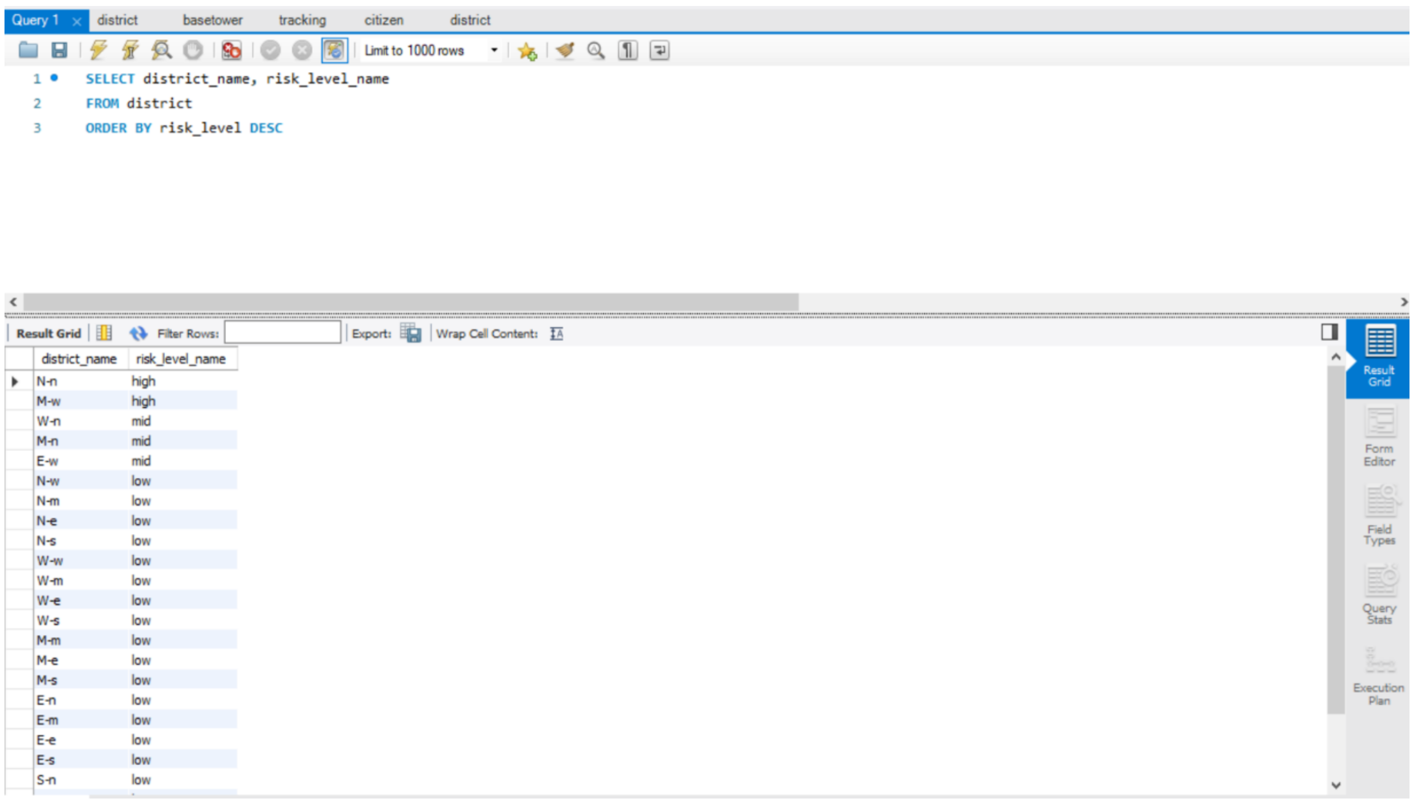
Firstly, define The Lukewarm Kingdom possess 5 region, North-area, West-area, Mid-area, East-area, and South-area. And each area has 5 district, North-district, West-district, Mid-district, East-district, and South-district. So, the name of district's format could be recognized as "region-district". For example, "N-w" means in the "west" district of "North" region.

Secondly, I define each region has its number, 1=North, 2=West, 3=Mid, 4=East, 5=South.

Thirdly, I define each risk level with correspond number to represent, low=1, mid=2, high=3.

This test data set contains districts with different risk level. The expected result of the query should only contain the name of districts and the name of risk level and order by high-mid-low. Therefore, for this test to work, I should only select district_name and risk_level_name to show and order by resk_level.

The result of the SELECT statement (screenshot):



```
SELECT district_name, risk_level_name
FROM district
ORDER BY risk_level DESC
```

district_name	risk_level_name
N-n	high
M-w	high
W-n	mid
M-n	mid
E-w	mid
N-w	low
N-m	low
N-e	low
N-s	low
W-w	low
W-m	low
W-e	low
W-s	low
M-m	low
M-e	low
M-s	low
E-n	low
E-m	low
E-e	low
E-s	low
S-n	low

Use case 6: List all positive cases found in the district called “Centre Lukewarm Hillside” on 2021-10-04. The result should include the names and phone numbers of people tested to be positive.

Your SQL statement:

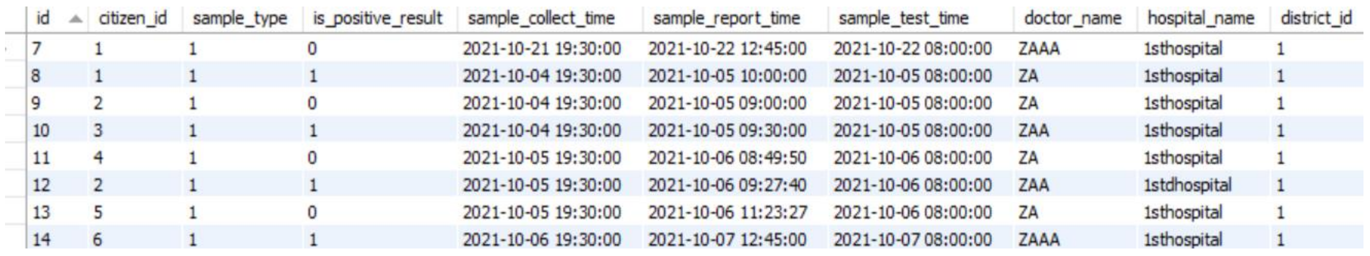
```
SELECT citizen.`name`, citizen.mobile

FROM citizen , testresult , district

WHERE citizen.id = testresult.citizen_id AND testresult.district_id = district.id

AND testresult.is_positive_result = 1

AND (testresult.sample_collect_time BETWEEN '2021-10-04 00:00:00' AND '2021-10-04 23:59:59')

AND district.district_name = "Centre Lukewarm Hillside"
```

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

In this case, I have update the name of which district.id =1 from “N-n” to “Centre Lukewarm Hillside”. There are 6 citizens from id=1 to id=6 in the test data. All positive cases found on 2021-10-04 should be the test collected time during the time between 2021-10-04 00:00:00 and 2021-10-04 23:59:59.

Citizen.id	name	age	gender	mobile
1	Mark	20	1	233636
2	AA	50	1	100000
3	AB	40	1	110000
4	AC	30	1	120000
5	AD	15	1	130000
6	BA	20	0	140000

(Figure 1)



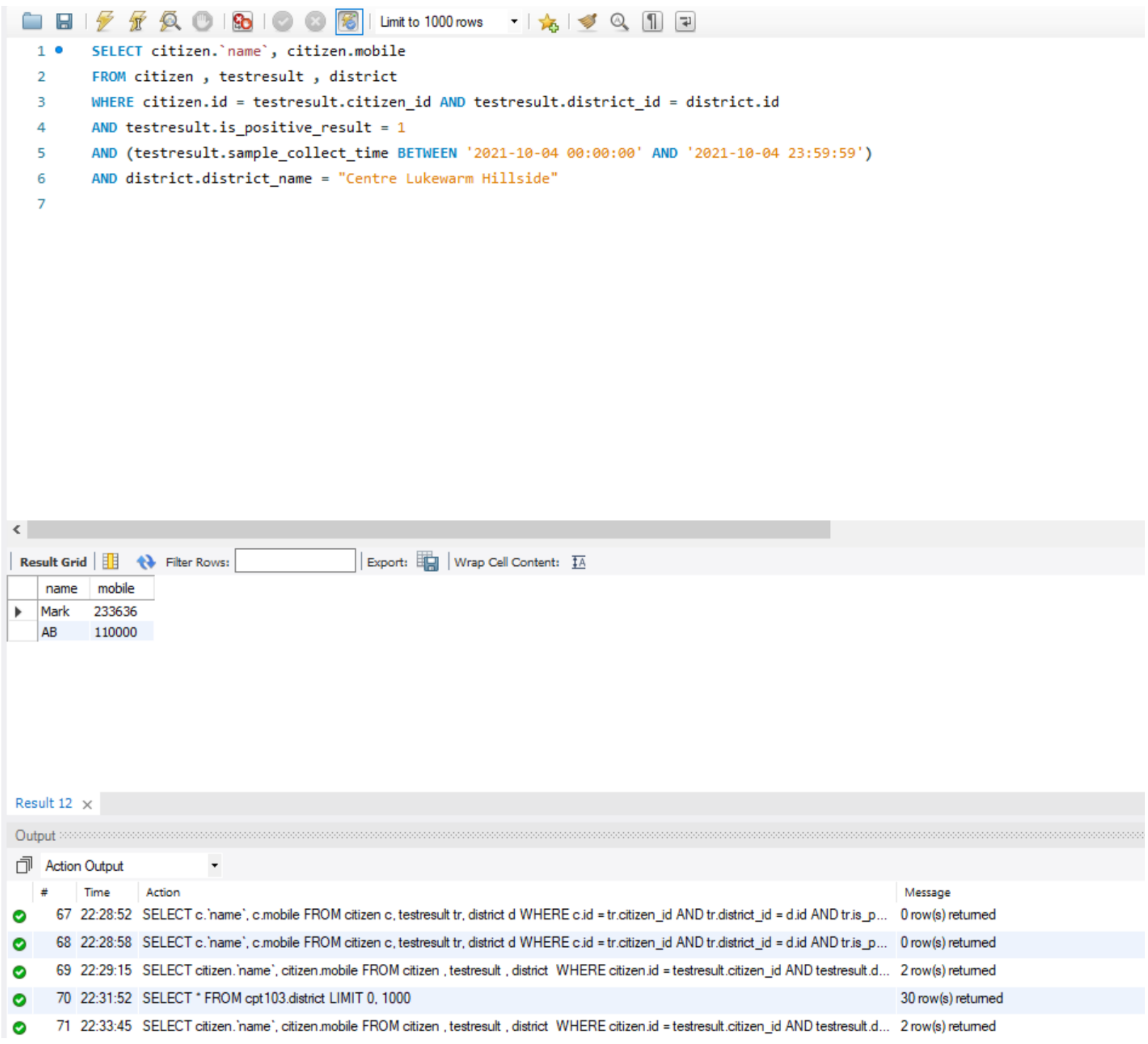
id	citizen_id	sample_type	is_positive_result	sample_collect_time	sample_report_time	sample_test_time	doctor_name	hospital_name	district_id
7	1	1	0	2021-10-21 19:30:00	2021-10-22 12:45:00	2021-10-22 08:00:00	ZAAA	1sthospital	1
8	1	1	1	2021-10-04 19:30:00	2021-10-05 10:00:00	2021-10-05 08:00:00	ZA	1sthospital	1
9	2	1	0	2021-10-04 19:30:00	2021-10-05 09:00:00	2021-10-05 08:00:00	ZA	1sthospital	1
10	3	1	1	2021-10-04 19:30:00	2021-10-05 09:30:00	2021-10-05 08:00:00	ZAA	1sthospital	1
11	4	1	0	2021-10-05 19:30:00	2021-10-06 08:49:50	2021-10-06 08:00:00	ZA	1sthospital	1
12	2	1	1	2021-10-05 19:30:00	2021-10-06 09:27:40	2021-10-06 08:00:00	ZAA	1stdhospital	1
13	5	1	0	2021-10-05 19:30:00	2021-10-06 11:23:27	2021-10-06 08:00:00	ZA	1sthospital	1
14	6	1	1	2021-10-06 19:30:00	2021-10-07 12:45:00	2021-10-07 08:00:00	ZAAA	1sthospital	1

(Figure 2)

Figure 1 is the information of 6 citizens, and Figure 2 is the test result for each of citizens.

As it can be seen in Figure 2, citizens’ sample_collect_time during the day of 2021-10-04 are: Mark(id=1), AB(id=2), AB(id=3). Therefore, although citizen BA’s test result is positive(is_positive_result=1), the result of all positive cases found in the district “Centre Lukewarm Hillside” on 2021-10-04 remains Mark and AB, and their mobile phone number are 233636 and 110000.(Mobile number could be found in Figure 1)

The result of the SELECT statement (screenshot):



```
SELECT citizen.`name`, citizen.mobile
FROM citizen , testresult , district
WHERE citizen.id = testresult.citizen_id AND testresult.district_id = district.id
AND testresult.is_positive_result = 1
AND (testresult.sample_collect_time BETWEEN '2021-10-04 00:00:00' AND '2021-10-04 23:59:59')
AND district.district_name = "Centre Lukewarm Hillside"
```

name	mobile
Mark	233636
AB	110000

```
#   Time      Action                                                                                                       Message
67  22:28:52  SELECT c.`name`, c.mobile FROM citizen c, testresult tr, district d WHERE c.id = tr.citizen_id AND tr.district_id = d.id AND tr.is_p...   0 row(s) returned
68  22:28:58  SELECT c.`name`, c.mobile FROM citizen c, testresult tr, district d WHERE c.id = tr.citizen_id AND tr.district_id = d.id AND tr.is_p...   0 row(s) returned
69  22:29:15  SELECT citizen.`name`, citizen.mobile FROM citizen , testresult , district  WHERE citizen.id = testresult.citizen_id AND testresult.d...   2 row(s) returned
70  22:31:52  SELECT * FROM cpt103.district LIMIT 0, 1000                                                                  30 row(s) returned
71  22:33:45  SELECT citizen.`name`, citizen.mobile FROM citizen , testresult , district  WHERE citizen.id = testresult.citizen_id AND testresult.d...   2 row(s) returned
```

Use case 7: Calculate the increase in new positive cases in the district called “Centre Lukewarm Hillside” on 2021-10-05 compared to 2021-10-04. The result should show a single number indicating the increment. If there are fewer new positive cases than yesterday, this number should be negative.

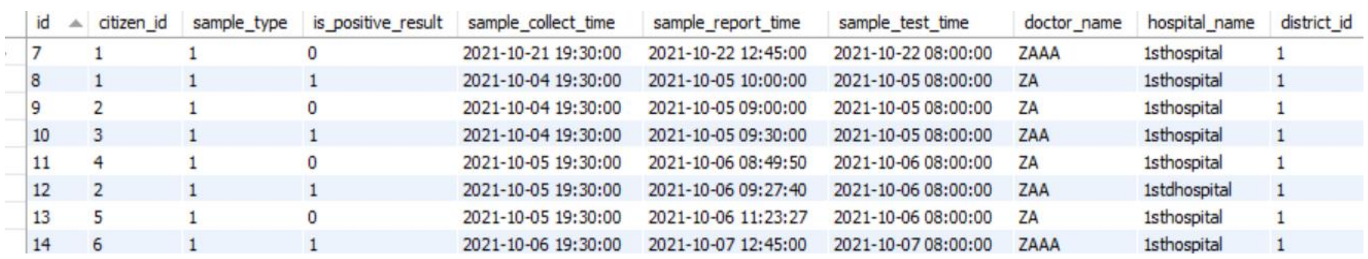
Your SQL statement:

```
SELECT (
                (SELECT COUNT(*) FROM (SELECT c.`name`, c.mobile
        FROM citizen c, testresult tr, district d
        WHERE c.id = tr.citizen_id AND tr.district_id = d.id
                AND tr.is_positive_result = 1
                AND (tr.sample_collect_time BETWEEN '2021-10-05 00:00:00' AND '2021-10-05 
23:59:59')
        AND d.district_name = "Centre Lukewarm Hillside") as tmp1) -
                (SELECT COUNT(*) FROM (SELECT c.`name`, c.mobile
                FROM citizen c, testresult tr, district d
                WHERE c.id = tr.citizen_id AND tr.district_id = d.id
                AND tr.is_positive_result = 1
               AND (tr.sample_collect_time BETWEEN '2021-10-04 00:00:00' AND '2021-10-4 23:59:59')
                AND d.district_name = "Centre Lukewarm Hillside") as tmp2)
) result
```

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

All positive cases found on 2021-10-04 should be the test collected time during the time between 2021-10-04 00:00:00 and 2021-10-04 23:59:59.

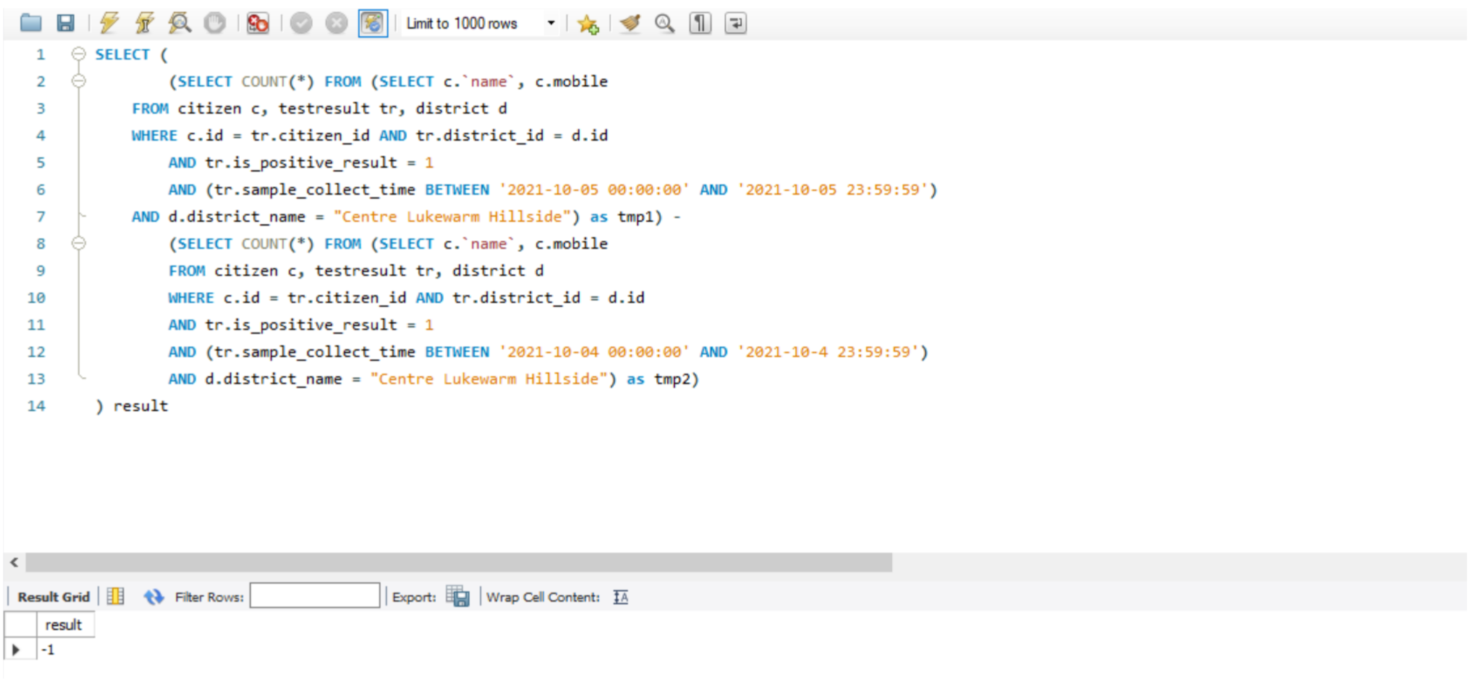
All positive cases found on 2021-10-05 should be the test collected time during the time between 2021-10-05 00:00:00 and 2021-10-05 23:59:59.



id	citizen_id	sample_type	is_positive_result	sample_collect_time	sample_report_time	sample_test_time	doctor_name	hospital_name	district_id
7	1	1	0	2021-10-21 19:30:00	2021-10-22 12:45:00	2021-10-22 08:00:00	ZAAA	1sthospital	1
8	1	1	1	2021-10-04 19:30:00	2021-10-05 10:00:00	2021-10-05 08:00:00	ZA	1sthospital	1
9	2	1	0	2021-10-04 19:30:00	2021-10-05 09:00:00	2021-10-05 08:00:00	ZA	1sthospital	1
10	3	1	1	2021-10-04 19:30:00	2021-10-05 09:30:00	2021-10-05 08:00:00	ZAA	1sthospital	1
11	4	1	0	2021-10-05 19:30:00	2021-10-06 08:49:50	2021-10-06 08:00:00	ZA	1sthospital	1
12	2	1	1	2021-10-05 19:30:00	2021-10-06 09:27:40	2021-10-06 08:00:00	ZAA	1stdhospital	1
13	5	1	0	2021-10-05 19:30:00	2021-10-06 11:23:27	2021-10-06 08:00:00	ZA	1sthospital	1
14	6	1	1	2021-10-06 19:30:00	2021-10-07 12:45:00	2021-10-07 08:00:00	ZAAA	1sthospital	1

As it can be seen in the table `testresult`, I choose the testresult.id from 7 to 14 as example to simulate this operation. There are 2 cases are positive on 2021-10-04(`id`=8 and `id`=10), and one positive in 2021-10-05(`id`= 12). Therefore, the query shouldn't choose other dates that does not meet the requirement, and the result must be -1.

The result of the SELECT statement (screenshots):



```
SELECT (
        (SELECT COUNT(*) FROM (SELECT c.`name`, c.mobile
    FROM citizen c, testresult tr, district d
    WHERE c.id = tr.citizen_id AND tr.district_id = d.id
        AND tr.is_positive_result = 1
        AND (tr.sample_collect_time BETWEEN '2021-10-05 00:00:00' AND '2021-10-05 23:59:59')
    AND d.district_name = "Centre Lukewarm Hillside") as tmp1) -
        (SELECT COUNT(*) FROM (SELECT c.`name`, c.mobile
        FROM citizen c, testresult tr, district d
        WHERE c.id = tr.citizen_id AND tr.district_id = d.id
        AND tr.is_positive_result = 1
        AND (tr.sample_collect_time BETWEEN '2021-10-04 00:00:00' AND '2021-10-4 23:59:59')
        AND d.district_name = "Centre Lukewarm Hillside") as tmp2)
) result
```

Limit to 1000 rows | Result Grid | Filter Rows: | Export: | Wrap Cell Content:

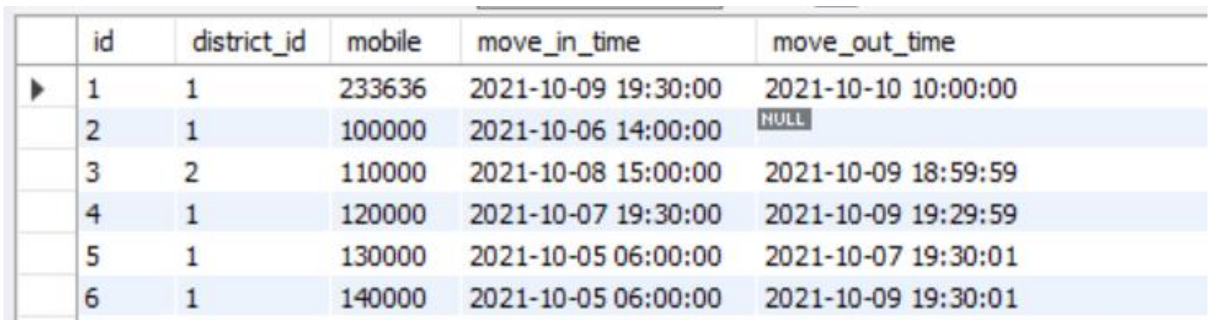
result
-1

number of people among them that later confirmed to be infected in 14 days. Again, assume that a person called Mark (telephone number is 233636) was tested to be positive at 19:30 on 09-Oct-2021 and he is the only person in the country that has coughid-19. Please write a query that calculates the spread rate of the virus.

Your SQL statement:

```
SELECT(
        (
                SELECT COUNT(mobile) total_num
                FROM tracking
                WHERE
                                        move_in_time <= '2021-10-09 19:30:00'
                                        AND
                                        (
                                                ISNULL(move_out_time) = 1 OR (move_out_time
BETWEEN '2021-10-07 19:30:00' AND '2021-10-09 19:30:00')
                                        )

                AND district_id =
                (SELECT district_id
                FROM tracking
                WHERE mobile = 233636)
        )
        /
        (
                SELECT COUNT(c.mobile)
                FROM citizen c, testresult tr, district d
                WHERE c.id = tr.citizen_id AND tr.district_id = d.id
                AND tr.is_positive_result = 1
                AND (tr.sample_report_time BETWEEN '2021-10-09 19:30:00' AND '2021-10-23
19:30:00')
        )

) transmission_rate
```

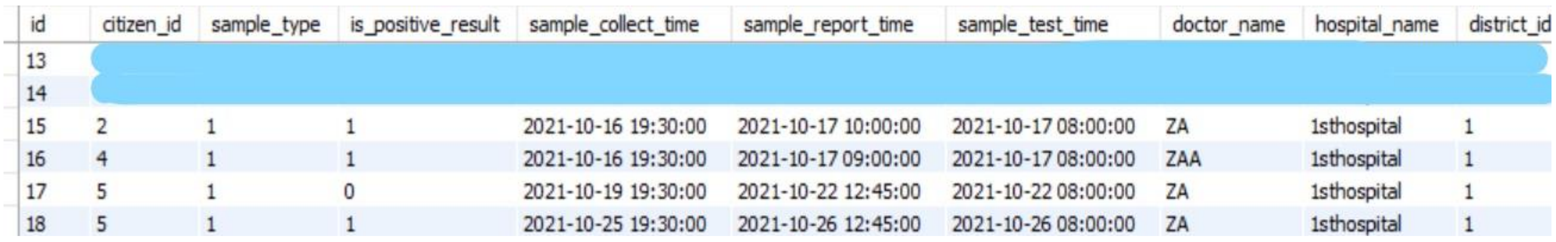
Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):



	id	district_id	mobile	move_in_time	move_out_time
▶	1	1	233636	2021-10-09 19:30:00	2021-10-10 10:00:00
	2	1	100000	2021-10-06 14:00:00	NULL
	3	2	110000	2021-10-08 15:00:00	2021-10-09 18:59:59
	4	1	120000	2021-10-07 19:30:00	2021-10-09 19:29:59
	5	1	130000	2021-10-05 06:00:00	2021-10-07 19:30:01
	6	1	140000	2021-10-05 06:00:00	2021-10-09 19:30:01

(Figure 1)

Figure 1 is the dates in table tracking, it is used to simulate the condition in this case, and only id=2 did not match the requirement. Therefore, the total number of people that were in the same district(district_id = 1) as the positive case with 48 hours is 5.

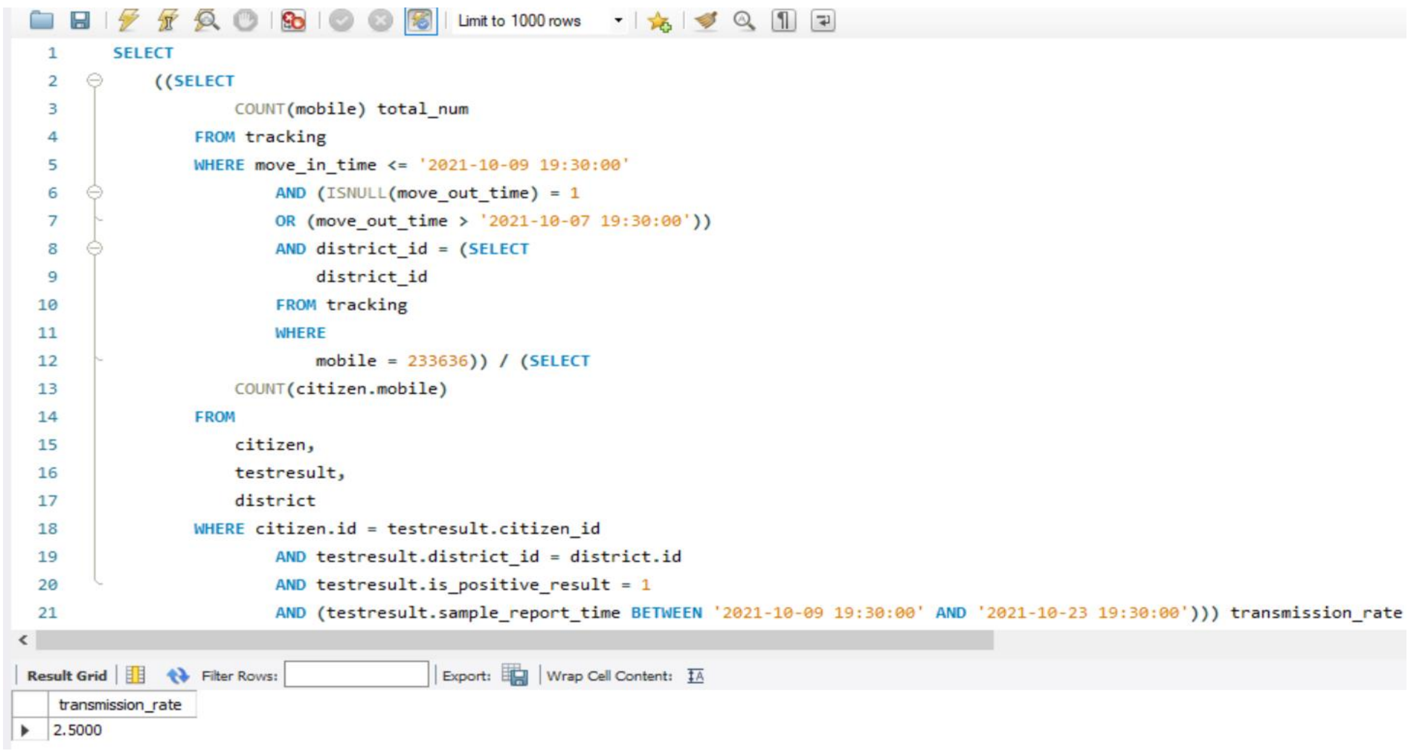


id	citizen_id	sample_type	is_positive_result	sample_collect_time	sample_report_time	sample_test_time	doctor_name	hospital_name	district_id
13	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
14	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
15	2	1	1	2021-10-16 19:30:00	2021-10-17 10:00:00	2021-10-17 08:00:00	ZA	1sthospital	1
16	4	1	1	2021-10-16 19:30:00	2021-10-17 09:00:00	2021-10-17 08:00:00	ZAA	1sthospital	1
17	5	1	0	2021-10-19 19:30:00	2021-10-22 12:45:00	2021-10-22 08:00:00	ZA	1sthospital	1
18	5	1	1	2021-10-25 19:30:00	2021-10-26 12:45:00	2021-10-26 08:00:00	ZA	1sthospital	1

(Figure 2)

As it can be seen in the Figure 2, in order to simulate the condition in this case, I have insert 4 more data into table testresult from id=15 to id=18. These citizens are in the same district(district_id = 1), and citizen_id=2,4 need to be counted because they are positive and meet the requirement of this case and should not be affect by different doctor in the same hospital. In addition, citizen_id=5 is a disturbance term in test data should not be counted, during the time period between 2021-10-09 19:30:00 and 2021-10-23 19:30:00 is nagetive, after 2021-10-23 19:30:00 is positive. Therefore, 2 citizens(citizen_id=2,4) were inferred by Mark.

The result of the SELECT statement (screenshots):



```sql
SELECT
    ((SELECT
            COUNT(mobile) total_num
        FROM tracking
        WHERE move_in_time <= '2021-10-09 19:30:00'
                AND (ISNULL(move_out_time) = 1
                OR (move_out_time > '2021-10-07 19:30:00'))
                AND district_id = (SELECT
                    district_id
                FROM tracking
                WHERE
                    mobile = 233636)) / (SELECT
            COUNT(citizen.mobile)
        FROM
            citizen,
            testresult,
            district
        WHERE citizen.id = testresult.citizen_id
                AND testresult.district_id = district.id
                AND testresult.is_positive_result = 1
                AND (testresult.sample_report_time BETWEEN '2021-10-09 19:30:00' AND '2021-10-23 19:30:00'))) transmission_rate
```

```
transmission_rate
2.5000
```

Extended Use Cases

Apart from the use cases proposed in the previous section, your database could also support more scenarios. Please follow the same format in the previous section and write down your own 10 use cases. You are allowed to use keywords learned outside of the lectures. Practical use cases displaying good innovations will receive higher marks.

Use case 1: List the ages of all citizens from the oldest to the youngest.

Your SQL statement:

SELECT `name`, age

FROM citizen

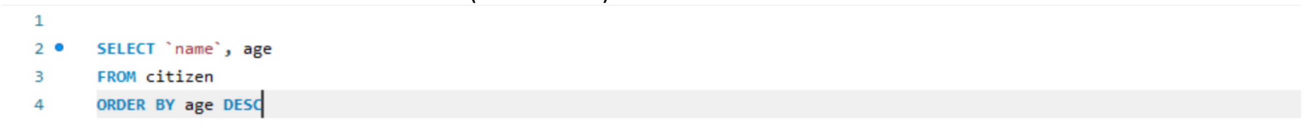
ORDER BY age DESC

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

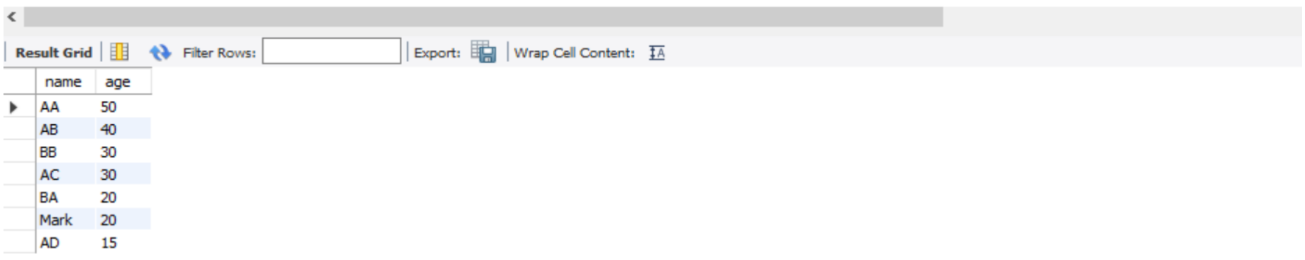
id	name	age	gender	mobile
1	Mark	20	1	233636
2	AA	50	1	100000
3	AB	40	1	110000
4	AC	30	1	120000
5	AD	15	1	130000
6	BA	20	0	140000
7	BB	30	0	111111

As it can be seen in the table, supposed that all citizens in The Lukewarm Kingdom are in this figure, it is clear to see that other attributes would not affect the result.

The result of the SELECT statement (screenshot):



```

SELECT `name`, age
FROM citizen
ORDER BY age DESC
```



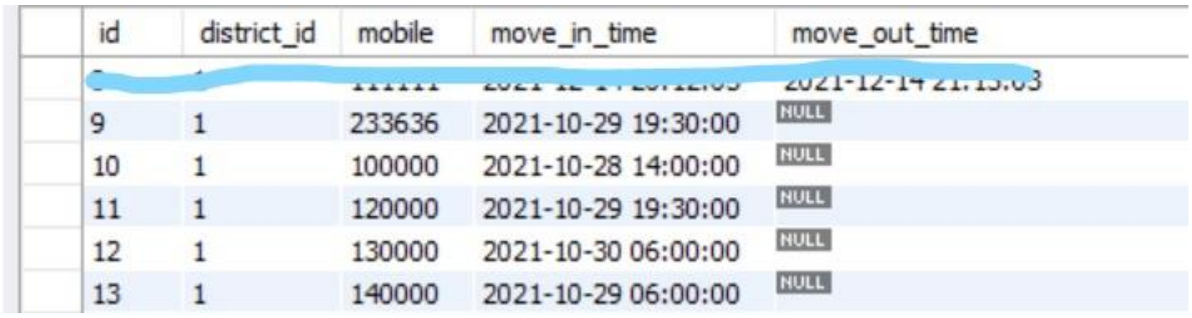
name	age
AA	50
AB	40
BB	30
AC	30
BA	20
Mark	20
AD	15

Use case 2: A citizen named “Mrs. Yangzhou Old Lady” is still playing mah-jong in District “Centre Lukewarm Hillside” on 2021-10-30 and didn’t realize that she was positive. District “Centre Lukewarm Hillside” was blocked and asking for all men in district “Centre Lukewarm Hillside” for community volunteers in age 20-40 years old, please list their mobile numbers.

Your SQL statement:

```
SELECT citizen.`name`, citizen.mobile

FROM citizen , tracking , district

WHERE citizen.mobile = tracking.mobile AND tracking.district_id = district.id

AND (tracking.move_in_time <= '2021-10-30 19:30:00' AND ISNULL(tracking.move_out_time))

AND district.district_name = "Centre Lukewarm Hillside"

AND citizen.age BETWEEN 20 AND 40

AND citizen.gender = 1
```

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):



id	district_id	mobile	move_in_time	move_out_time
[redacted]	[redacted]	[redacted]	[redacted]	[redacted]
9	1	233636	2021-10-29 19:30:00	NULL
10	1	100000	2021-10-28 14:00:00	NULL
11	1	120000	2021-10-29 19:30:00	NULL
12	1	130000	2021-10-30 06:00:00	NULL
13	1	140000	2021-10-29 06:00:00	NULL

(Figure 1)

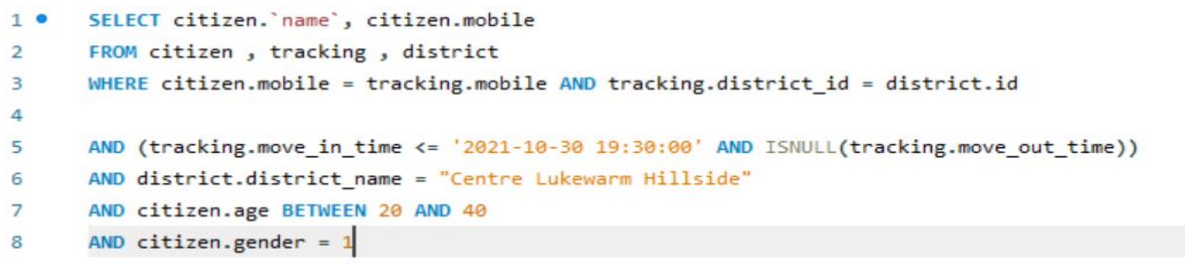
id	name	age	gender	mobile
1	Mark	20	1	233636
2	AA	50	1	100000
3	AB	40	1	110000
4	AC	30	1	120000
5	AD	15	1	130000
6	BA	20	0	140000
7	BB	30	0	111111

(Figure 2)

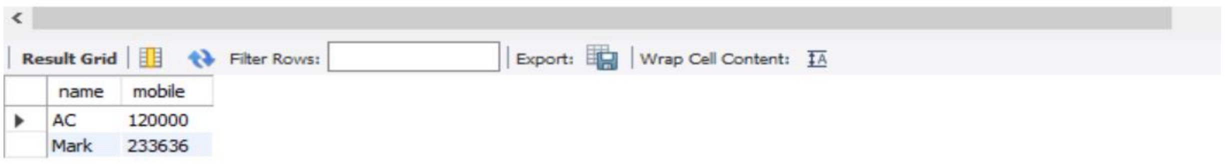
Figure 1 shows the tracking information of citizens, only five tracking record in this table (id=9-13), which means that 5 citizen is move into district “Centre Lukewarm Hillside”.

Figure shows the information of citizens, as it can be seen in the figure, only Mark and AC meet the requirement to be volunteer.

The result of the SELECT statement (screenshot):



```
SELECT citizen.`name`, citizen.mobile
FROM citizen , tracking , district
WHERE citizen.mobile = tracking.mobile AND tracking.district_id = district.id

AND (tracking.move_in_time <= '2021-10-30 19:30:00' AND ISNULL(tracking.move_out_time))
AND district.district_name = "Centre Lukewarm Hillside"
AND citizen.age BETWEEN 20 AND 40
AND citizen.gender = 1
```



```
name   mobile
AC     120000
Mark   233636
```

Use case 3:

Your SQL statement: Please calculate the gender ratio result of Male/Female in district “Centre Lukewarm Hillside”.

Your SQL statement:

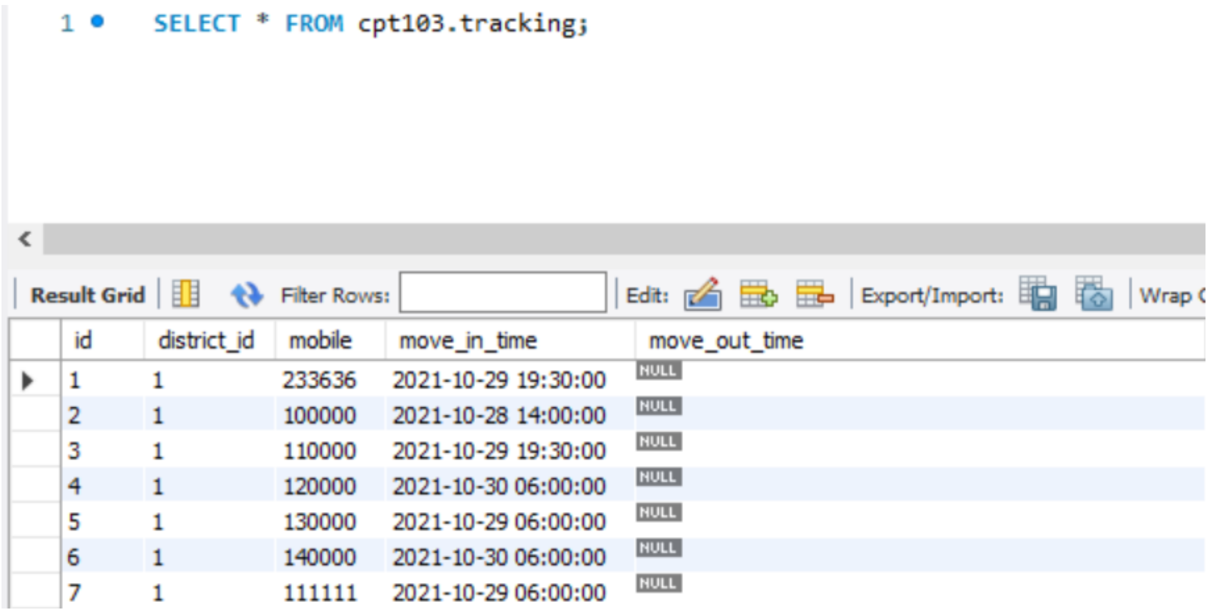
```
SELECT
  (SELECT
      COUNT(gender) total_num
    FROM
      citizen,
      tracking,
      district
    WHERE
      citizen.gender = 1
        AND tracking.district_id = district.id
        AND tracking.mobile = citizen.mobile
        AND district.district_name = 'Centre Lukewarm Hillside') / (SELECT
      COUNT(gender) total_num
    FROM
```

```
        citizen,
        tracking,
        district
    WHERE
        citizen.gender = 0
            AND tracking.district_id = district.id
            AND tracking.mobile = citizen.mobile
            AND district.district_name = 'Centre Lukewarm Hillside') `gender ratio`
```

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

id	name	age	gender	mobile
1	Mark	20	1	233636
2	AA	50	1	100000
3	AB	40	1	110000
4	AC	30	1	120000
5	AD	15	1	130000
6	BA	20	0	140000
7	BB	30	0	111111

(Figure 1)



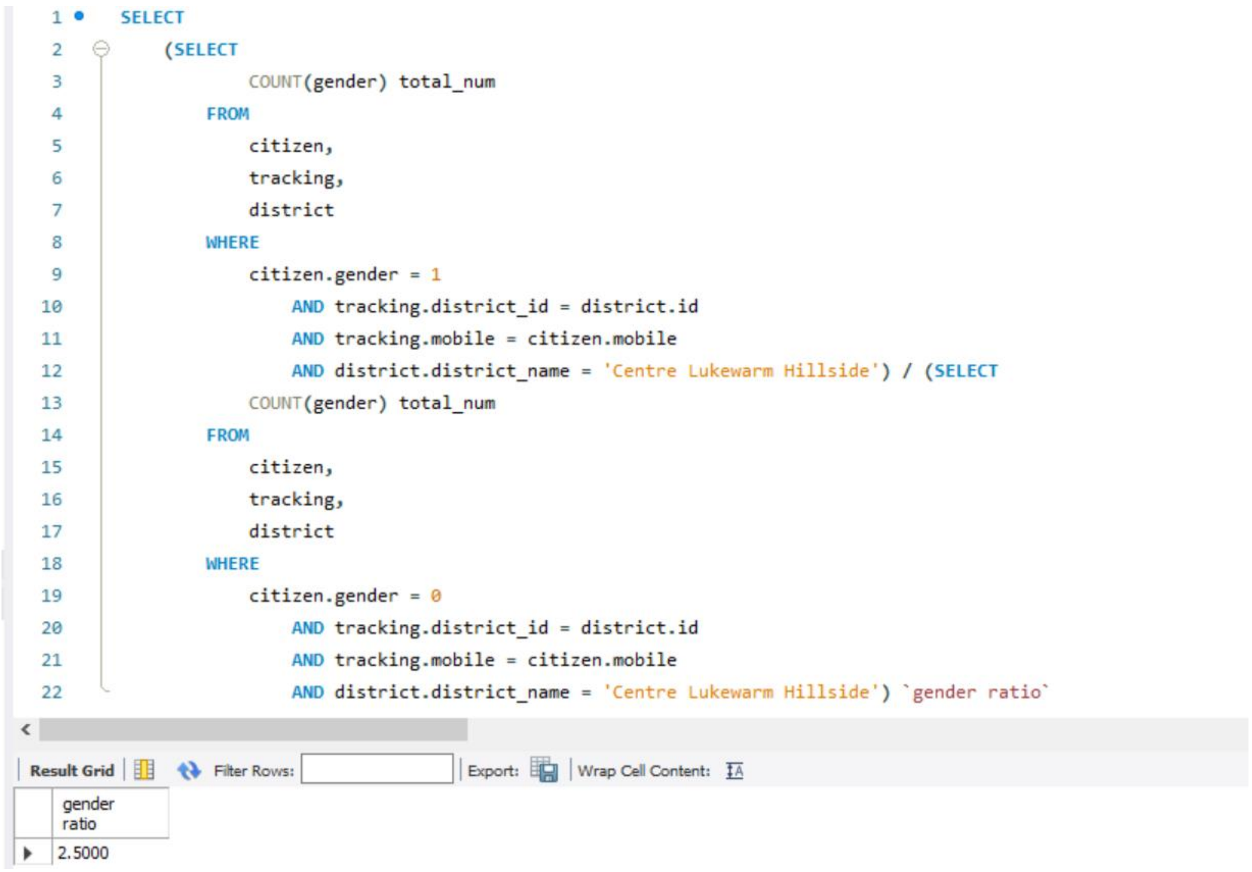
```
SELECT * FROM cpt103.tracking;
```

id	district_id	mobile	move_in_time	move_out_time
1	1	233636	2021-10-29 19:30:00	NULL
2	1	100000	2021-10-28 14:00:00	NULL
3	1	110000	2021-10-29 19:30:00	NULL
4	1	120000	2021-10-30 06:00:00	NULL
5	1	130000	2021-10-29 06:00:00	NULL
6	1	140000	2021-10-30 06:00:00	NULL
7	1	111111	2021-10-29 06:00:00	NULL

(Figure 2)

In order to simulate this condition, supposed that only 7 citizens in this case in Figure 1. Then, I truncate table tracking first and insert the following data in Figure 2, and supposed that 7 citizens did not went out of district “Centre Lukewarm Hillside”. Therefore, all data in Figure 2 should be counted, and the result is Male/Female = 5/2.

The result of the SELECT statement (screenshot):



```
SELECT
    (SELECT
            COUNT(gender) total_num
        FROM
            citizen,
            tracking,
            district
        WHERE
            citizen.gender = 1
                AND tracking.district_id = district.id
                AND tracking.mobile = citizen.mobile
                AND district.district_name = 'Centre Lukewarm Hillside') / (SELECT
            COUNT(gender) total_num
        FROM
            citizen,
            tracking,
            district
        WHERE
            citizen.gender = 0
                AND tracking.district_id = district.id
                AND tracking.mobile = citizen.mobile
                AND district.district_name = 'Centre Lukewarm Hillside') `gender ratio`
```

gender ratio
2.5000

Use case 4:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):

Use case 5:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):

Use case 6:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):

Use case 7:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):

Use case 8:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):

Use case 9:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):

Use case 10:

Your SQL statement:

Your test data and why it can prove that the SELECT statement works (Important! Please explain carefully):

The result of the SELECT statement (screenshot):