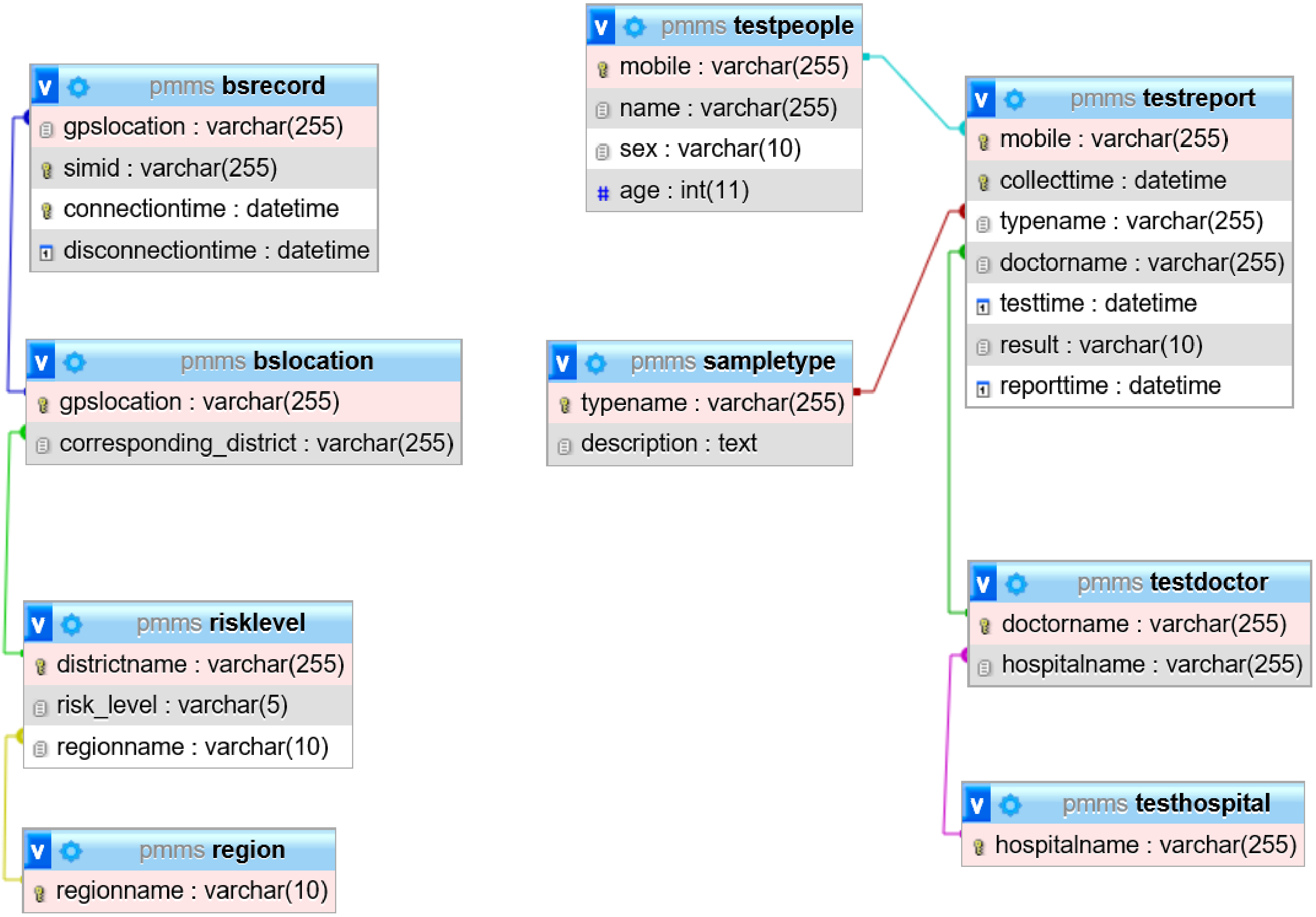
Your ID: 2036328

Name: Yuzhen Chen

Email Address: Yuzhen.CHEN20@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.



pmms **bsrecord**
- gpslocation : varchar(255)
- simid : varchar(255)
- connectiontime : datetime
- disconnectiontime : datetime

pmms **bslocation**
- gpslocation : varchar(255)
- corresponding_district : varchar(255)

pmms **risklevel**
- districtname : varchar(255)
- risk_level : varchar(5)
- regionname : varchar(10)

pmms **region**
- regionname : varchar(10)

pmms **testpeople**
- mobile : varchar(255)
- name : varchar(255)
- sex : varchar(10)
- age : int(11)

pmms **sampletype**
- typename : varchar(255)
- description : text

pmms **testreport**
- mobile : varchar(255)
- collecttime : datetime
- typename : varchar(255)
- doctorname : varchar(255)
- testtime : datetime
- result : varchar(10)
- reporttime : datetime

pmms **testdoctor**
- doctorname : varchar(255)
- hospitalname : varchar(255)

pmms **testhospital**
- hospitalname : varchar(255)

Table name: **region**

Table design explanation:

**This table is used only to store all names of five regions in the Lukewarm Kingdom. The primary key is regionname as every region in the Lukewarm Kingdom is unique.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **regionname varchar(255) primary key** | **All valid region names are acceptable. For example, 'north', 'south'** | **The name of every region. As the requirement in coursework mentioned, there are only five different regions in the Lukewarm Kingdom. So the value should be 'north', 'south', 'east', 'west' and 'central', this cannot be checked directly by the database, so manual check is required when entering data.** |

Foreign keys and reasons:

**No foreign keys in this table.**

Table name: **risklevel**

Table design explanation:

**This table is used to store all information of the districts in the Lukewarm Kingdom. The primary key is districtname as every district is unique in the Lukewarm Kingdom and in real life cases, generally, the name of every different district in one country do not repeat.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **districtname varchar(255) primary key** | **All valid district names are acceptable. For example, 'Lenny town', 'Glow Sand district'.** | **The name of existing district in Lukewarm Kingdom.** |
| **risk_level varchar(5) check(risk_level in('high','mid','low'))** | **The risklevel should either be 'high' or 'mid' or 'low'.** | **The risk level of every district in this country. As required in the coursework, there are only three risk levels. This domain is checked by the domain constraint added in this column.<br>This column can be null. If the risk level is not sure, then null can be filled in as a temporary value.** |
| **regionname varchar(10) not null** | **All valid region names are acceptable. For example, 'north', 'south'** | **The region where a district belongs to.<br>This column cannot be null as every district must have a corresponding region.** |

Foreign keys and reasons:

**The column regionname references region.regionname, which is to make sure the region name entered in this table match the correct region name in the Lukewarm Kingdom. It corresponds the 1:M relationship of regions and districts in the ER diagram.**

Table name: **bslocation**

Table design explanation:

**This table is used to store the location information of all base stations in the Lukewarm Kingdom. The primary key is gpslocation as the location of a base station is unique so that different GPS data can match different base station in the Lukewarm Kingdom.**

**The format of GPS location is found in this website:**

**https://stackoverflow.com/questions/614215/storing-gps-locations-in-a-database-varchar-field**

**This GPS format consists of latitude and longitude to represent a geographical location.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **gpslocation varchar(255) primary key** | **Valid GPS location is 'XXX.YYY, AAA.BBB'. For example,**<br>**'53.445, 89.7577'**<br>**'77, 23.692'** | **The GPS location of a base station. The datatype of this column is varchar, which means the GPS data are stored as chars. XXX.YYY represents the latitude of the base station, followed by a comma and one space, then AAA.BBB represents the longitude of the base station.**<br>**This format cannot be directly checked by the database. As a result, manual check is required when entering data.** |
| **corresponding_district varchar(255) not null** | **All valid district names which are the real existing district name in the country are acceptable. For example,**<br>**'lenny town'.** | **The district that a base station locates in.**<br>**This column cannot be null as a base station must belong to a district in the Lukewarm Kingdom.** |

Foreign keys and reasons:

**The column corresponding_district references risklevel.districtname, which makes sure that the district name in this table matches the real and correct district name in Lukewarm Kingdom. it corresponds to the 1:M relationship of district and base station in the ER diagram as in real life cases there are multiple base stations in one district.**

Table name: **bsrecord**

Table design explanation:

**This table is used to store the traveling information of the people who 'is connecting' or 'connected' to the base station. The primary key is the simid(sim card id, which is the mobile number of someone) and connectiontime as a person connecting to the base station at some point is unique.**

**The GPS location has been illustrated in the bslocation table.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **gpslocation varchar(255)** | **Valid GPS location is 'XXX.YYY, AAA.BBB'. For example,<br>'53.445, 89.7577'<br>'77, 23.692'** | **The GPS location of a base station.<br>In this column, the data entered can be checked by the database as there is a foreign key which will be illustrated below.** |
| **simid varchar(255)** | **All valid phone numbers in the Lukewarm Kingdom are acceptable. For example,<br>'233636'.** | **The mobile number of a person who 'is connecting' or 'connected' to the base station.** |
| **connectiontime datetime** | **All valid time is acceptable. For example,<br>'2021-10-1 19:00:00'** | **The connection time of a person enters into the range of a base station.** |
| **disconnectiontime datetime** | **All valid time is acceptable. For example,<br>'2021-10-1 19:00:00'** | **The disconnection time of a person when he or she moves out of the range of a base station.<br>This column can be null. Someone may stay in a fixed area of daily activity and do not move out of this area for a long time, in this way there will be no disconnection time since this person connected to this base station. And the data will be updated with real disconnection time when someday the person moves out of the range of the base station.<br>The disconnection time should be later than the connection time and this cannot be checked directly by this database, it should be manually checked when entering data.** |

Foreign keys and reasons:

**The column gpslocation references bslocation.gpslocation, which makes sure that the location format of a base station is valid that matches the real and correct location in the Lukewarm Kingdom. It corresponds to the 1:1 relationship of a base station and district in the ER diagram. One base station locates in exactly one district and one base station only serves one district at the same time.**

Table name: **testhospital**

Table design explanation:

**This table is used to store the information of all hospitals which are capable of doing the virus tests. The primary key is hospitalname. But in real life cases, one hospital may have different branches in different districts, which causes hospitals locates in different area have the same name. Under this circumstance, I add numbers to the hospital name to identify branches of the same hospital in different districts, which can make every hospital name unique and every hospital can be seen as unique.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **hospitalname varchar(255) primary key** | **All valid hosipital names are acceptable. For example, 'Central hospital'.** | **The name of the hospital which can do the virus tests.** |
| **districtname varchar(255)<br>not null** | **All valid district names which are the real existing district name in the country are acceptable. For example, 'lenny town'.** | **The district that the hospital locates in.<br>The district name is not null as every hospital locates in a corresponding district.** |

Foreign keys and reasons:

**The column districtname references risklevel.districtname, which makes sure that the names of the districts in this table matches the real existing districts in Lukewarm Kingdom. it corresponds to the 1:1 relationship of hospital and district in the ER diagram. One hospital locates in exactly one districts.**

Table name: **sampletype**

Table design explanation:

**This table is used to store the information of different sample types. The primary key is typename as it is unique for every sample type.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **typename varchar(255) primary key** | **All valid virus names are acceptable. For example, 'COVID-19'.** | **The name of a specific virus needs to be tested to identify whether a person is infected by this virus.** |
| **description text not null** | **All text information can be acceptable. For example, 'all people tested to be positive should rest well'.** | **The footnote of sample type. It is a clear and short description for the sample type.<br>This column is not null as every sample type needs a description text on the virus test report.** |

Foreign keys and reasons:

**No foreign keys in this table.**

Table name: **testpeople**

Table design explanation:

**This table is used to store the brief and basic information of all people who did the virus test, not storing all people in this country. The primary key is the mobile phone number as every citizen must has a unique telephone number.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **mobile varchar(255) primary key** | **All valid phone numbers in the Lukewarm Kingdom are acceptable. For example, '233636'.** | **The mobile number of a person who gets the virus test.** |
| **name varchar(255) not null** | **All valid names of people are acceptable. For example, 'James', 'Jason'** | **The name of a person who gets the virus test. Before doing the virus tests, some personal information is registered by the hospital which includes phone number, name, gender and age. So this column cannot be null as the person's name is needed before collecting the sample.**<br>**This column is not unique as two different people maybe have the same name.** |
| **sex varchar(10) not null check(sex in ('Female','Male'))** | **the data in sex column should be either 'Female' or 'Male'.** | **The gender of a person who gets the virus test. Before doing the virus tests, some personal information is registered by the hospital which includes phone number, name, gender and age. So This column cannot be null as the person's gender is needed before collecting the sample.**<br>**The gender should either be male or female. This domain is checked by the domain constraint added to this column.** |
| **age int not null check(age >= 0)** | **All valid non-negative numbers are acceptable. For example, 12, 34.** | **The age of a person who gets the virus test. Before doing the virus tests, some personal information is registered by the hospital which includes phone number, name, gender and age. So this column cannot be null as the person's age is needed before collecting the sample.**<br>**The age is registered as a non-negative integer. This domain is checked by the domain constraint added to this column. Age 0 represents the baby under 1 year old.** |

Foreign keys and reasons:

**There are no foreign keys in this table.**

Table name: **testdoctor**

Table design explanation:

**This table is used to store the information of the doctors who collect the samples when doing the virus test. The primary key is doctorname as the name is the only attribute which can identify different doctors though two doctors may have the same name. I add numbers to the name when encountering two different doctors who have the same name to identify two doctors, under this circumstance the name is unique for the doctors.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **doctorname varchar(255) primary key** | **all valid name can be acceptable. For example,<br>'Lilly','Mina'.** | **The name of a doctor who collects the sample in the hospital.** |
| **hospitalname varchar(255) not null** | **All valid hospital names which are the real existing hospital names in the Lukewarm Kingdom can be acceptable. For example,<br>'Central hospital'.** | **The name of the hospital that the doctors stored in this table work in.<br>This column cannot be null as a doctor who collects the sample only works in one corresponding hospital.** |

Foreign keys and reasons:

**The column hospitalname references testhospital.hosiptalname, which makes sure that the hospital name matches the real and existing hospital in the Lukewarm Kingdom. It corresponds to the 1:M relationship of hospitals and test doctors in the ER diagram. In one hospital, there are multiple doctors who are capable of collect the samples.**

Table name: **testreport**

Table design explanation:

**This table is used to store the information of a person's virus test report. The primary key is mobile and collecttime as it is unique for a person to do the virus test at some point.**

| Column Definition | Domain | Explanation |
|---|---|---|
| **mobile varchar(255)** | **All valid phone numbers in the Lukewarm Kingdom are acceptable. For example, '233636'.** | **The mobile number of a person who gets the virus test.<br>This column is not unique as the same person may do the virus test multiple times, but mobile and collecttime is unique.** |
| **collecttime datetime** | **All valid time is acceptable. For example, '2021-10-09 10:59:00'** | **The sample collect time of a person who did the test.<br>This column is not unique as at a moment there are multiple people doing the test, but mobile and collecttime is unique.** |
| **typename varchar(255) not null** | **All the valid virus type names are acceptable. For example, 'COVID-19'** | **The name of a specific virus needs to be tested to identify whether a person is infected by this virus.** |
| **doctorname varchar(255)** | **All valid names of the doctors who collect samples in the hospital are acceptable. For example, 'Lilly'.** | **The name of a doctor who collects virus sample for people in the hospital.<br>This column can be null as the report generated with the doctor's name after the result is confirmed (the coursework has mentioned this).** |
| **testtime datetime** | **All valid time is acceptable. For example, '2021-10-09 13:43:00'** | **The test time of the collected sample.<br>the sample cannot be tested immediately after the doctor collects the samples of people, it takes time to analyse and test the samples. The whole report generates after the results confirmed. So this column can be null.** |
| **result varchar(10) check(result in('positive','negative'))** | **The result is either 'positive' or 'negative'.** | **The results of the sample tests.<br>the sample cannot be tested immediately after the doctor collects the samples of people, it takes time to analyse and test the samples. The whole report generates after the results confirmed. So this column can be null.<br>The test result is either positive or negative. Positive means get infected and negative means not. This domain can be checked by the domain constraint added to this column.** |
| **reporttime datetime** | **All valid time is acceptable. For example, '2021-10-09 15:48:00'** | **The time that the report generates.<br>the sample cannot be tested immediately after the doctor collects the samples of people, it takes time to analyse and test the samples. The whole report generates after the results confirmed. So this column can be null.** |

Foreign keys and reasons:

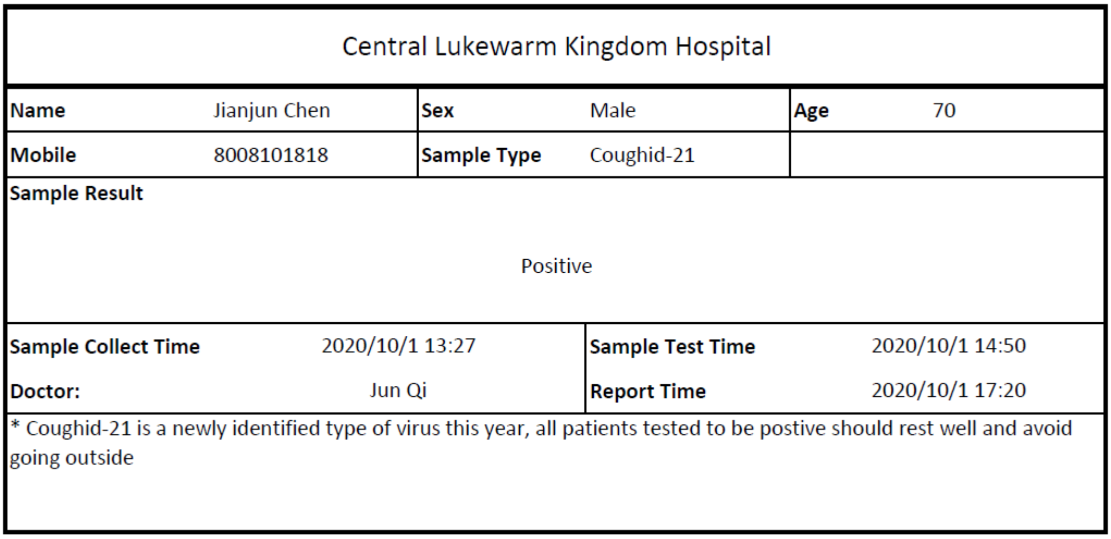
**There are three foreign keys added to this table.**

**The first one: The column mobile references testpeople.mobile, which makes sure that the mobile number in the report matches the mobile number of the people who registered to do the virus test. It corresponds to the 1:1 relationship of the mobile number and a person in the ER diagram. A person must have a unique mobile number.**

**The second one: The column typename references sampletype.typename, which makes sure that the type name in the report matches the virus test that the hospital are able to do. It corresponds to the M:1 relationship of the test type and a person in the ER diagram. One person can be tested different types of viruses to identify whether infected or not.**

**The third one: The column doctorname references testdoctor.doctorname, which makes sure that the doctor generated in the report matches the doctors who can collect the samples in the hospital. It corresponds to the 1:1 relationship of the doctor and a person at some point the ER diagram. At some point one person only have one doctor to collect his or her sample.**

# 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:

**Mobile, collect time, name, sex, age, doctor, hospital, test time, result, report time, sample type, sample description**

FDs (Indicate partial or transitive dependencies):

**The primary key is mobile and collect time as these two attributes determine other attributes.**

**In 1NF, it still has partial dependencies and transitive dependencies and these two dependencies do not need to be removed in 1NF.**

Normalised tables and which normal form they are currently in:

**Table1:**

| **PK** | **PK** | **Column1** | **Column2** | **Column3** | **Column4** | **Column5** | **Column6** | **Column7** | **Column8** | **Column9** | **Column 10** |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **mobile** | **collect time** | **name** | **sex** | **age** | **doctor** | **hospital** | **test time** | **result** | **report time** | **sample type** | **sample description** |

**The current form is 1NF as there are still partial dependencies and transitive dependencies.**

**The table still need to be normalised**

Stage 2

Attributes:

**Mobile, collect time, name, sex, age, doctor, hospital, test time, result, report time, sample type, sample description**

FDs (Indicate partial or transitive dependencies):

**In 2NF, partial dependencies need to be removed.**

**In 2NF, transitive dependencies do not need to be removed.**

**Partial dependencies:**

**mobile determines name, sex, age (the mobile is unique for the citizens in the Lukewarm Kingdom, one mobile phone matches one specific person and one person only has unique name, gender and age)**

**mobile and collect time determine doctor, hospital, test time, result, report time, sample type, sample description. (a person to do a specific virus test at some point, the doctor who collects sample for him, the hospital he chooses to do the test, test time, result, report time, the sample type he chooses to test, sample description are all unique. So mobile and collect time determine doctor, hospital, test time, result, report time, sample type, sample description)**

Normalised tables and which normal form they are currently in:

**Table 1:**

| **PK** | **Column1** | **Column2** | **Column3** |
|---|---|---|---|
| **mobile** | **name** | **sex** | **age** |

**Table 2:**

| **PK** | **PK** | **Column1** | **Column2** | **Column3** | **Column4** | **Column5** | **Column6** | **Column7** |
|---|---|---|---|---|---|---|---|---|
| **mobile** | **Collet time** | **doctor** | **hospital** | **test time** | **result** | **report time** | **sample type** | **description** |

**The tables are in 2NF as there are no partial dependencies but there still are transitive dependencies.**

**The tables still need to be normalised.**

Stage 3

Attributes:

**Mobile, collect time, name, sex, age, doctor, hospital, test time, result, report time, sample type, sample description**

FDs (Indicate partial or transitive dependencies):

**The partial dependencies are all removed in stage 2.**

**In 3NF, all the transitive dependencies need to be removed.**

**The transitive dependencies:**

**Doctor determines hospital. (A doctor only woks in one hospital at the same time, but one hospital has multiple doctors who can collect samples. One doctor who collects the sample matches the a unique hospital in the country. So, doctor determines hospital)**

**Sample type determines sample description. (As the coursework requirement mentioned that all samples of the same type will have the same description, which means one type has the same description. So the sample type determines the description)**

Normalised tables and which normal form they are currently in:

**Table 1:**

| **PK** | **Column1** | **Column2** | **Column3** |
|---|---|---|---|
| **mobile** | **name** | **sex** | **Age** |

**Table 2:**

| **PK** | **PK** | **Column1** | **Column2** | **Column3** | **Column4** | **Column5** |
|---|---|---|---|---|---|---|
| **mobile** | **collect time** | **doctor** | **test time** | **result** | **report time** | **sample type** |

**Table 3:**

| **PK** | **Column1** |
|---|---|
| **doctor** | **hospital** |

**Table 4:**

| **PK** | **Column1** |
|---|---|
| **sample type** | **description** |

**The tables are in 3NF as all partial and transitive dependencies are removed.**

**The normalisation process is finished.**

# Use Cases

***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. 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 t.simid from

(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r
WHERE

l.corresponding_district in

(select l.corresponding_district from bsrecord as r inner join bslocation as l ON

r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t

where

(t.connectiontime > '2021-10-07 19:30:00' or

t.disconnectiontime is null OR

t.disconnectiontime > '2021-10-07 19:30:00') and t.simid <> '233636'
```

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

**The following travel information of people living in this country is added to the bsrecord table.**

| gpslocation | simid | connectiontime | disconnectiontime |
|---|---|---|---|
| ‘1,1’ | ‘123450’ | ‘2021-01-01 00:00:00’ | Null |
| ‘2,2’ | ‘123451’ | ‘2021-10-01 08:00:00’ | Null |
| ‘2,2’ | ‘123452’ | ‘2021-10-01 08:00:00’ | Null |
| ‘1,1’ | ‘123453’ | ‘2021-10-01 00:00:00’ | Null |
| ‘2,2’ | ‘123454’ | ‘2021-10-08 11:00:00’ | Null |
| ‘3,3’ | ‘123455’ | ‘2021-10-06 15:00:00’ | Null |
| ‘3,3’ | ‘233636’ | ‘2021-01-01 00:00:00’ | Null |
| ‘5,5’ | ‘123456’ | ‘2021-09-28 08:00:00’ | Null |
| ’11,11’ | ‘123457’ | ‘2021-10-01 09:00:00’ | ‘2021-10-08 09:00:00’ |
| ‘7,7’ | ‘123457’ | ‘2021-10-08 09:00:00’ | Null |

**The corresponding GPS location data above have been added to the bslocation table. Other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**The test data set contains people who stayed in the same district as Mark and people who did not stay in the same district with mark during last 48 hours. The expected result of the query should only contain the mobile numbers of people who stayed in the same district as Mark during last 48 hours. People during last 48 hours who did not stay in the same district as Mark should be properly filtered out.**

**During last 48 hours, Mark probably stayed in one place or Mark went to other places. In this data set, the case is that Mark stayed in one place during last 48 hours. The other case has been already tested, the select query can also work.**

**The detailed illustrations are as follows:**

**1.There are multiple base stations in one district. During last 48 hours, Mark maybe stayed in his living district or maybe he went to some other places. So, the first step is to select all base station that Mark has connected to. Then all target base stations are selected, the corresponding districts that Mark has been to can be selected.**

```
select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation
```

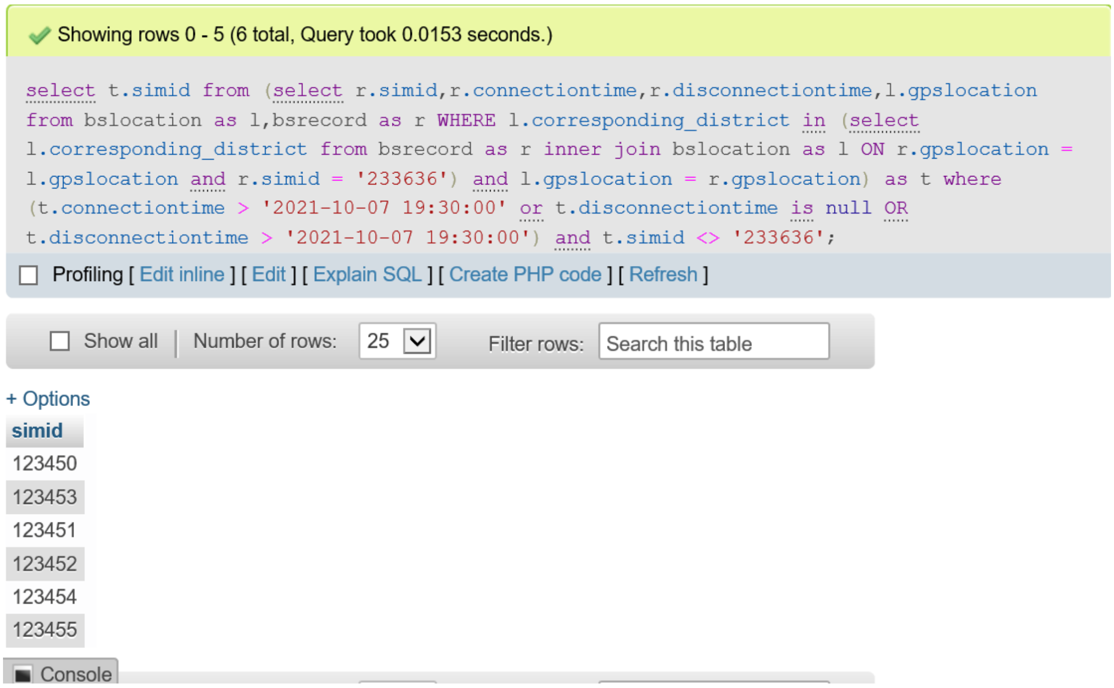
**2. As all the districts that Mark has been to are selected. Then all the mobile numbers of people, connection time, disconnection time need to be selected, which selected all necessary information of these people who have ever been to the same districts as Mark.**

```
select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation
```

**3. As all the mobile numbers of people who have ever been to the same district as Mark during last 48 hours have been selected. The final step is to select all the people who has been to those districts only during last 48 hours, which requires that the connection time after 2021-10-07 19:30:00 or the people did not leave the corresponding district which means the disconnection time is null or the people just left the corresponding district after 2021-10-07 19:30:00. So, the final select query is as follows:**

```
select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00') and t.simid <> '233636'
```

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 5 (6 total, Query took 0.0153 seconds.)

```
select t.simid from (select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation
from bslocation as l,bsrecord as r WHERE l.corresponding_district in (select
l.corresponding_district from bsrecord as r inner join bslocation as l ON r.gpslocation =
l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t where
(t.connectiontime > '2021-10-07 19:30:00' or t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00') and t.simid <> '233636';
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all
Number of rows: 25
Filter rows: Search this table
+ Options
Console

| simid |
|---|
| 123450 |
| 123453 |
| 123451 |
| 123452 |
| 123454 |
| 123455 |

***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):

**This GPS format is a varchar consisting of latitude and longitude to represent a geographical location.**

**The website link:**

**https://stackoverflow.com/questions/614215/storing-gps-locations-in-a-database-varchar-field**

Your SQL statement(s) for travel record insertion:

**insert into bsrecord values ('2,2','233636','2021-01-01 00:00:00',null)**

**update bsrecord set disconnectiontime = '2021-10-01 08:00:00' where simid = 233636 and connectiontime = '2021-01-01 00:00:00'**

**insert into bsrecord values ('3,3','233636','2021-10-01 08:00:00',null)**

**update bsrecord set disconnectiontime = '2021-10-01 09:00:00' where simid = 233636 and connectiontime = '2021-10-01 08:00:00'**

**insert into bsrecord values ('2,2','233636','2021-10-01 09:00:00',null)**

**select l.corresponding_district,r.gpslocation,r.connectiontime,r.disconnectiontime from bsrecord as r, bslocation as l**

**WHERE r.simid = '233636' and l.gpslocation = r.gpslocation**

the explanation of the test data:

the detailed process of what happens to my database when a user moves into the range of a base station and then moves out one hour later is as follows:

Assume a person named Mark (his mobile number is 233636) lives in district a, he left the range of one base station in district a on '2021-10-01 08:00:00' and then moved into the other range of the base station in district a, finally one hour later he came back to where he lived (the area covered by the initial base station).

1. as Mark initially lived in district a, his data is recorded in the table bsrecord

insert into bsrecord values ('2,2','233636','2021-01-01 00:00:00',null)

2. then he moved out of the range of the initial base station on ‘2021-10-01 08:00:00’, then the disconnection time would be updated with the real data. As the moment he left one base station, he entered into the other base station. His new travel data would also be recorded in the database.

update bsrecord set disconnectiontime = '2021-10-01 08:00:00' where simid = 233636 and connectiontime = '2021-01-01 00:00:00'

insert into bsrecord values ('3,3','233636','2021-10-01 08:00:00',null)

3.final step: one hour later he came back to the area of the initial base station, which means he would move out of a base station so that the disconnection time would be updated with the real data. At the moment he disconnected, he would connect to the initial base station. His new travel data would be recorded in the database.

update bsrecord set disconnectiontime = '2021-10-01 09:00:00' where simid = 233636 and connectiontime = '2021-10-01 08:00:00'

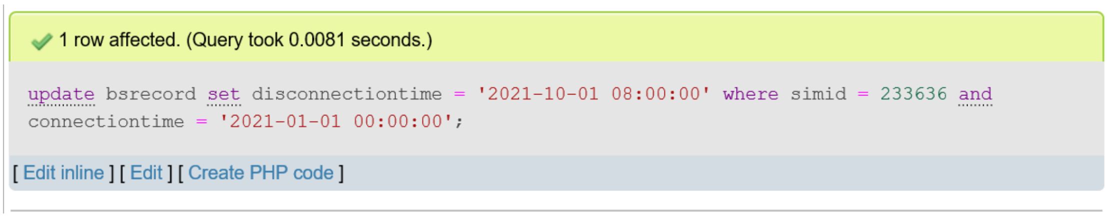
insert into bsrecord values ('2,2','233636','2021-10-01 09:00:00',null)

the select query help to find the all the travel history of Mark during last one hour:

select l.corresponding_district,r.gpslocation,r.connectiontime,r.disconnectiontime from bsrecord as r, bslocation as l

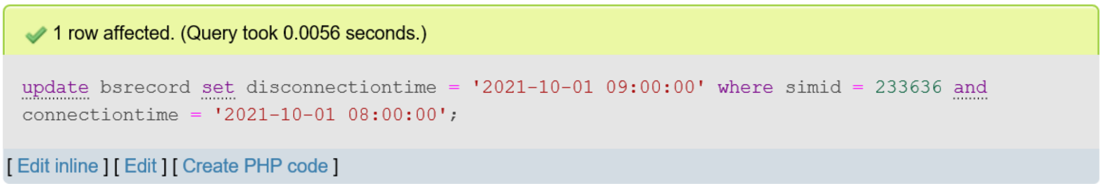
WHERE r.simid = '233636' and l.gpslocation = r.gpslocation

The result of the SELECT statements (screenshot):

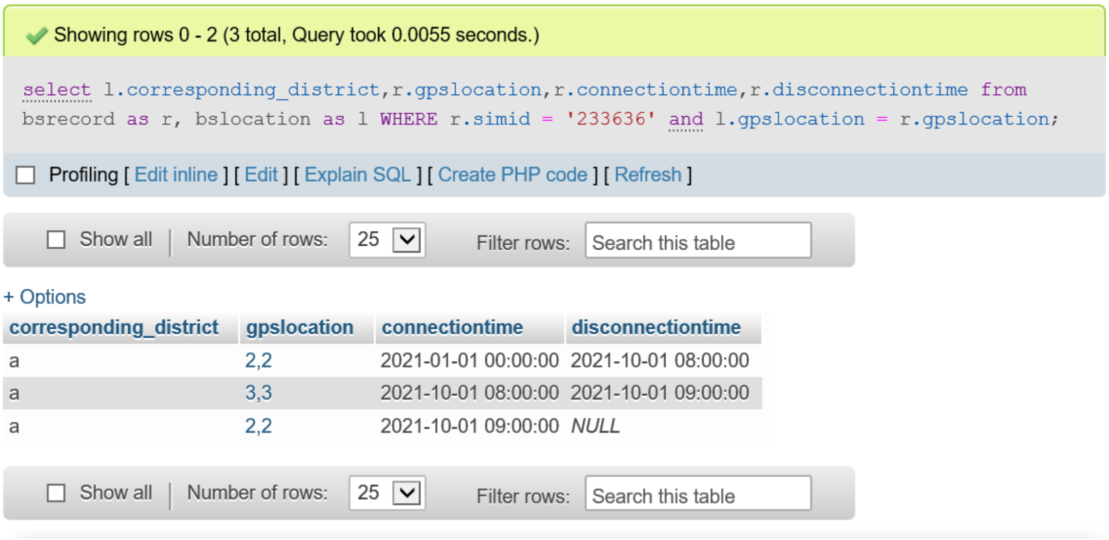


✓ 1 row affected. (Query took 0.0081 seconds.)

```
update bsrecord set disconnectiontime = '2021-10-01 08:00:00' where simid = 233636 and
connectiontime = '2021-01-01 00:00:00';
```


✓ 1 row affected. (Query took 0.0056 seconds.)

```
update bsrecord set disconnectiontime = '2021-10-01 09:00:00' where simid = 233636 and
connectiontime = '2021-10-01 08:00:00';
```


✓ Showing rows 0 - 2 (3 total, Query took 0.0055 seconds.)

```
select l.corresponding_district,r.gpslocation,r.connectiontime,r.disconnectiontime from
bsrecord as r, bslocation as l WHERE r.simid = '233636' and l.gpslocation = r.gpslocation;
```

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
☐ Show all | Number of rows: 25 Filter rows: Search this table
+ Options

| corresponding_district | gpslocation | connectiontime | disconnectiontime |
|---|---|---|---|
| a | 2,2 | 2021-01-01 00:00:00 | 2021-10-01 08:00:00 |
| a | 3,3 | 2021-10-01 08:00:00 | 2021-10-01 09:00:00 |
| a | 2,2 | 2021-10-01 09:00:00 | *NULL* |

☐ Show all | Number of rows: 25 Filter rows: Search this table

***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 hospitalname from**

**(select hospitalname,AVG(doctor_averagetime) as hospital_averagetime from**

**testdoctor natural join**

**(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr**

**where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1**

**group by hospitalname**

**having hospital_averagetime <= ALL**

**(select AVG(doctor_averagetime) from testdoctor natural join**

**(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr**

**where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1))**

**as t2**

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

**The following information of hospitals which can do the virus test is added to the testhospital table.**

| **hospitalname** | **districtname** |
|---|---|
| **‘aa’** | **‘a’** |
| **‘bb’** | **‘b’** |

| | |
|---|---|
| ‘cc’ | ‘c’ |
| ‘dd’ | ‘d’ |

**The following virus test report information of people is added to the testreport table.**

| mobile | collecttime | typename | doctorname | testtime | result | reporttime |
|---|---|---|---|---|---|---|
| '000001' | '2021-12-1 08:00:00' | 'Coughid-21' | ‘Alen’ | '2021-12-1 10:22:00' | ‘negative’ | '2021-12-1 13:11:00' |
| '000002' | '2021-12-1 09:00:00' | 'Coughid-21' | ‘Alen’ | '2021-12-1 11:00:00' | ‘negative’ | '2021-12-1 14:00:00' |
| ‘000003’ | '2021-12-1 19:43:00' | 'Coughid-21' | ‘Amy’ | '2021-12-1 21:00:00' | null | null |
| ‘000004’ | '2021-12-1 15:55:00' | 'Coughid-21' | ‘Betty’ | '2021-12-1 17:21:00' | ‘negative’ | '2021-12-1 19:33:00' |
| ‘000005’ | '2021-12-1 11:52:00’ | 'Coughid-21' | ‘Bob’ | '2021-12-1 14:00:00' | ‘negative’ | '2021-12-1 16:43:00' |
| ‘000006’ | '2021-12-1 08:32:00' | 'Coughid-21' | ‘Cindy’ | '2021-12-1 10:00:00' | ‘negative’ | '2021-12-1 11:17:00' |
| ‘000007’ | '2021-12-1 16:59:00' | 'Coughid-21' | ‘Curry’ | '2021-12-1 19:00:00' | ‘negative’ | '2021-12-1 21:19:00' |
| ‘000008’ | '2021-12-1 21:12:00' | 'Coughid-21' | ‘Curry’ | '2021-12-2 08:11:00' | ‘negative’ | '2021-12-2 10:00:00' |
| ‘000009’ | '2021-12-1 09:45:00' | 'Coughid-21' | ‘Cindy’ | null | null | null |
| ‘000010’ | '2021-12-1 10:12:00' | 'Coughid-21' | ‘Dave’ | '2021-12-1 12:00:00' | ‘negative’ | '2021-12-1 14:53:00' |
| ‘000011’ | '2021-12-1 13:51:00' | 'Coughid-21' | ‘Dave’ | '2021-12-1 15:29:00' | ‘negative’ | '2021-12-1 18:00:00' |
| ‘000012’ | '2021-12-1 15:21:00' | 'Coughid-21' | ‘Dick’ | '2021-12-1 17:09:00' | ‘negative’ | '2021-12-1 19:00:00' |

**The corresponding mobile number, doctor information and district data above have been added to the testpeople, testdoctor and risklevel table. Other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**The test data set contains all hospitals which can do virus test in the Lukewarm Kingdom and all the viral test records of people who have did the test before (which contain all the report time and sample test time data of the corresponding citizens so that the average generation time of every hospital can be calculated). The test data set contains data that the report time or the test time is still null (The test process is impossible to be done immediately after the sample collection, which means there are some report results still pending when doing the calculation), which mimic some real-life situation. The expected result of the query should only contain the hospital that is the most efficient. Hospital that are not most efficient should be properly filtered out.**

**The detailed illustrations are as follows:**

**1.As there are multiple doctors who can collect samples in one hospital. To calculate the average report generation time, first to calculate the average report generation time of every doctor (if the average report generation time of every doctor has been calculated, the average report generation time of every hospital can be easily calculated). The test process is impossible to be done immediately after the sample collection, which means there are some report results still pending. So, the condition**

**that the test time and report time is null should be filtered out (if the result is still pending, then it cannot be calculated in the average time).**

```
SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr
where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname
```

**2.As the average report generation time of every doctor who collects samples in the Lukewarm Kingdom has already been calculated. The next step is to gain the report generation time of every hospital which is capable of doing the virus test. Obviously, the average value of the corresponding average report generation time of every doctor who works in the same hospital is the average report generation time of every single hospital.**

```
select hospitalname,AVG(doctor_averagetime) as hospital_averagetime from
testdoctor natural join
(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr
where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1
group by hospitalname
```

**3.As the average report generation time of a hospital has been gained. The next step is to find the least average time among all the hospitals. If the value of an average report generation time is smaller than or equal to all the others, then this value is the smallest, which means it is the most efficient.**

```
select hospitalname,AVG(doctor_averagetime) as hospital_averagetime from
testdoctor natural join
(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr
where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1
group by hospitalname
having hospital_averagetime <= ALL
(select AVG(doctor_averagetime) from testdoctor natural join
(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr
where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1
```

**4.As the least average time has been selected. The final step is to select the corresponding hospital which has the least average time.**

```
select hospitalname from
(select hospitalname,AVG(doctor_averagetime) as hospital_averagetime from
testdoctor natural join
(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr
where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1
group by hospitalname
```

**having hospital_averagetime <= ALL**

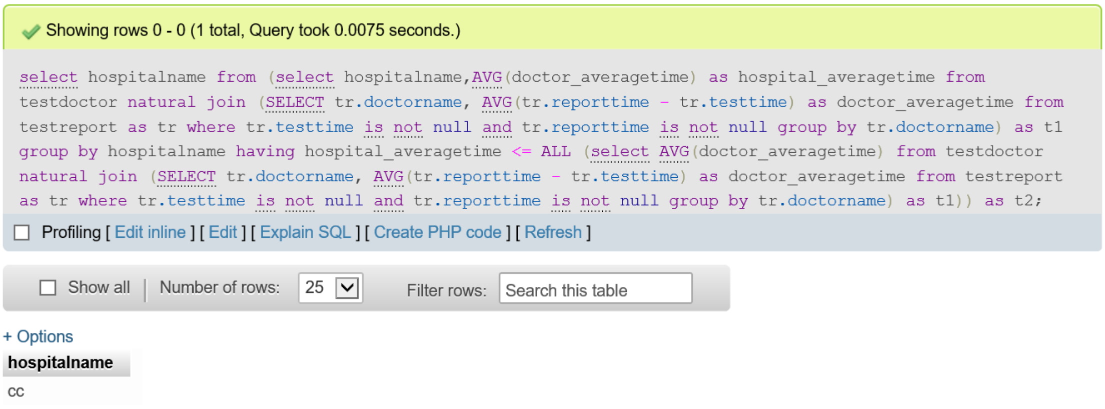
**(select AVG(doctor_averagetime) from testdoctor natural join**

**(SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport as tr**

**where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1))**

**as t2**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 0 (1 total, Query took 0.0075 seconds.)

```
select hospitalname from (select hospitalname,AVG(doctor_averagetime) as hospital_averagetime from
testdoctor natural join (SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from
testreport as tr where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1
group by hospitalname having hospital_averagetime <= ALL (select AVG(doctor_averagetime) from testdoctor
natural join (SELECT tr.doctorname, AVG(tr.reporttime - tr.testtime) as doctor_averagetime from testreport
as tr where tr.testtime is not null and tr.reporttime is not null group by tr.doctorname) as t1)) as t2;
```

☐ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
☐ Show all | Number of rows: 25
Filter rows: Search this table
+ Options

| hospitalname |
|---|
| cc |

***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:

**select mobile from**

**(select mobile,MAX(collecttime)-MIN(collecttime) as gaptime from**

**(select mobile,collecttime from testreport**

**where mobile in**

**(select t.mobile from**

**(select count(mobile) as testtimes, mobile, collecttime from testreport**

**group by mobile) as t**

**where testtimes >= 2)) as t1**

**where collecttime >= '2021-10-3 00:00:00' and collecttime <= '2021-10-5 00:00:00'**

**group by mobile**

**having gaptime >= 1000000) as t2;**

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

**The following virus test report information of people is added to the testreport table (some attributes are hidden as they are not related to this task)**

| mobile | collecttime |
|---|---|
| '000001' | '2021-12-1 08:00:00' |
| '000002' | '2021-12-1 09:00:00' |
| ‘000003’ | '2021-12-1 19:43:00' |
| ‘000004’ | '2021-12-1 15:55:00' |
| ‘000005’ | '2021-12-1 11:52:00’ |
| ‘000006’ | '2021-12-1 08:32:00' |
| ‘000007’ | '2021-12-1 16:59:00' |
| ‘000008’ | '2021-12-1 21:12:00' |
| ‘000009’ | '2021-12-1 09:45:00' |
| ‘000010’ | '2021-12-1 10:12:00' |
| ‘000011’ | '2021-12-1 13:51:00' |
| ‘000012’ | '2021-12-1 15:21:00' |
| ‘000013’ | ‘2021-10-3 08:00:00’ |
| ‘000013’ | ‘2021-10-4 10:00:00’ |
| ‘000014’ | ‘2021-10-3 11:09:00’ |
| ‘000014’ | ‘2021-10-4 08:28:00’ |
| ‘000015’ | ‘2021-10-4 10:00:00’ |
| ‘000016’ | ‘2021-10-3 11:00:00’ |
| ‘000017’ | ‘2021-10-4 12:00:00’ |
| ‘000018’ | ‘2021-10-3 11:00:00’ |
| ‘000019’ | ‘2021-10-3 11:00:59’ |
| ‘000020’ | ‘2021-10-3 11:1:47’ |

The corresponding mobile number data above have been added to the testpeople table. Other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.

The test data contains the viral test report data of people in Lukewarm Kingdom, which means the test data set contains people who did the virus test from ‘2021-10-3 00:00’ to ‘2021-10-5 00:00’ and the people who did virus test in other time. The test data also contains people who did only one test during the target time and people who did two or more times during the target time. The expected result of the query only contains people who did two viral tests having a gap time of at least 24 hours during the target time. Other people who failed to satisfy this condition should be filtered out.

The detailed illustrations are as follows:

1.the use case requires that the people should do two viral tests which have a gap time of at least 24 hours, which means we need to select the people who did at least two times (if a person did more than two times and there exists two times which have gap time of at least 24 hours, then we can also say he did two viral tests having a gap time of at least 24 hours). The first step is to count the test times of every people who did the tests.

```
select count(mobile) as testtimes, mobile, collecttime from testreport
group by mobile
```

2. As the test number has been already calculated. The people who did more than one times need to be selected.

```
select mobile,collecttime from testreport
where mobile in
(select t.mobile from
(select count(mobile) as testtimes, mobile, collecttime from testreport
group by mobile) as t
where testtimes >= 2
```

3.As all people who did at least two times tests are selected, the next step is to select all people among them who did the test during the target time.

```
select mobile,collecttime from testreport
where mobile in
(select t.mobile from
(select count(mobile) as testtimes, mobile, collecttime from testreport
group by mobile) as t
where testtimes >= 2)) as t1
where collecttime >= '2021-10-3 00:00:00' and collecttime <= '2021-10-5 00:00:00'
```

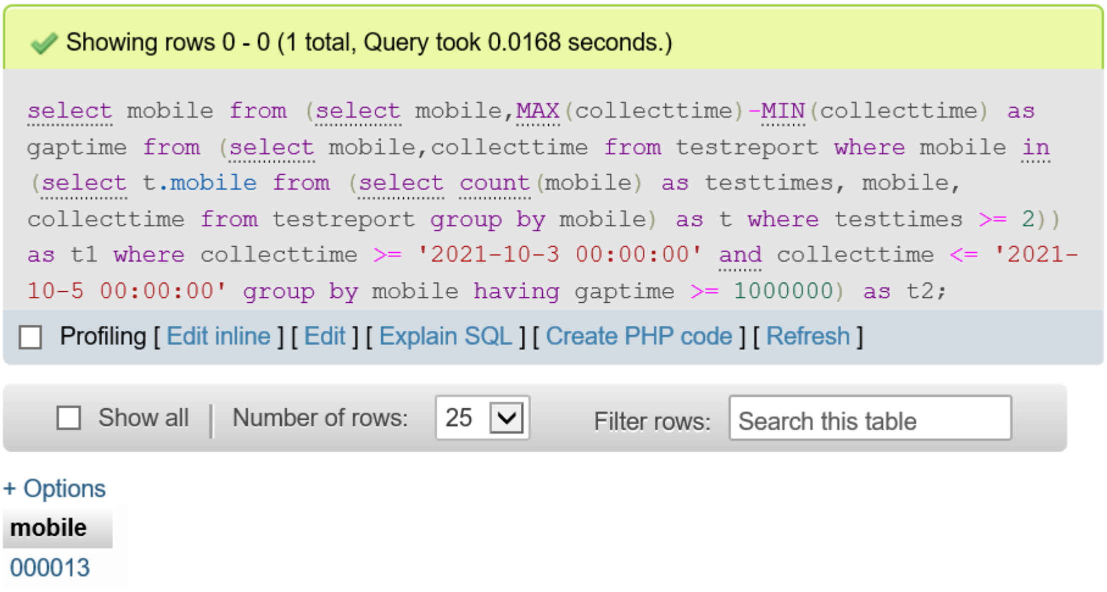
4.As all people who did more than one time and did the test during the target time are selected, next step is to select the people among them who did two viral test having a gap time of 24 hours. We use the ‘group by’ to the mobile column to calculate max(collectime) – min(collectime). For people who did multiple times during the target time, if the latest date and the earliest have a gap time of at least 24 hours then we need to select this person.

```
select mobile,MAX(collecttime)-MIN(collecttime) as gaptime from
(select mobile,collecttime from testreport
where mobile in
(select t.mobile from
(select count(mobile) as testtimes, mobile, collecttime from testreport
group by mobile) as t
where testtimes >= 2)) as t1
where collecttime >= '2021-10-3 00:00:00' and collecttime <= '2021-10-5 00:00:00'
group by mobile
having gaptime >= 1000000
```

**5. the final step is to just select the mobile number from the table only containing the one who meets the requirements.**

```
select mobile from
(select mobile,MAX(collecttime)-MIN(collecttime) as gaptime from
(select mobile,collecttime from testreport
where mobile in
(select t.mobile from
(select count(mobile) as testtimes, mobile, collecttime from testreport
group by mobile) as t
where testtimes >= 2)) as t1
where collecttime >= '2021-10-3 00:00:00' and collecttime <= '2021-10-5 00:00:00'
group by mobile
having gaptime >= 1000000) as t2
```

The result of the SELECT statement (screenshot):



Showing rows 0 - 0 (1 total, Query took 0.0168 seconds.)

```
select mobile from (select mobile,MAX(collecttime)-MIN(collecttime) as
gaptime from (select mobile,collecttime from testreport where mobile in
(select t.mobile from (select count(mobile) as testtimes, mobile,
collecttime from testreport group by mobile) as t where testtimes >= 2))
as t1 where collecttime >= '2021-10-3 00:00:00' and collecttime <= '2021-
10-5 00:00:00' group by mobile having gaptime >= 1000000) as t2;
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all
Number of rows: 25
Filter rows: Search this table
+ Options

| mobile |
|---|
| 000013 |

***Use case 5***: List the high-risk, mid-risk and low-risk regions using one query. High-risk districts should be listed first, followed by mid-risk districts and then low-risk regions. 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 districtname as district_name,risk_level from risklevel**

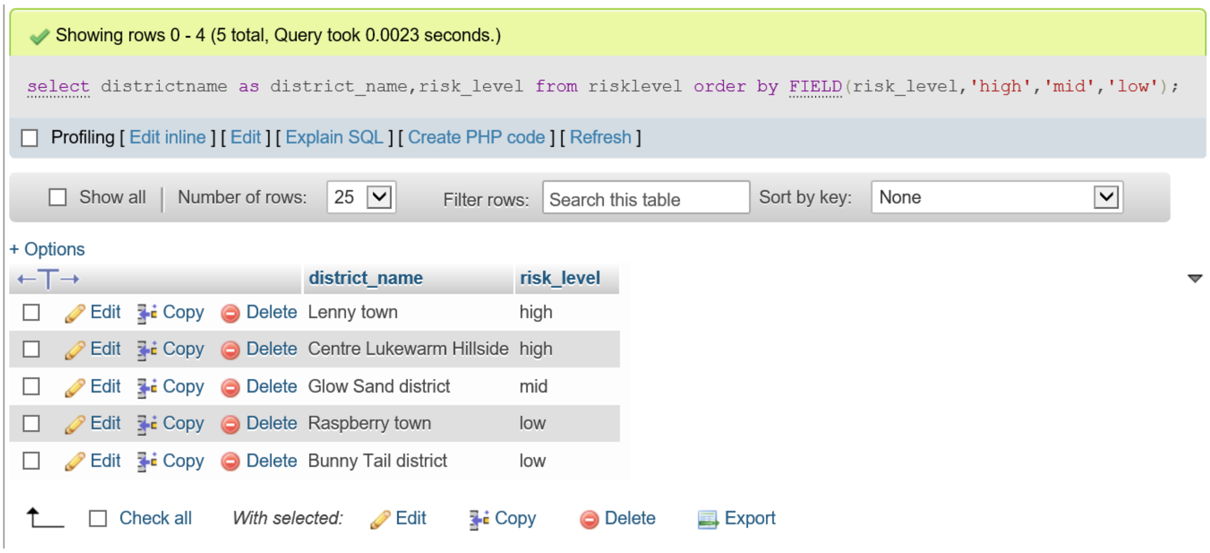
**order by FIELD(risk_level,'high','mid','low')**

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

**The data created by me is the data given above in the example. The risk-level order has already been disrupted when inserting the data.**

**In this case, the function ‘field’ is used to order in the sequence of ‘high’, ‘mid’, ‘low’.**

The result of the SELECT statement (screenshot):



Showing rows 0 - 4 (5 total, Query took 0.0023 seconds.)

```
select districtname as district_name,risk_level from risklevel order by FIELD(risk_level,'high','mid','low');
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

□ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

| | district_name | risk_level |
|---|---|---|
| □ Edit Copy Delete | Lenny town | high |
| □ Edit Copy Delete | Centre Lukewarm Hillside | high |
| □ Edit Copy Delete | Glow Sand district | mid |
| □ Edit Copy Delete | Raspberry town | low |
| □ Edit Copy Delete | Bunny Tail district | 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 name,mobile from**

**testdoctor natural join**

**(select name,r.mobile,doctorname from testpeople natural join testreport as r where**

**(r.collecttime >= '2021-10-4 00:00:00' and r.collecttime < '2021-10-5 00:00:00') and**

**r.result = 'positive') as t1**

**where**

**hospitalname in (select hospitalname from testhospital**

**where districtname = 'Centre Lukewarm Hillside')**

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

**The following personal information of people who did the viral tests is added to the testpeople table (some attributes are hidden as they are not related to this task)**

| mobile | name |
|---|---|
| ‘000001’ | ‘James’ |
| ‘000002’ | ‘Jason’ |
| ‘000003’ | ‘Jack’ |
| ‘000004’ | ‘Durant’ |
| ‘000005’ | ‘Morant’ |
| ‘000006’ | ‘Mina’ |
| ‘000007’ | ‘Lexie’ |
| ‘000008’ | ‘Cinderela’ |
| ‘000009’ | ‘Harden’ |

| | |
|---|---|
| ‘000010’ | ‘Coco’ |
| ‘000018’ | ‘Bond’ |
| ‘000019’ | ‘Lurix’ |
| ‘000020’ | ‘Lang’ |

**The following virus test report information of people is added to the testreport table (some attributes are hidden as they are not related to this task)**

| mobile | collecttime | result |
|---|---|---|
| ‘000001’ | ‘2021-10-4 08:00:00’ | ‘negative’ |
| ‘000002’ | ‘2021-10-4 09:00:00’ | ‘negative’ |
| ‘000003’ | ‘2021-10-4 19:43:00’ | ‘negative’ |
| ‘000004’ | ‘2021-10-4 15:55:00’ | ‘positive’ |
| ‘000005’ | ‘2021-10-4 11:52:00’ | ‘positive’ |
| ‘000006’ | ‘2021-10-4 08:32:00’ | ‘positive’ |
| ‘000007’ | ‘2021-10-4 16:59:00’ | ‘negative’ |
| ‘000008’ | ‘2021-10-4 21:12:00’ | ‘negative’ |
| ‘000009’ | ‘2021-10-4 09:45:00’ | ‘negative’ |
| ‘0000010’ | ‘2021-10-4 10:12:00’ | ‘negative’ |
| ‘000018’ | ‘2021-10-3 11:00:00’ | ‘negative’ |
| ‘000019’ | ‘2021-10-3 11:00:59’ | ‘negative’ |
| ‘000020’ | ‘2021-10-3 11:01:47’ | ‘negative’ |

**The following information of hospital which can do the viral test is added to the testhospital table**

| hospitalname | districtname |
|---|---|
| ‘aa’ | ‘Centre Lukewarm Hillside’ |
| ‘bb’ | ‘Centre Lukewarm Hillside’ |
| ‘cc’ | ‘Centre Lukewarm Hillside’ |
| ‘dd’ | 'Lenny town' |
| ‘ee’ | 'Lenny town' |

**Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This test data set contains the viral test report (including the result of the test of people who get the viral test) of people who are tested on ‘20210-10-04’ and people who are not tested on this day. The expected result of the query should only contain people who are tested to be positive in central Lukewarm Hillside district on ‘2021-10-04’. People who are negative or people who are positive but in other district or people who are positive but not found on ‘2021-10-04’ should be filtered out.**

**The detailed illustrations are as follows:**

**1.We want to find all positive cases in one district and only the hospitals can test people who are positive. As we need to select the positive cases in Central Lukewarm Hillside, the first step is to select all hospital which can do the viral tests in Central Lukewarm Hillside.**

```
select hospitalname from testhospital

where districtname = 'Centre Lukewarm Hillside'
```

**2.As the all hospitals capable of doing the viral test have already been selected, the final step is to select the name and mobile number of people who did viral test on ‘2021-10-4’ and tested to be positive in Central Lukewarm Hillside.**

**select name,mobile from**

**testdoctor natural join**

**(select name,r.mobile,doctorname from testpeople natural join testreport as r where**

**(r.collecttime >= '2021-10-4 00:00:00' and r.collecttime < '2021-10-5 00:00:00') and**

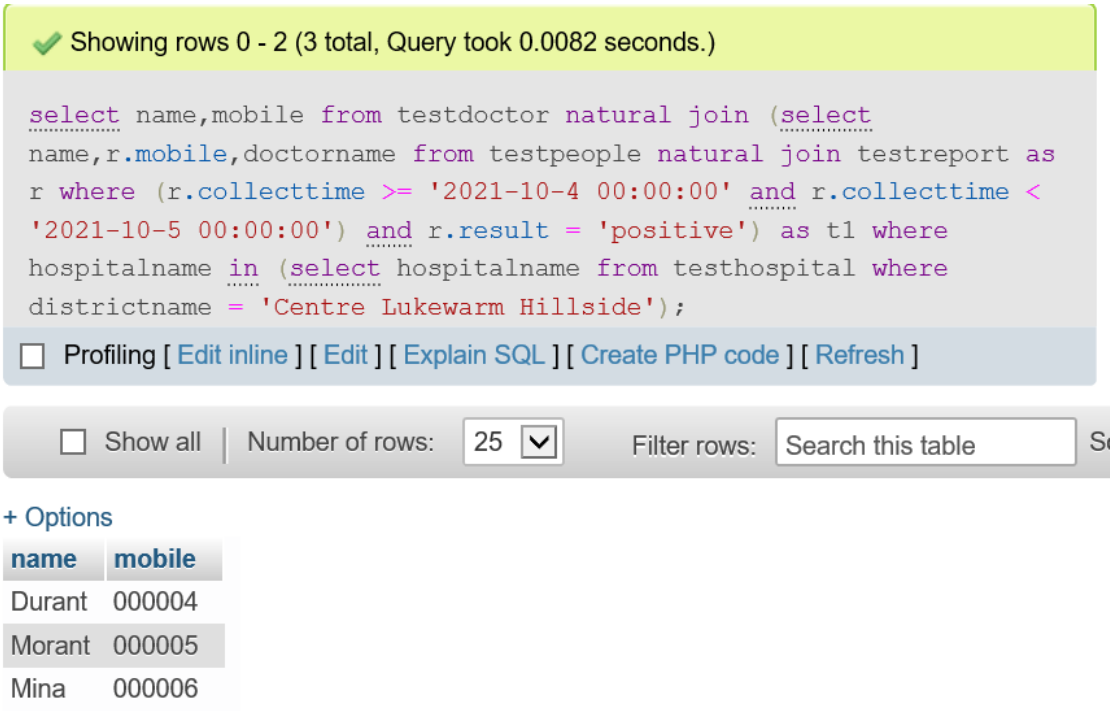
**r.result = 'positive') as t1**

**where**

**hospitalname in (select hospitalname from testhospital**

**where districtname = 'Centre Lukewarm Hillside')**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 2 (3 total, Query took 0.0082 seconds.)

```
select name,mobile from testdoctor natural join (select
name,r.mobile,doctorname from testpeople natural join testreport as
r where (r.collecttime >= '2021-10-4 00:00:00' and r.collecttime <
'2021-10-5 00:00:00') and r.result = 'positive') as t1 where
hospitalname in (select hospitalname from testhospital where
districtname = 'Centre Lukewarm Hillside');
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all
Number of rows: 25
Filter rows: Search this table
S
+ Options

| name | mobile |
|---|---|
| Durant | 000004 |
| Morant | 000005 |
| Mina | 000006 |

***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 (date5positive - date4positive) as increment from**

**(select * from (select count(mobile) as date5positive from testreport WHERE**

**(collecttime >= '2021-10-5 00:00:00' and collecttime < '2021-10-6 00:00:00') and result = 'positive'**

**and doctorname in (select doctorname from testhospital natural join testdoctor**

```
where districtname = 'Centre Lukewarm Hillside')) as t1 natural join
(select count(mobile) as date4positive from testreport WHERE
(collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00') and result = 'positive'
and doctorname in (select doctorname from testhospital natural join testdoctor
where districtname = 'Centre Lukewarm Hillside')) as t2) as t3
```

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

**The following information is added to the testdoctor table**

| doctorname | hospitalname |
|---|---|
| ‘Alen’ | ‘aa’ |
| ‘Amy’ | ‘aa’ |
| ‘Alice’ | ‘aa’ |
| ‘Bob’ | ‘bb’ |
| ‘Billy’ | ‘bb’ |
| ‘Betty’ | ‘bb’ |
| ‘Cathy’ | ‘cc’ |
| ‘Cindy’ | ‘cc’ |
| ‘Curry’ | ‘cc’ |

**The following viral test report information is added to the testreport table (some attributes are hidden as they are not related to this task).**

| mobile | collecttime | doctorname | result |
|---|---|---|---|
| ‘000001’ | ‘2021-10-04 08:00:00’ | ‘Alen’ | ‘negative’ |
| ‘000002’ | ‘2021-10-04 09:00:00’ | ‘Alen’ | ‘negative’ |
| ‘000003’ | ‘2021-10-04 19:43:00’ | ‘Amy’ | ‘negative’ |
| ‘000004’ | ‘2021-10-04 15:55:00’ | ‘Betty’ | ‘negative’ |
| ‘000005’ | ‘2021-10-04 11:52:00’ | ‘Bob’ | ‘negative’ |
| ‘000006’ | ‘2021-10-04 08:32:00’ | ‘Cindy’ | ‘negative’ |
| ‘000007’ | ‘2021-10-04 16:59:00’ | ‘Curry’ | ‘negative’ |
| ‘000008’ | ‘2021-10-04 21:12:00’ | ‘Curry’ | ‘positive’ |
| ‘000009’ | ‘2021-10-04 09:45:00’ | ‘Cindy’ | ‘negative’ |
| ‘000010’ | ‘2021-10-04 10:12:00’ | ‘Curry’ | ‘negative’ |
| ‘000011’ | ‘2021-10-05 13:51:00’ | ‘Alen’ | ‘negative’ |
| ‘000012’ | ‘2021-10-05 15:21:00’ | ‘Alen’ | ‘positive’ |
| ‘000013’ | ‘2021-10-05 10:00:00’ | ‘Bob’ | ‘negative’ |
| ‘000014’ | ‘2021-10-05 11:09:00’ | ‘Cindy’ | ‘positive’ |
| ‘000015’ | ‘2021-10-05 10:00:00’ | ‘Betty’ | ‘negative’ |
| ‘000016’ | ‘2021-10-05 11:00:00’ | ‘Curry’ | ‘positive’ |
| ‘000017’ | ‘2021-10-05 12:00:00’ | ‘Curry’ | ‘negative’ |
| ‘000018’ | ‘2021-10-05 11:00:00’ | ‘Curry’ | ‘negative’ |
| ‘000019’ | ‘2021-10-05 11:00:59’ | ‘Curry’ | ‘negative’ |
| ‘000020’ | ‘2021-10-05 11:01:47’ | ‘Curry’ | ‘negative’ |

**The corresponding mobile numbers and hospital names above have already been added to the testpeople and testhospital table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

The test data set contains people who did the viral test on ‘2021-10-04’ and ‘2021-10-05’ and among all these people there are positive cases and negative cases. The expected result of the query should only contain the value of positive cases on ‘2021-10-05’ minus the positive cases on ‘2021-10-04’. The people who did not do the viral test on ‘2021-10-04’ or ‘2021-10-05’ and the people who are not positive should be filtered out.

The detailed illustrations are as follows:

1.The task is to calculate the increase in new positive cases in the Central Lukewarm Hillside. The first step is to select the hospitals which can do the viral test in this district.

```
select hospitalname from testhospital

where districtname = 'Centre Lukewarm Hillside'
```

2. As all hospitals capable of doing the test have been selected, the next step is to select all doctors who can collect samples working in hospital in Centre Lukewarm Hillside.

```
select doctorname from testhospital natural join testdoctor

where districtname = 'Centre Lukewarm Hillside'
```

3. Count the number of all the positive patients on ‘2021-12-05’ in central Lukewarm Hillside.

```
select count(mobile) as date5positive from testreport WHERE

(collecttime >= '2021-10-5 00:00:00' and collecttime < '2021-10-6 00:00:00') and result = 'positive'

and doctorname in (select doctorname from testhospital natural join testdoctor

where districtname = 'Centre Lukewarm Hillside')
```

4.Count the number of all the positive patients on ‘2021-12-04’ in central Lukewarm Hillside.

```
select count(mobile) as date4positive from testreport WHERE

(collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00') and result = 'positive'

and doctorname in (select doctorname from testhospital natural join testdoctor

where districtname = 'Centre Lukewarm Hillside')
```

5.As the task is to calculate the increment of the positive cases, the idea is to use the number of positive cases on ‘2021-10-5’ minus the number of positive cases on ‘2021-10-4’. The number of the positive cases have been calculated above (step 3 and step 4). Use ‘natural join’ to assign the value of ‘2021-10-04’ and ‘2021-10-05’ to two different columns in order to make it easy to subtract to get the final answer.

```
select * from (select count(mobile) as date5positive from testreport WHERE

(collecttime >= '2021-10-5 00:00:00' and collecttime < '2021-10-6 00:00:00') and result = 'positive'

and doctorname in (select doctorname from testhospital natural join testdoctor

where districtname = 'Centre Lukewarm Hillside')) as t1 natural join
```

**(select count(mobile) as date4positive from testreport WHERE**

**(collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00') and result = 'positive'**

**and doctorname in (select doctorname from testhospital natural join testdoctor**

**where districtname = 'Centre Lukewarm Hillside')) as t2**

**6.As the two different columns 'date5positive' and 'date4positive' only have one value for each, just simply use 'date5positive' to minus 'date4positive', which can finally get the increment. The final query is as follows:**

**select (date5positive - date4positive) as increment from**

**(select * from (select count(mobile) as date5positive from testreport WHERE**

**(collecttime >= '2021-10-5 00:00:00' and collecttime < '2021-10-6 00:00:00') and result = 'positive'**

**and doctorname in (select doctorname from testhospital natural join testdoctor**

**where districtname = 'Centre Lukewarm Hillside')) as t1 natural join**

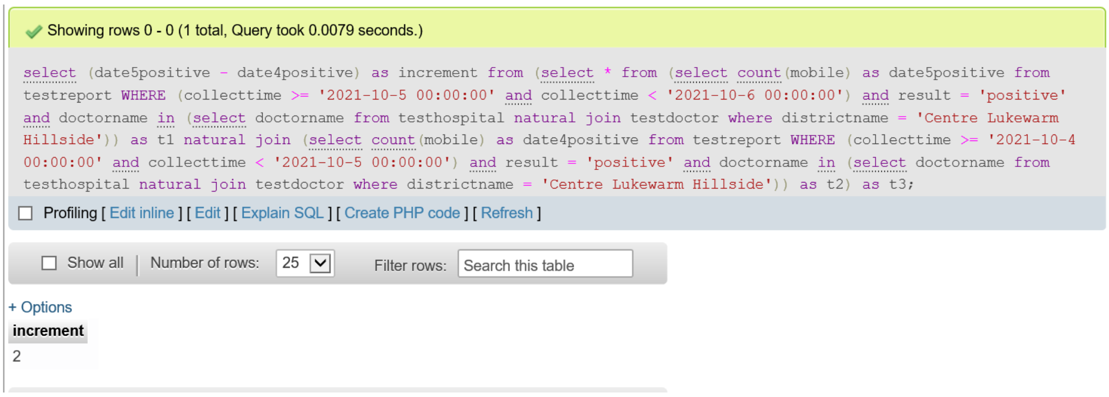
**(select count(mobile) as date4positive from testreport WHERE**

**(collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00') and result = 'positive'**

**and doctorname in (select doctorname from testhospital natural join testdoctor**

**where districtname = 'Centre Lukewarm Hillside')) as t2) as t3**

The result of the SELECT statement (screenshots):



✓ Showing rows 0 - 0 (1 total, Query took 0.0079 seconds.)

```
select (date5positive - date4positive) as increment from (select * from (select count(mobile) as date5positive from
testreport WHERE (collecttime >= '2021-10-5 00:00:00' and collecttime < '2021-10-6 00:00:00') and result = 'positive'
and doctorname in (select doctorname from testhospital natural join testdoctor where districtname = 'Centre Lukewarm
Hillside')) as t1 natural join (select count(mobile) as date4positive from testreport WHERE (collecttime >= '2021-10-4
00:00:00' and collecttime < '2021-10-5 00:00:00') and result = 'positive' and doctorname in (select doctorname from
testhospital natural join testdoctor where districtname = 'Centre Lukewarm Hillside')) as t2) as t3;
```

| increment |
|---|
| 2 |

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this table
+ Options

***Use case 8***: Assume that the spread rate of a virus is calculated by dividing the total number of people that were in the same district as the positive case with 48 hours (calculated in ***use case 1***) by the total number of people among them that later confirmed to be infected in 14 days. Again, assume that a person called Mark 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 totalnumber / infectednumber as spreadrate from**

**(select count(t1.simid) as totalnumber from**

**(select t.simid from**

```
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')) as t1) as tn1,
(
select COUNT(mobile) as infectednumber from
(select * from testreport inner join
(select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')) as t1 ON
testreport.mobile = simid) as t2 WHERE
result = 'positive' and collecttime >= '2021-10-09 19:30:00' and collecttime <= '2021-10-23 19:30:00') as tn2
```

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

**The following travel information of people living in this country is added to the bsrecord table.**

| gpslocation | simid | connectiontime | disconnectiontime |
|---|---|---|---|
| ‘1,1’ | ‘123450’ | ‘2021-01-01 00:00:00’ | Null |
| ‘2,2’ | ‘123451’ | ‘2021-10-01 08:00:00’ | Null |
| ‘2,2’ | ‘123452’ | ‘2021-10-01 08:00:00’ | Null |
| ‘1,1’ | ‘123453’ | ‘2021-10-01 00:00:00’ | Null |
| ‘2,2’ | ‘123454’ | ‘2021-10-08 11:00:00’ | Null |
| ‘3,3’ | ‘123455’ | ‘2021-10-06 15:00:00’ | Null |
| ‘3,3’ | ‘233636’ | ‘2021-01-01 00:00:00’ | Null |

**The following viral test information of people is added to the testreport table (some attributes are hidden as they are not related to this task).**

| Mobile | collecttime | result |
|---|---|---|
| ‘123450’ | ‘2021-10-11 08:00:00’ | ‘negative’ |
| ‘123451’ | ‘2021-10-14 11:00:00’ | ‘positive’ |
| ‘123452’ | ‘2021-10-16 12:00:00’ | ‘positive’ |
| ‘123453’ | ‘2021-10-18 15:00:00’ | ‘negative’ |
| ‘123454’ | ‘2021-10-19 11:00:00’ | ‘negative’ |
| ‘123455’ | ‘2021-10-13 16:00:00’ | ‘negative’ |

**The corresponding mobile numbers and GPS locations above have been already added to the testpeople and bslocation table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**The test data set contains the travelling records of people in Lukewarm Kingdom, which means the set contains people who were in the same district as Mark and people among them who later confirmed to be positive and also people among them who were still healthy (the background is the use case 1). The expected result of the query should only contain a value. To calculate the value, we need to know the number of people who were in the same district as the positive case with 48 hours (has already been calculated by use case 1) and the number of people among them that later confirmed to be positive in 14 days. So, the people who were tested to be negative in 14 days should be filtered out.**

**The detailed illustrations are as follows:**

**1.To calculate the target value, two values should be calculated. One is the total number of people who were in the same district as Mark during the specific period. The other one is the number of people among them who were tested to be positive in 14 days. The first step is to select the people who stayed in the same district with Mark (including mark himself) during last 48 hours (the same method as use case 1).**

```
select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')
```

**2. count the total number in step 1 (in this way, the first necessary value has been calculated)**

```
select count(t1.simid) as totalnumber from
(select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r
WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')) as t1
```

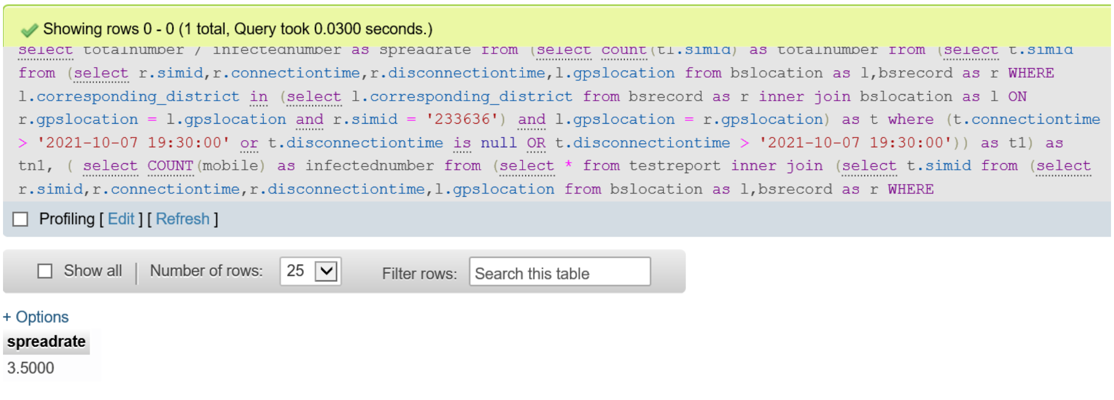
**3.The next step is to select the number of people among them who were later in 14 days to be positive. Then count the number of these people. To achieve this, we need to find all people who did the viral test within 14 days and the result is positive, then select the people among them and also in use case 1 condition. In this way, finally selects all people among the target group who later confirmed to be positive. The following subquery realized this function.**

```
select count(mobile) as infectednumber from
(select * from testreport inner join
(select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r
WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')) as t1 ON
testreport.mobile = simid) as t2 WHERE
result = 'positive' and collecttime >= '2021-10-09 19:30:00' and collecttime <= '2021-10-23 19:30:00'
```

**4. as in step 2 and step 3, the selected two table only contain one column for each and for each column only has one corresponding values. The two values are the two necessary values which can used to calculate the viral spread rate. Use 'cross join' to connect the two table, then select the final spread rate by simply divide the two columns. The final select is shown below:**

```
select totalnumber / infectednumber as spreadrate from
(select count(t1.simid) as totalnumber from
(select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')) as t1) as tn1,
(select COUNT(mobile) as infectednumber from
(select * from testreport inner join
(select t.simid from
(select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in
(select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t
where
(t.connectiontime > '2021-10-07 19:30:00' or
t.disconnectiontime is null OR
t.disconnectiontime > '2021-10-07 19:30:00')) as t1 ON
testreport.mobile = simid) as t2 WHERE
result = 'positive' and collecttime >= '2021-10-09 19:30:00' and collecttime <= '2021-10-23 19:30:00') as tn2
```

The result of the SELECT statement (screenshots):



✓ Showing rows 0 - 0 (1 total, Query took 0.0300 seconds.)

```
select totalnumber / infectednumber as spreadrate from (select count(t1.simid) as totalnumber from (select t.simid
from (select r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
l.corresponding_district in (select l.corresponding_district from bsrecord as r inner join bslocation as l ON
r.gpslocation = l.gpslocation and r.simid = '233636') and l.gpslocation = r.gpslocation) as t where (t.connectiontime
> '2021-10-07 19:30:00' or t.disconnectiontime is null OR t.disconnectiontime > '2021-10-07 19:30:00')) as t1) as
tn1, ( select COUNT(mobile) as infectednumber from (select * from testreport inner join (select t.simid from (select
r.simid,r.connectiontime,r.disconnectiontime,l.gpslocation from bslocation as l,bsrecord as r WHERE
```

□ Profiling [ Edit ] [ Refresh ]
□ Show all | Number of rows: 25
Filter rows: Search this table
+ Options

| spreadrate |
|---|
| 3.5000 |

## 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***:

Mark is a student studying in Lukewarm university located in district a. Mark's phone number is 233636. The university requires that the viral test should be done 48 hours before the new semester starts (the new semester will start on '2021-10-06', which means a student should do the viral test after '2021-10-4 00:00:00') and only if the result is negative, this student can enter the campus. Today is '2021-10-5', Mark wants to make sure whether he can enter the campus in the beginning of the new semester. Please write a query to find whether Mark has a qualification to enter the campus.

Your SQL statement:

**select tp.name,tr.collecttime,tr.result from testreport as tr natural join testpeople as tp**

**where**

**mobile = '233636' and collecttime >= '2021-10-04 00:00:00'**

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

**The following personal information of people who did the viral tests is added to the testpeople table (some attributes are hidden as they are not related to this task)**

| mobile | name |
|---|---|
| '000001' | 'James' |
| '000002' | 'Jason' |
| '000003' | 'Jack' |
| '000004' | 'Durant' |
| '233636' | 'Mark' |

**The following viral test information of people is added to the testreport table (some attributes are hidden as they are not related to this task).**

| mobile | collecttime | result |
|---|---|---|
| '000001' | '2021-10-4 08:00:00' | 'negative' |
| '000002' | '2021-10-4 09:00:00' | 'negative' |
| '000003' | '2021-10-4 19:43:00' | 'negative' |
| '000004' | '2021-10-4 15:55:00' | 'negative' |
| '233636' | '2021-10-4 12:00:00' | 'negative' |
| '233636' | '2021-10-4 10:00:00' | 'negative' |

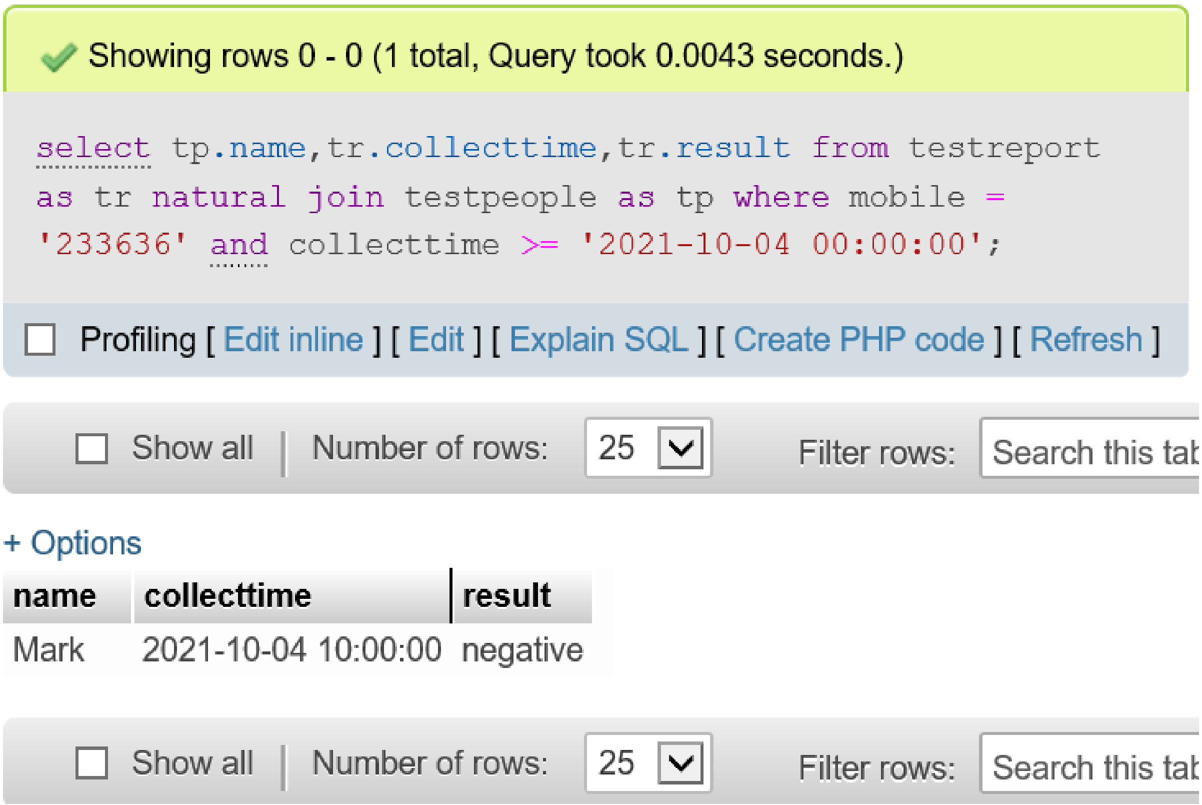
**Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**The test data set contains people including all Mark's test records and other's records. The expected result of the query should only contain Mark's test report which did after '2021-10-04' and the result is negative. Other people's test report and all invalid ('invalid' here means Mark failed to reach the requirement to make him enter the campus in new semester) reports of Mark should be filtered out.**

**The detailed illustrations are as follows:**

**The select query above make sure that select all the test report which belongs to Mark (mobile = ‘233636’). Then select the report which has done after ‘2021-10-04’ (collecttime >= ‘2021-10-04 00:00:00’), which can help Mark check whether he can enter the campus or not.**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 0 (1 total, Query took 0.0043 seconds.)

```
select tp.name,tr.collecttime,tr.result from testreport
as tr natural join testpeople as tp where mobile =
'233636' and collecttime >= '2021-10-04 00:00:00';
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this tab
+ Options

| name | collecttime | result |
|---|---|---|
| Mark | 2021-10-04 10:00:00 | negative |

□ Show all | Number of rows: 25 Filter rows: Search this tab

***Use case 2***:

Mark is a university student studying in Lukewarm Kingdom. During the summer vacation, Mark wants to travel in Lukewarm Kingdom. it is dangerous if Mark just randomly chooses an area for travelling as the epidemic has broken out so he decided only go to low-risk district for a trip. Please list all districts which are now in low risk level.

Your SQL statement:

**select districtname from risklevel WHERE**

**risk_level = 'low';**

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

**The following district and risk level information is added to the risklevel table.**

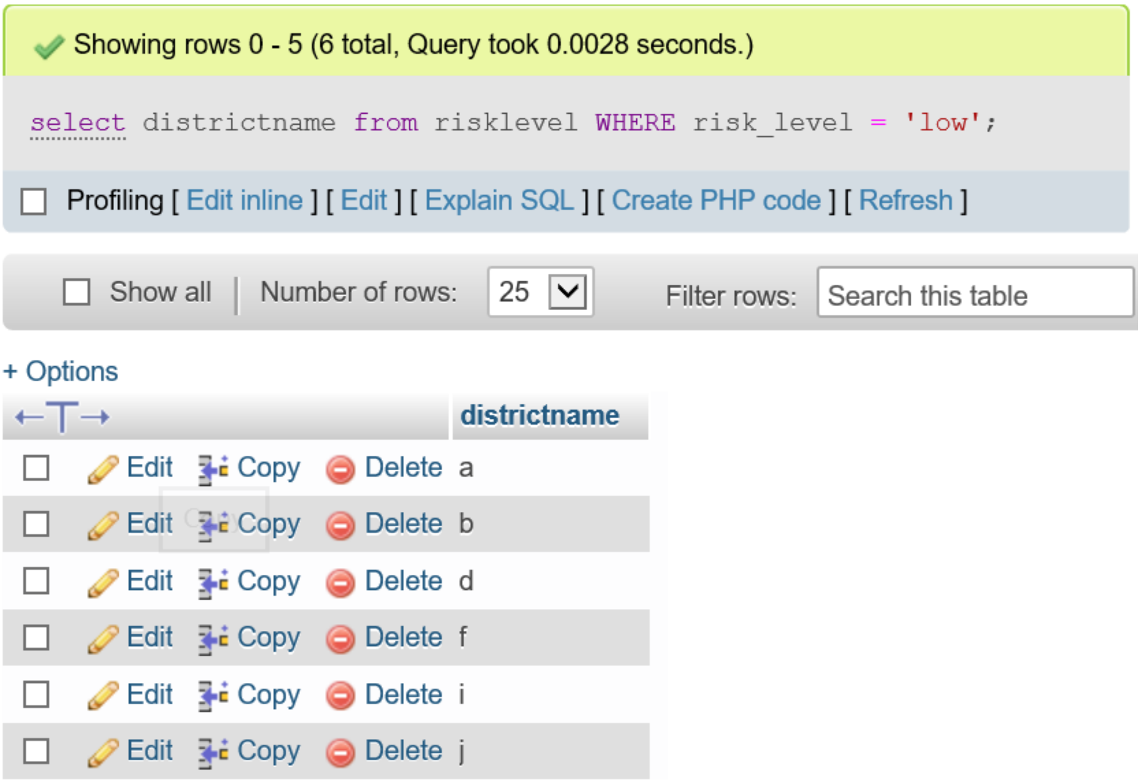
| districtname | risk_level | regionname |
|---|---|---|
| ‘a’ | ‘low’ | ‘central’ |
| ‘b’ | ‘low’ | ‘central’ |

| | | |
|---|---|---|
| **‘c’** | **‘mid’** | **‘north’** |
| **‘d’** | **‘low’** | **‘north’** |
| **‘e’** | **‘mid’** | **‘south’** |
| **‘f’** | **‘low’** | **‘south’** |
| **‘g’** | **‘high’** | **‘east’** |
| **‘h’** | **‘high’** | **‘east’** |
| **‘i’** | **‘low’** | **‘west’** |
| **‘j’** | **‘low’** | **‘west’** |

**The corresponding region names above have already been added to the regionname table.**

**This test data set contains districts that are in low risk level and the districts that are not in low risk level. The expected result of the query should only contain districts that are in low risk level. The districts that are not in low risk level should be filtered out.**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 5 (6 total, Query took 0.0028 seconds.)

```
select districtname from risklevel WHERE risk_level = 'low';
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all
Number of rows: 25
Filter rows: Search this table
+ Options
←T→

| | districtname |
|---|---|
| □ Edit Copy Delete | a |
| □ Edit Copy Delete | b |
| □ Edit Copy Delete | d |
| □ Edit Copy Delete | f |
| □ Edit Copy Delete | i |
| □ Edit Copy Delete | j |

***Use case 3***:

Mark lives in Lukewarm Kingdom. Mark wants to go shopping on ‘2021-10-15’ in a shopping mall. But it requires that Mark cannot go to districts which are high or mid risk level to enter the shopping mall. Please write a select query to list the mid-risk or high-risk level districts that Mark have been to during last 14 days (assume during last 14 days, the risk level of all districts did not change).

Your SQL statement:

**select districtname from**

**(select gpslocation from bsrecord where simid = '233636' and connectiontime >= '2021-10-01 00:00:00') as t1**

**natural join**

**(select gpslocation, r.districtname from risklevel r inner JOIN bslocation l on r.districtname = l.corresponding_district where**

**risk_level = 'high' or risk_level = 'mid') as t2**

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

**The following travelling history information is added to the bsrecord table (some attributes are hidden as they are not related to this task).**

| gpslocation | simid | connectiontime |
|---|---|---|
| ‘5,5’ | ‘123450’ | ‘2021-10-02 00:00:00’ |
| ‘7,7’ | ‘123451’ | ‘2021-10-03 12:00:00’ |
| ‘1,1’ | ‘233636’ | ‘2021-09-26 08:00:00’ |
| ’15,15’ | ‘233636’ | ‘2021-10-02 09:00:00’ |
| ‘3,3’ | ‘233636’ | ‘2021-10-05 11:00:00’ |
| ‘1,1’ | ‘233636’ | ‘2021-10-07 09:00:00’ |

**The following districts and corresponding risk level information is added to this risklevel table (some attributes are hidden as they are not related to this task).**

| districtname | risk_level |
|---|---|
| ‘a’ | ‘low’ |
| ‘b’ | ‘low’ |
| ‘c’ | ‘mid’ |
| ‘d’ | ‘low’ |
| ‘e’ | ‘mid’ |
| ‘f’ | ‘low’ |
| ‘g’ | ‘high’ |
| ‘h’ | ‘high’ |
| ‘i’ | ‘low’ |

**The corresponding GPS location information has already been added to the bslocation table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This data set contains the travelling information of people in Lukewarm Kingdom, which means this data set contains all the low-risk level districts that Mark has been to during last 14 days and all the non-low risk level districts that Mark has been during last 14 days. The expected result of the query should only contain the mid-risk or high-risk level districts that Mark has been to during last 14 days. The low-risk level districts that Mark has been to during last 14 days should be filtered out.**

**The detailed illustrations are as follows:**

**1. first to select all the base station location that mark has ever been to during last 14 days.**

**select gpslocation from bsrecord where**

**simid = '233636' and connectiontime >= '2021-10-01 00:00:00'**

**2. select all mid or high risk-level districts and the corresponding gpslocation**

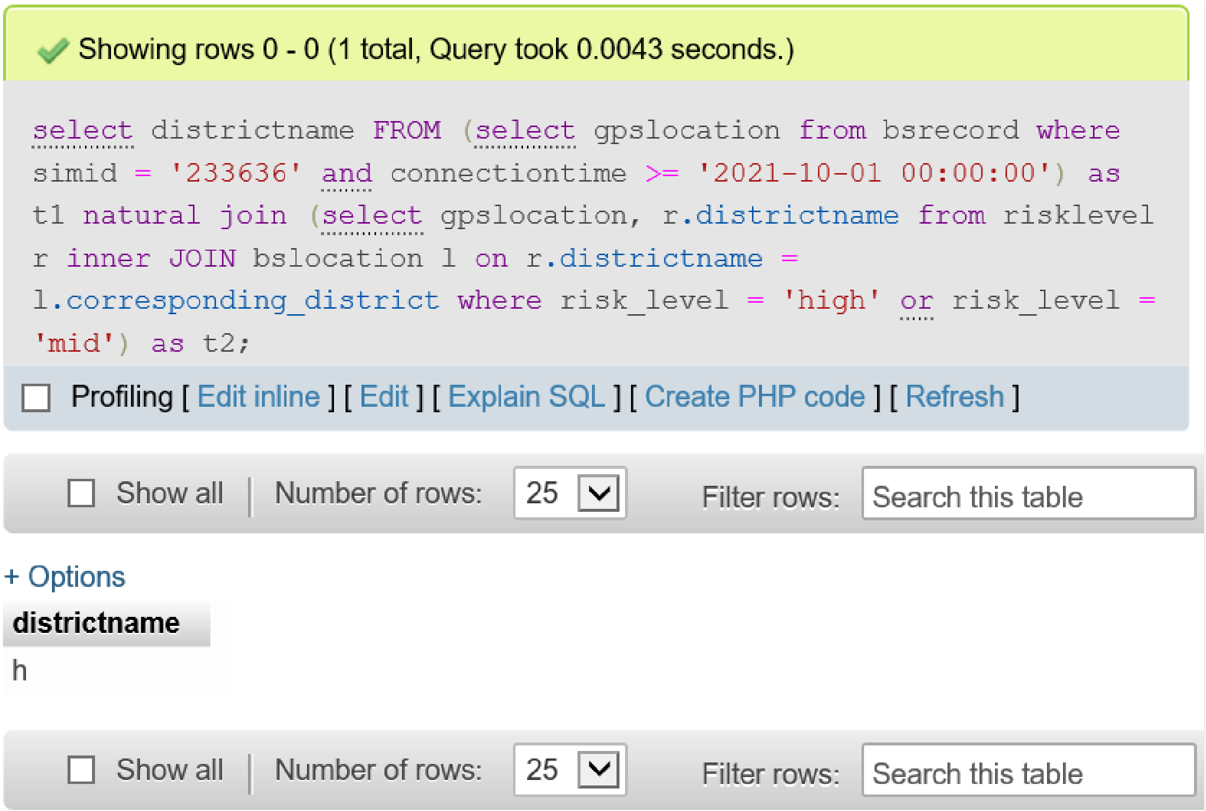
**select gpslocation, r.districtname from risklevel r inner JOIN**

```
bslocation l on
r.districtname = l.corresponding_district
where
risk_level = 'high' or risk_level = 'mid'
```

**3. As all districts that are not in low-risk level and the base station area that Mark has been to is selected. The final step is to use ‘natural join’ to connect the two tables, finally the target districts can be selected.**

```
select districtname from
(select gpslocation from bsrecord where simid = '233636' and connectiontime >= '2021-10-01 00:00:00') as t1
natural join
(select gpslocation, r.districtname from risklevel r inner JOIN bslocation l on r.districtname = l.corresponding_district where
risk_level = 'high' or risk_level = 'mid') as t2
```

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 0 (1 total, Query took 0.0043 seconds.)

```
select districtname FROM (select gpslocation from bsrecord where
simid = '233636' and connectiontime >= '2021-10-01 00:00:00') as
t1 natural join (select gpslocation, r.districtname from risklevel
r inner JOIN bslocation l on r.districtname =
l.corresponding_district where risk_level = 'high' or risk_level =
'mid') as t2;
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 | Filter rows: Search this table
+ Options
□ Show all | Number of rows: 25 | Filter rows: Search this table

| districtname |
|---|
| h |

*Use case 4*:

Mark is a biological scientist in Lukewarm Kingdom. Mark's team has developed the valid vaccine to this virus and on '2021-10-01' the vaccine can finally be applied to human beings to prevent from being infected by this virus. According some important experimental data, it may be dangerous for children under 12 or elders older than 70 to be vaccinated. In district 'a', almost everyone has accepted to do the viral test after '2021-10-01' to find whether they can be vaccinated (the one who are tested to be positive cannot be vaccinated). Please write a select query to list the name and mobile number of all people in district 'a' who are qualified to be vaccinated.

Your SQL statement:

**select name,mobile from**

**testdoctor natural join**

**(select name,r.mobile,doctorname from testpeople natural join testreport as r where**

**r.collecttime >= '2021-10-1 00:00:00' and age >= 12 and age <= 70 and**

**r.result = 'negative') as t1**

**where**

**hospitalname in (select hospitalname from testhospital**

**where districtname = 'a')**

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

**The following personal information of people who did the viral tests is added to this testpeople table (some attributes are hidden as they are not related to this task).**

| mobile | age |
|---|---|
| **'000001'** | **9** |
| **'000002'** | **23** |
| **'000003'** | **25** |
| **'000004'** | **32** |
| **'000005'** | **21** |
| **'000006'** | **24** |
| **'000007'** | **29** |
| **'000008'** | **11** |
| **'000009'** | **74** |
| **'000010'** | **11** |

**The following viral test report information is added to the testreport table (some attributes are hidden as they are not related to this task).**

| mobile | collecttime | doctorname | result |
|---|---|---|---|
| **'000001'** | **'2021-10-04 08:00:00'** | **'Alen'** | **'negative'** |
| **'000002'** | **'2021-10-04 09:00:00'** | **'Alen'** | **'negative'** |
| **'000003'** | **'2021-10-04 19:43:00'** | **'Amy'** | **'negative'** |
| **'000004'** | **'2021-10-04 15:55:00'** | **'Betty'** | **'negative'** |
| **'000005'** | **'2021-10-04 11:52:00'** | **'Bob'** | **'negative'** |
| **'000006'** | **'2021-10-04 08:32:00'** | **'Cindy'** | **'negative'** |

| | | | |
|---|---|---|---|
| ‘000007’ | ‘2021-10-04 16:59:00’ | ‘Curry’ | ‘negative’ |
| ‘000008’ | ‘2021-10-04 21:12:00’ | ‘Curry’ | ‘negative’ |
| ‘000009’ | ‘2021-10-04 09:45:00’ | ‘Cindy’ | ‘negative’ |
| ‘000010’ | ‘2021-10-04 10:12:00’ | ‘Curry’ | ‘negative’ |

**The corresponding mobile numbers, doctor names above have been already added to the testpeople, testdoctor table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This data set contains the viral test report data of people in Lukewarm Kingdom, which means this test data set contains people in district ‘a’ who are qualified to be vaccinated and people who are not qualified to be vaccinated. The expected result of the query should only contain people who are qualified to be vaccinated. The people who are not qualified should be properly filtered out.**

**The detailed illustrations are as follows:**

**1.the task requires to find qualified people in district ‘a’. the first step is to find all the hospitals capable of doing the viral test in district ‘a’.**

```
select hospitalname from testhospital

where districtname = ‘a’
```

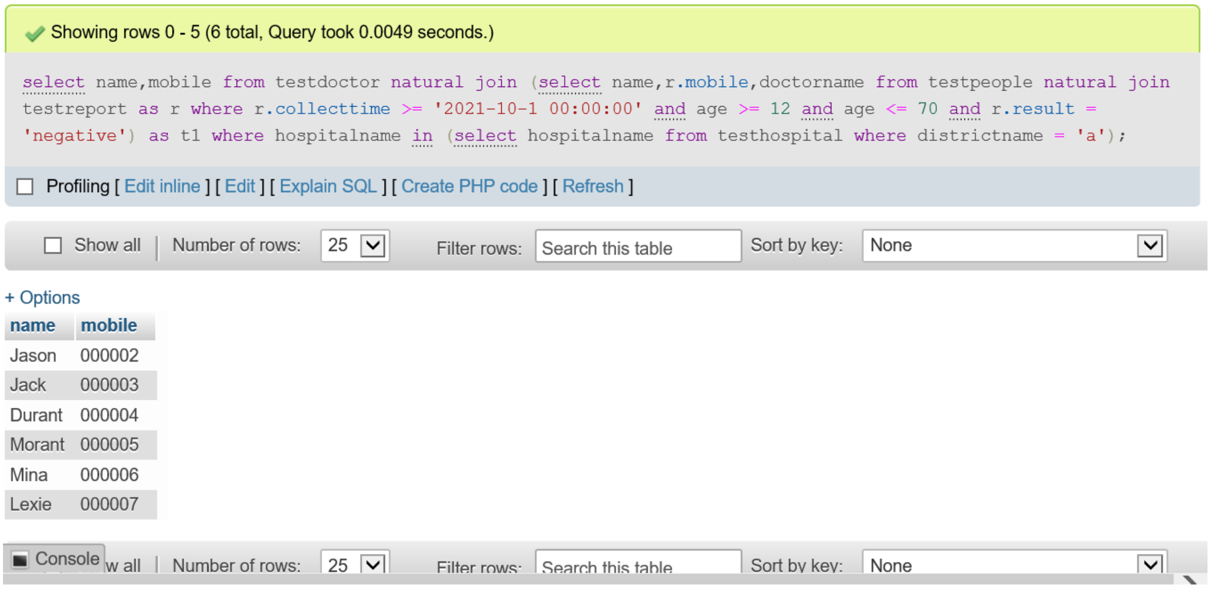
**2.select all people whose age is between 12 to 70 and tested to be negative after ‘2021-10-1’**

```
select name,r.mobile,doctorname from testpeople natural join testreport as r where

r.collecttime >= '2021-10-1 00:00:00'  and age >= 12 and age <= 70 and

r.result = 'negative'
```

**3. the final step is to combine step 2 and step 1 so that all people who are qualified in district ‘a’ can be selected.**

```
select name,mobile from

testdoctor natural join

(select name,r.mobile,doctorname from testpeople natural join testreport as r where

r.collecttime >= '2021-10-1 00:00:00'  and age >= 12 and age <= 70 and

r.result = 'negative') as t1

where

hospitalname in (select hospitalname from testhospital

where districtname = 'a')
```

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 5 (6 total, Query took 0.0049 seconds.)

```
select name,mobile from testdoctor natural join (select name,r.mobile,doctorname from testpeople natural join
testreport as r where r.collecttime >= '2021-10-1 00:00:00' and age >= 12 and age <= 70 and r.result =
'negative') as t1 where hospitalname in (select hospitalname from testhospital where districtname = 'a');
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25
Filter rows: Search this table
Sort by key: None
+ Options

| name | mobile |
|---|---|
| Jason | 000002 |
| Jack | 000003 |
| Durant | 000004 |
| Morant | 000005 |
| Mina | 000006 |
| Lexie | 000007 |

***Use case 5***:

The virus has broken out since last year (2020), for a whole year Lukewarm Kingdom tried its best to fight the epidemic. And this country wants to know the change number of all positive people from last year (2020) to the current year (2021), which can help the country to identify whether the viral preventative measures are effective or not. Using number of all positive cases in 2021 minus the number of all positive cases in 2021 as the 'changenumber'. If the 'changenumber' is negative, which means positive cases this year has decreased. Please write a query to list the value of the 'changenumber' (assume everyone in this country has done the viral test. If a person is cured after he was infected, he is infected again. In this case, it'll be counted as 2 positive cases).

Your SQL statement:

**select (pc2021 - pc2020) as changenumber from**

**(select count(mobile) as pc2021 from**

**(select distinct mobile from testreport where**

**result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t2) as t3,**

**(select count(mobile) as pc2020 from**

**(select distinct mobile from testreport where**

**result = 'positive' and collecttime >= '2020-01-01 00:00:00' and collecttime < '2021-01-01 00:00:00') as t1) as t4**

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

**The following viral test report information is added to the testreport table (some attributes are hidden as they are not related to this task).**

| mobile | collecttime | result |
|---|---|---|
| **'000001'** | **'2020-10-04 08:00:00'** | **'negative'** |

| | | |
|---|---|---|
| ‘000002’ | ‘2020-10-04 09:00:00’ | ‘negative’ |
| ‘000003’ | ‘2020-10-04 19:43:00’ | ‘negative’ |
| ‘000004’ | ‘2020-10-04 15:55:00’ | ‘negative’ |
| ‘000005’ | ‘2020-10-04 11:52:00’ | ‘negative’ |
| ‘000006’ | ‘2020-10-04 08:32:00’ | ‘negative’ |
| ‘000007’ | ‘2020-10-04 16:59:00’ | ‘negative’ |
| ‘000008’ | ‘2020-10-04 21:12:00’ | ‘positive’ |
| ‘000009’ | ‘2020-10-04 09:45:00’ | ‘negative’ |
| ‘000010’ | ‘2020-10-04 10:12:00’ | ‘negative’ |
| ‘000011’ | ‘2020-10-05 13:51:00’ | ‘negative’ |
| ‘000012’ | ‘2020-10-05 15:21:00’ | ‘positive’ |
| ‘000013’ | ‘2020-10-05 10:00:00’ | ‘negative’ |
| ‘000014’ | ‘2020-10-05 11:09:00’ | ‘positive’ |
| ‘000015’ | ‘2020-10-05 10:00:00’ | ‘negative’ |
| ‘000016’ | ‘2020-10-05 11:00:00’ | ‘positive’ |
| ‘000017’ | ‘2020-10-05 12:00:00’ | ‘negative’ |
| ‘000018’ | ‘2020-10-05 11:00:00’ | ‘negative’ |
| ‘000019’ | ‘2020-10-05 11:00:59’ | ‘negative’ |
| ‘000020’ | ‘2020-10-05 11:01:47’ | ‘negative’ |
| ‘000001’ | ‘2021-10-04 08:00:00’ | ‘negative’ |
| ‘000002’ | ‘2021-10-04 09:00:00’ | ‘negative’ |
| ‘000003’ | ‘2021-10-04 19:43:00’ | ‘negative’ |
| ‘000004’ | ‘2021-10-04 15:55:00’ | ‘negative’ |
| ‘000005’ | ‘2021-10-04 11:52:00’ | ‘negative’ |
| ‘000006’ | ‘2021-10-04 08:32:00’ | ‘negative’ |
| ‘000007’ | ‘2021-10-04 16:59:00’ | ‘negative’ |
| ‘000008’ | ‘2021-10-04 21:12:00’ | ‘negative’ |
| ‘000009’ | ‘2021-10-04 09:45:00’ | ‘negative’ |
| ‘000010’ | ‘2021-10-04 10:12:00’ | ‘negative’ |
| ‘000011’ | ‘2021-10-05 13:51:00’ | ‘negative’ |
| ‘000012’ | ‘2021-10-05 15:21:00’ | ‘negative’ |
| ‘000013’ | ‘2021-10-05 10:00:00’ | ‘negative’ |
| ‘000014’ | ‘2021-10-05 11:09:00’ | ‘positive’ |
| ‘000015’ | ‘2021-10-05 10:00:00’ | ‘negative’ |
| ‘000016’ | ‘2021-10-05 11:00:00’ | ‘negative’ |
| ‘000017’ | ‘2021-10-05 12:00:00’ | ‘negative’ |
| ‘000018’ | ‘2021-10-05 11:00:00’ | ‘negative’ |
| ‘000019’ | ‘2021-10-05 11:00:59’ | ‘negative’ |
| ‘000020’ | ‘2021-10-05 11:01:47’ | ‘negative’ |

**The corresponding mobile numbers above have been already added to the testpeople table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This test data set contains the viral test report data of people in Lukewarm Kingdom, which means this test data set contains all people tested to be positive and negative last year and all the people tested to be positive and negative this current year. As the task requires the value of the subtraction of two years, the expected result of the query should only contain the subtraction value of people who are tested to be positive in two years. People who are negative should be properly filtered out.**

**The detailed illustrations are as follows:**

**1.The task is to calculate the difference value of two years. The core is to select the number of all positive cases in 2020 and the number of all positive cases in 2021. The first step is to count the number of positive people in 2020.**

```
select count(mobile) as pc2020 from
(select mobile from testreport where
result = 'positive' and collecttime >= '2020-01-01 00:00:00' and collecttime < '2021-01-01 00:00:00') as t1
```

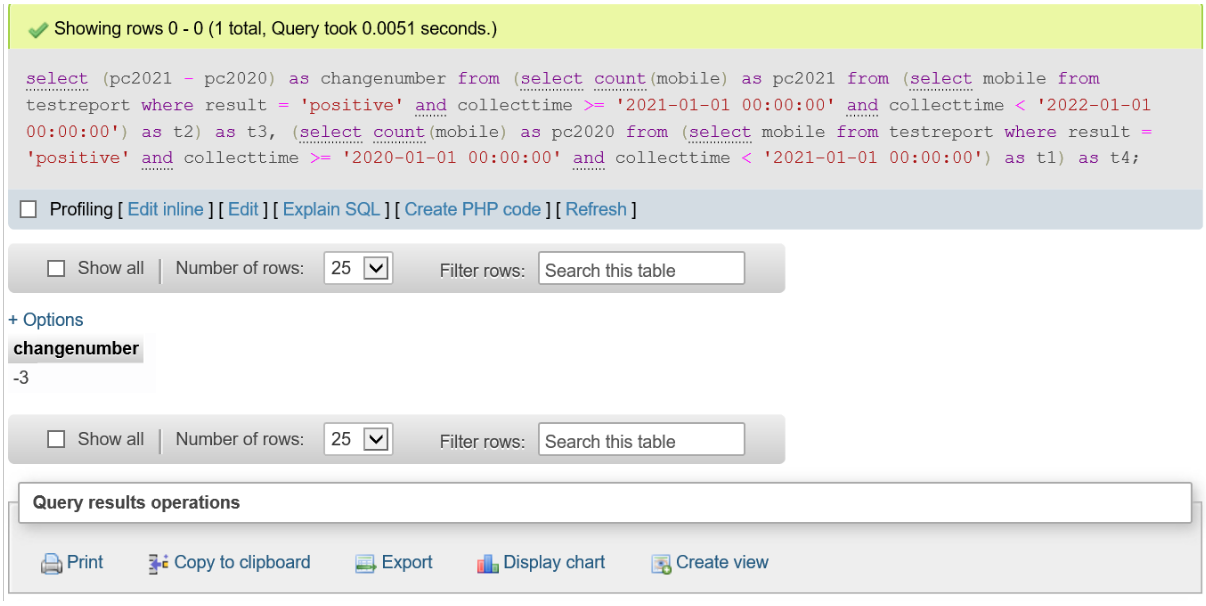
**2.the next step is to calculate all the positive cases in 2021 (the same method with step 1).**

```
select count(mobile) as pc2021 from
(select mobile from testreport where
result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t2
```

**3. in step 1 and step 2, the column pc2020 and column pc2021 only have one value for each. Pc2021 represents all positive cases in 2021 and the same for pc2020. The final step is to simply connect two tables in step 1 and step 2 and then calculated the difference value of pc2021 and pc2020 to get the final answer.**

```
select (pc2021 - pc2020) as changenumber from
(select count(mobile) as pc2021 from
(select mobile from testreport where
result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t2) as t3,
(select count(mobile) as pc2020 from
(select mobile from testreport where
result = 'positive' and collecttime >= '2020-01-01 00:00:00' and collecttime < '2021-01-01 00:00:00') as t1) as t4
```

The result of the SELECT statement (screenshot):



Showing rows 0 - 0 (1 total, Query took 0.0051 seconds.)

```
select (pc2021 - pc2020) as changenumber from (select count(mobile) as pc2021 from (select mobile from
testreport where result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01
00:00:00') as t2) as t3, (select count(mobile) as pc2020 from (select mobile from testreport where result =
'positive' and collecttime >= '2020-01-01 00:00:00' and collecttime < '2021-01-01 00:00:00') as t1) as t4;
```

| changenumber |
|---|
| -3 |

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this table
+ Options
□ Show all | Number of rows: 25 Filter rows: Search this table
Query results operations
Print
Copy to clipboard
Export
Display chart
Create view

***Use case 6***:

Mark is the chief biological scientists in Lukewarm Kingdom. One day Mark has proposed a way to fight the epidemic, which is called ‘group immunity’. He delivered a speech that almost 60 percent of population in Lukewarm Kingdom need to be infected by this virus, then the whole country will gain ‘group immunity’. So many citizens have criticized him after he made the speech, because they think it will cause unpredictable number of deaths and so many children and elders may die. Finally, Mark’s proposal did not be taken by Lukewarm Kingdom. Now, we define the infect rate by dividing the number of all people infected before by the total number of people in a specific area. Please write a query to calculate the infect rate in Lukewarm Kingdom in 2021 (assume all people did the viral test in 2021 and assume the database will delete the people who have already passed away).

Your SQL statement:

**select (infect2021 /total2021) as infectrate from**

**(select count(mobile) as infect2021 from**

**(select distinct mobile from testreport WHERE**

**result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t2) as t3,**

**(select count(mobile) as total2021 from**

**(select distinct mobile from testreport where**

**collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t1) as t4**

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

**The following viral test report information is added to the testreport table (some attributes are hidden as they are not related to this task).**

| mobile | collecttime | result |
|---|---|---|
| ‘000001’ | ‘2021-10-04 08:00:00’ | ‘negative’ |
| ‘000002’ | ‘2021-10-04 09:00:00’ | ‘negative’ |
| ‘000003’ | ‘2021-10-04 19:43:00’ | ‘negative’ |
| ‘000004’ | ‘2021-10-04 15:55:00’ | ‘negative’ |
| ‘000005’ | ‘2021-10-04 11:52:00’ | ‘negative’ |
| ‘000006’ | ‘2021-10-04 08:32:00’ | ‘negative’ |
| ‘000007’ | ‘2021-10-04 16:59:00’ | ‘negative’ |
| ‘000008’ | ‘2021-10-04 21:12:00’ | ‘negative’ |
| ‘000009’ | ‘2021-10-04 09:45:00’ | ‘negative’ |
| ‘000010’ | ‘2021-10-04 10:12:00’ | ‘negative’ |
| ‘000011’ | ‘2021-10-05 13:51:00’ | ‘negative’ |
| ‘000012’ | ‘2021-10-05 15:21:00’ | ‘negative’ |
| ‘000013’ | ‘2021-10-05 10:00:00’ | ‘negative’ |
| ‘000014’ | ‘2021-10-05 11:09:00’ | ‘positive’ |
| ‘000015’ | ‘2021-10-05 10:00:00’ | ‘negative’ |
| ‘000016’ | ‘2021-10-05 11:00:00’ | ‘negative’ |
| ‘000017’ | ‘2021-10-05 12:00:00’ | ‘negative’ |
| ‘000018’ | ‘2021-10-05 11:00:00’ | ‘negative’ |
| ‘000019’ | ‘2021-10-05 11:00:59’ | ‘negative’ |
| ‘000020’ | ‘2021-10-05 11:01:47’ | ‘negative’ |

**The corresponding mobile numbers above have already been added to the testpeople table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**The task is to calculate the infect rate. it is core to calculate two values, one is the total number of people in Lukewarm Kingdom in 2021 and the other one is the total number of people who were infected in 2021. This data set contain all viral test report data of people who did test in Lukewarm Kingdom in 2021. The expected result of query should only contain the total number of people who was infected in 2021 and the total number of people in Lukewarm Kingdom in 2021 in order to calculate the infect rate. other information should be properly filtered out.**

**The detailed illustrations are as follows:**

**1.The first step is to calculate the total number of people in Lukewarm in 2021. According to the extend use case 6, the people who were passed away have already been deleted and all people did the viral test, which means the number of people recorded in table testpeople equals the total number of people in the country. The usage of keyword ‘distinct’ is crucial as some people probably were tested multiple times, ‘distinct’ makes sure to avoid counting repeatedly.**

```
select count(mobile) as total2021 from

(select distinct mobile from testreport where

collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t1
```

**2.The next step is to calculate the number of people who were infected before. The usage of keyword ‘distinct’ is crucial as the same person may be infected multiple times, which means he may be tested to be positive multiple times. ‘distinct’ makes sure to avoid counting repeatedly.**

```
select count(mobile) as infect2021 from

(select distinct mobile from testreport WHERE
```

**result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t2**

**3.the final step is to combine step 2 and step 3. In step 2 and step 3, the table only contain one column for each and the column only contain one value. So, the final step is to simply divide infect2021 by total2021 to get the final answer.**

**select (infect2021 /total2021) as infectrate from**

**(select count(mobile) as infect2021 from**

**(select distinct mobile from testreport WHERE**

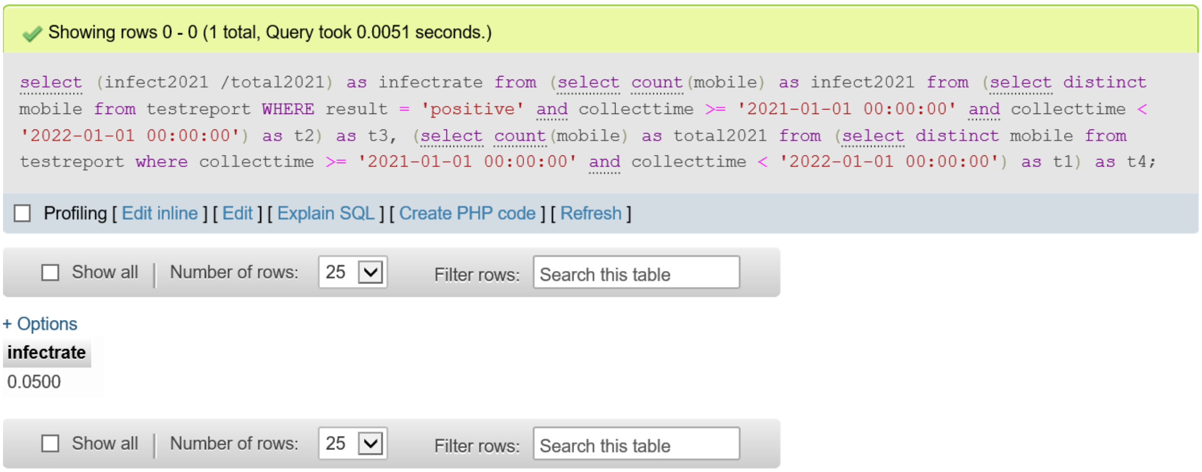
**result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t2) as t3,**

**(select count(mobile) as total2021 from**

**(select distinct mobile from testreport where**

**collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t1) as t4**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 0 (1 total, Query took 0.0051 seconds.)

```
select (infect2021 /total2021) as infectrate from (select count(mobile) as infect2021 from (select distinct
mobile from testreport WHERE result = 'positive' and collecttime >= '2021-01-01 00:00:00' and collecttime <
'2022-01-01 00:00:00') as t2) as t3, (select count(mobile) as total2021 from (select distinct mobile from
testreport where collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') as t1) as t4;
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this table
+ Options

| infectrate |
|---|
| 0.0500 |

□ Show all | Number of rows: 25 Filter rows: Search this table

***Use case 7***:

District ‘a’ is one of the districts in Lukewarm Kingdom. The number of people infected in district ‘a’ got substantial increase in a short period. The doctors working in district ‘a’ became very busy every day during this special period. There are multiple doctors capable of collecting samples. Assume the time that doctors responsible for collecting viral samples in district ‘a’ start to work is the same, but the time that doctors are off duty is different. Please write a query to find the hospital which the average off-duty time of doctors responsible for collecting samples is the latest on ‘2021-10-04’.

Your SQL statement:

**select hospitalname from**

**(select max(avgtime) as latest, hospitalname from**

**(select avg(offdutytime) as avgtime,hospitalname from**

```
(select max(collecttime) as offdutytime, doctorname,hospitalname from testreport natural join testdoctor
WHERE
doctorname in
(select doctorname from testdoctor WHERE
hospitalname IN
(select hospitalname from testhospital
where districtname = 'a')) and collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00'
group by doctorname) as t0
group by hospitalname) as t1) as t2
```

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

**The following viral test report information is added to the testreport table (some attributes are hidden as they are not related to this task).**

| doctorname | collecttime |
|---|---|
| ‘Alen’ | ‘2021-10-04 08:00:00’ |
| ‘Alen’ | ‘2021-10-04 21:00:00’ |
| ‘Amy’ | ‘2021-10-04 21:43:00’ |
| ‘Amy’ | ‘2021-10-04 15:55:00’ |
| ‘Alice’ | ‘2021-10-04 11:52:00’ |
| ‘Alice’ | ‘2021-10-04 22:32:00’ |
| ‘Bob’ | ‘2021-10-04 16:59:00’ |
| ‘Bob’ | ‘2021-10-04 21:12:00’ |
| ‘Billy’ | ‘2021-10-04 09:45:00’ |
| ‘Billy’ | ‘2021-10-04 21:12:00’ |
| ‘Betty’ | ‘2021-10-04 13:51:00’ |
| ‘Betty’ | ‘2021-10-04 21:21:00’ |
| ‘Cathy’ | ‘2021-10-04 21:00:00’ |
| ‘Cathy’ | ‘2021-10-04 11:09:00’ |
| ‘Cindy’ | ‘2021-10-04 20:30:00’ |
| ‘Cindy’ | ‘2021-10-04 11:00:00’ |
| ‘Dick’ | ‘2021-10-04 20:50:00’ |
| ‘Dave’ | ‘2021-10-04 11:00:00’ |
| ‘Curry’ | ‘2021-10-05 11:00:59’ |
| ‘Alen’ | ‘2021-10-05 11:01:47’ |

**The following information of doctors responsible for colleting samples is added to the testdoctor table**

| doctorname | hospitalname |
|---|---|
| ‘Alen’ | ‘aa’ |
| ‘Amy’ | ‘aa’ |
| ‘Alice’ | ‘aa’ |
| ‘Bob’ | ‘bb’ |

| | |
|---|---|
| ‘Billy’ | ‘bb’ |
| ‘Betty’ | ‘bb’ |
| ‘Cathy’ | ‘cc’ |
| ‘Cindy’ | ‘cc’ |
| ‘Curry’ | ‘cc’ |
| ‘Dave’ | ‘dd’ |
| ‘Dick’ | ‘dd’ |
| ‘Duck’ | ‘dd’ |

Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.

This test data set contains doctors working in the district ‘a’ and doctors not working in district ‘a’. this data set also contains doctors working on ‘2021-10-04’ and doctors working on other dates. Extend use case 7 requires ‘in district a’ and ‘on 2021-10-04’. So, the expected result of query should only contain the average latest off-duty time of doctors working in district ‘a’ and only on ‘2021-10-4’, which make it easy to finally select the hospital. Other doctors who are not confirm to the specific condition should be properly filtered out (doctors mentioned above are all responsible for collecting samples).

The detailed illustrations are as follows:

1.The first step is to select doctors working in district ‘a’ only on '2021-10-4'. The doctors working in other districts or did not work on ‘2021-10-4’ should be filtered out.

```
select collecttime, doctorname,hospitalname from testreport natural join testdoctor

WHERE

doctorname in

(select doctorname from testdoctor WHERE

hospitalname IN

(select hospitalname from testhospital

where districtname = 'a')) and collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00'
```

2.use max and avg function in sql to select the average off-duty time of every different hospital in district ‘a’ only on ‘2021-10-4’

```
select avg(offdutytime) as avgtime,hospitalname from

(select max(collecttime) as offdutytime, doctorname,hospitalname from testreport natural join testdoctor

WHERE

doctorname in

(select doctorname from testdoctor WHERE

hospitalname IN

(select hospitalname from testhospital

where districtname = 'a')) and collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00'
```

**group by doctorname) as t0**

**group by hospitalname**

**3.final step is to select the longest average off-duty time, then select that hospital.**

**select hospitalname from**

**(select max(avgtime) as latest, hospitalname from**

**(select avg(offdutytime) as avgtime,hospitalname from**

**(select max(collecttime) as offdutytime, doctorname,hospitalname from testreport natural join testdoctor**

**WHERE**

**doctorname in**

**(select doctorname from testdoctor WHERE**

**hospitalname IN**

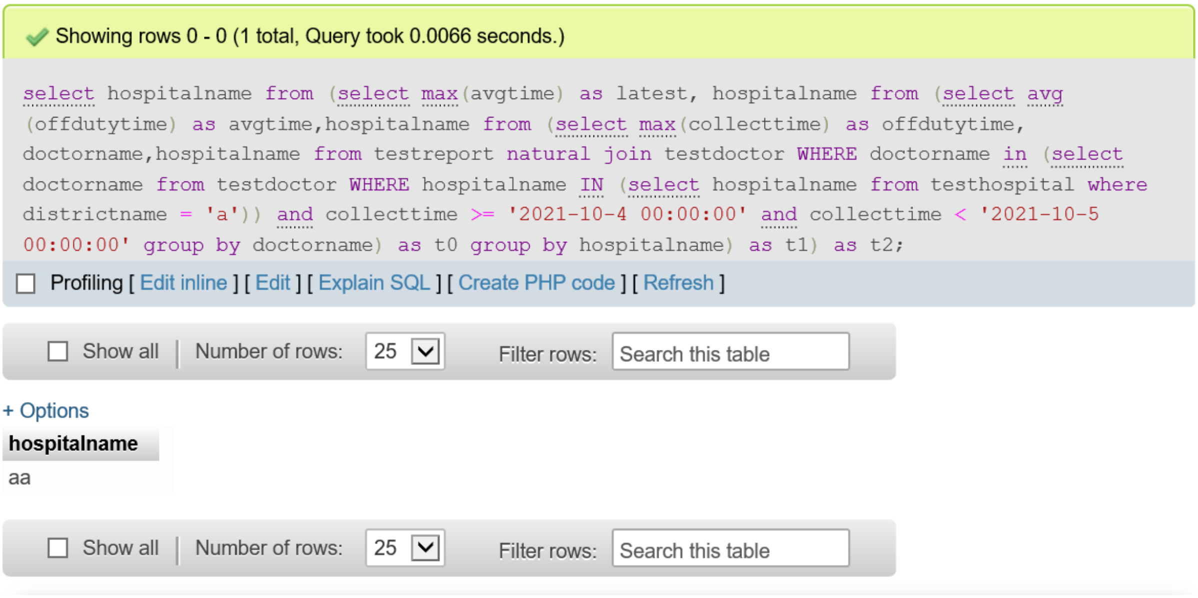
**(select hospitalname from testhospital**

**where districtname = 'a')) and collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5 00:00:00'**

**group by doctorname) as t0**

**group by hospitalname) as t1) as t2;**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 0 (1 total, Query took 0.0066 seconds.)

```
select hospitalname from (select max(avgtime) as latest, hospitalname from (select avg
(offdutytime) as avgtime,hospitalname from (select max(collecttime) as offdutytime,
doctorname,hospitalname from testreport natural join testdoctor WHERE doctorname in (select
doctorname from testdoctor WHERE hospitalname IN (select hospitalname from testhospital where
districtname = 'a')) and collecttime >= '2021-10-4 00:00:00' and collecttime < '2021-10-5
00:00:00' group by doctorname) as t0 group by hospitalname) as t1) as t2;
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this table
+ Options

| hospitalname |
|---|
| aa |

□ Show all | Number of rows: 25 Filter rows: Search this table

***Use case 8***:

Mark is a biological scientist in Lukewarm Kingdom. As this epidemic has broken out for a long time, Mark thought it is probable that the infection of this virus is related to people’s age. In Mark’s assumption, children and elders are the groups which are easier to be infected. Mark need infection data to analyze, then he plans to conduct some experiments and studies to show whether his assumption is correct or not. Please write a query to list in 2021 the number of positive cases at age of 0 to 12, 12 to 30, 30 to 60, over 60, respectively (assume all people in Lukewarm Kingdom have done at least one viral test in 2021. If a person is cured after he was infected, he is infected again. In this case, it'll be counted as 2 positive cases).

Your SQL statement:

```
select * from
(select count(mobile) as positivenumber12 from testpeople as tp natural join testreport as tr WHERE
(collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') and
age <= 12 and result = 'positive') as t1,
(select count(mobile) as positivenumber30 from testpeople as tp natural join testreport as tr WHERE
(collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') and
(age > 12 and age <= 30) and result = 'positive') as t2,
(select count(mobile) as positivenumber60 from testpeople as tp natural join testreport as tr WHERE
(collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') and
(age > 30 and age <= 60) and result = 'positive') as t3,
(select count(mobile) as positivenumberover60 from testpeople as tp natural join testreport as tr
WHERE
(collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') and
age > 60 and result = 'positive') as t4
```

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

**The following viral test report information is added to the testreport table (some attributes are hidden as they are not related to this task).**

| collecttime | result |
|---|---|
| '2021-08-04 08:00:00' | 'negative' |
| '2021-02-04 21:00:00' | 'negative' |
| '2021-10-04 21:43:00' | 'negative' |
| '2021-10-04 15:55:00' | 'positive' |
| '2021-10-04 11:52:00' | 'negative' |
| '2021-11-05 22:32:00' | 'negative' |
| '2021-12-04 16:59:00' | 'positive' |
| '2021-06-04 21:12:00' | 'negative' |
| '2021-10-04 09:45:00' | 'negative' |
| '2021-10-04 21:12:00' | 'positive' |
| '2021-10-04 13:51:00' | 'negative' |
| '2021-10-04 21:21:00' | 'negative' |
| '2021-10-04 21:00:00' | 'negative' |
| '2021-10-04 11:09:00' | 'positive' |
| '2021-10-04 20:30:00' | 'negative' |

| | |
|---|---|
| ‘2021-10-04 11:00:00’ | ‘positive’ |
| ‘2021-10-04 20:50:00’ | ‘positive’ |
| ‘2021-10-04 11:00:00’ | ‘negative’ |
| ‘2021-10-05 11:00:59’ | ‘negative’ |
| ‘2021-10-05 11:01:47’ | ‘positive’ |

**The following personal information of people who did the viral tests is added to this testpeople table (some attributes are hidden as they are not related to this task).**

| mobile | age |
|---|---|
| ‘000001’ | 6 |
| ‘000002’ | 8 |
| ‘000003’ | 11 |
| ‘000004’ | 12 |
| ‘000005’ | 4 |
| ‘000006’ | 24 |
| ‘000007’ | 29 |
| ‘000008’ | 18 |
| ‘000009’ | 16 |
| ‘000010’ | 21 |
| ‘000011’ | 54 |
| ‘000012’ | 60 |
| ‘000013’ | 33 |
| ‘000014’ | 41 |
| ‘000015’ | 45 |
| ‘000016’ | 63 |
| ‘000017’ | 76 |
| ‘000018’ | 81 |
| ‘000019’ | 71 |
| ‘000020’ | 69 |

**Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This test data set contains the viral test report date of people in Lukewarm Kingdom. The set contains people of all ages and people who were tested to be positive or not to be positive. The expected query should only contain people of all ages who were tested to be positive in 2021. People who were never tested to be positive or not in 2021 should be filtered out.**

**The detailed illustrations are as follows:**

**1.First step is to select all records which were be done in 2021. Then count all people among them who were tested to be positive at the target age groups. For example, the people at age of over 60 (the methods are similar):**

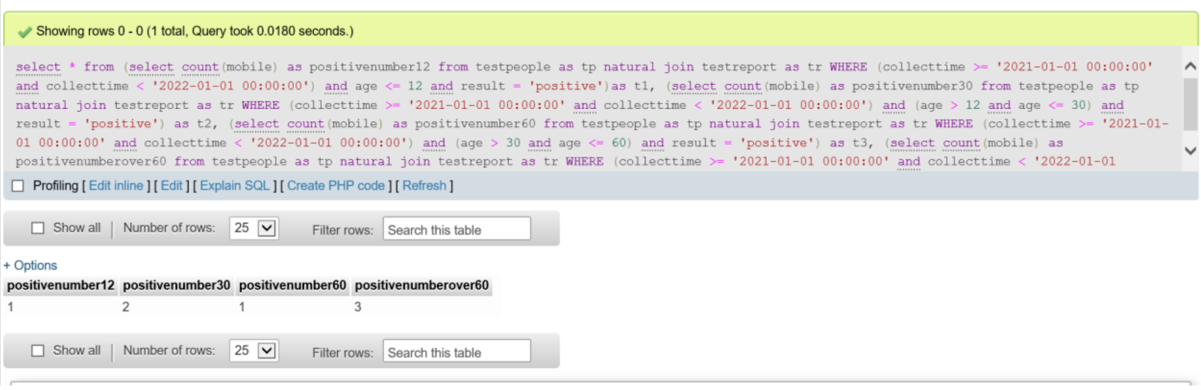
**select count(mobile) as positivenumberover60 from testpeople as tp natural join testreport as tr WHERE**

**(collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') and**

**age > 60 and result = 'positive'**

**2.final step is to combine all select queries in one select to represent the statistical data.**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 0 (1 total, Query took 0.0180 seconds.)

```
select * from (select count(mobile) as positivenumber12 from testpeople as tp natural join testreport as tr WHERE (collecttime >= '2021-01-01 00:00:00'
and collecttime < '2022-01-01 00:00:00') and age <= 12 and result = 'positive')as t1, (select count(mobile) as positivenumber30 from testpeople as tp
natural join testreport as tr WHERE (collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01 00:00:00') and (age > 12 and age <= 30) and
result = 'positive') as t2, (select count(mobile) as positivenumber60 from testpeople as tp natural join testreport as tr WHERE (collecttime >= '2021-01-
01 00:00:00' and collecttime < '2022-01-01 00:00:00') and (age > 30 and age <= 60) and result = 'positive') as t3, (select count(mobile) as
positivenumberover60 from testpeople as tp natural join testreport as tr WHERE (collecttime >= '2021-01-01 00:00:00' and collecttime < '2022-01-01
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this table
+ Options

| positivenumber12 | positivenumber30 | positivenumber60 | positivenumberover60 |
|---|---|---|---|
| 1 | 2 | 1 | 3 |

□ Show all | Number of rows: 25 Filter rows: Search this table

***Use case 9***:

Mark is the governor of central region in Lukewarm Kingdom. The epidemic broke out in district ‘a’ which locates in central region. The number of infected people in district ‘a’ increased fast every day. Finally, Mark decided to lock the whole district ‘a’ in order to keep people in other area safe. He determined that on ‘2021-10-04’ everyone no matter why he or she was in district ‘a’ on that day, he or she cannot move out of district ‘a’ until the government decides to open the district ‘a’ (which means the ‘lock district policy’ starts on ‘2021-10-04 00:00:00’). And other people cannot enter into district ‘a’. Please write a query to list the mobile number of all people who cannot move out of district ‘a’ since ‘2021-10-04’.

Your SQL statement:

**select simid from bsrecord WHERE**

**disconnectiontime is null and gpslocation IN**

**(select gpslocation from bslocation WHERE**

**corresponding_district = 'a')**

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

**The following travelling history information of people in Lukewarm Kingdom is added to the bsrecord table**

| gpslocation | simid | connectiontime | disconnectiontime |
|---|---|---|---|
| **’10,10’** | **‘000001’** | **‘2021-10-01 00:00:00’** | **Null** |
| **‘8,8’** | **‘000002’** | **‘2021-10-02 06:00:00’** | **‘2021-10-04 12:00:00’** |
| **‘7,7’** | **‘000002’** | **‘2021-10-04 12:00:00’** | **Null** |
| **’14,14’** | **‘000003’** | **‘2021-09-26 08:00:00’** | **‘2021-10-02 09:00:00’** |
| **’15,15’** | **‘000003’** | **‘2021-10-02 09:00:00’** | **Null** |
| **‘3,3’** | **‘000004’** | **‘2021-10-05 11:00:00’** | **Null** |
| **‘1,1’** | **‘000005’** | **‘2021-10-01 09:00:00’** | **Null** |
| **‘2,2’** | **‘000006’** | **‘2021-09-25 11:00:00’** | **Null** |
| **‘3,3’** | **‘000007’** | **‘2021-09-18 12:00:00’** | **‘2021-10-03 21:00:00’** |
| **‘2,2’** | **‘000007’** | **‘2021-10-03 21:00:00’** | **Null** |
| **‘1,1’** | **‘000008’** | **‘2021-09-20 00:00:00’** | **‘2021-10-02 20:00:00’** |
| **‘3,3’** | **‘000008’** | **‘2021-10-02 20:00:00’** | **‘2021-10-03 22:00:00’** |
| **‘1,1’** | **‘000008’** | **‘2021-10-03 22:00:00’** | **Null** |

| | | | |
|---|---|---|---|
| ‘2,2’ | ‘000009’ | ‘2021-08-01 12:00:00’ | Null |
| ‘1,1’ | ‘000010’ | ‘2021-09-01 08:00:00’ | Null |

**The corresponding GPS location information data above has been already added to the bslocation table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This data set contains the travelling history of all people in Lukewarm Kingdom, which means this data set contains the people who cannot move out of the district ‘a’ no matter he or she is a resident in district ‘a’ or just pass by in that special period. This data set also contains people who are not in district ‘a’ in that special period, which means these people do not need to be locked in district ‘a’. The expected result of the query should only contain people who cannot move out of the district. The people who are not in district ‘a’ in that special period should be filtered out.**

**The detailed illustrations are as follows:**

**1.the people who stayed in district ‘a’ need to be selected, which means that the disconnection time is null in the base station record database (it means the person did not move out of ‘a’).**

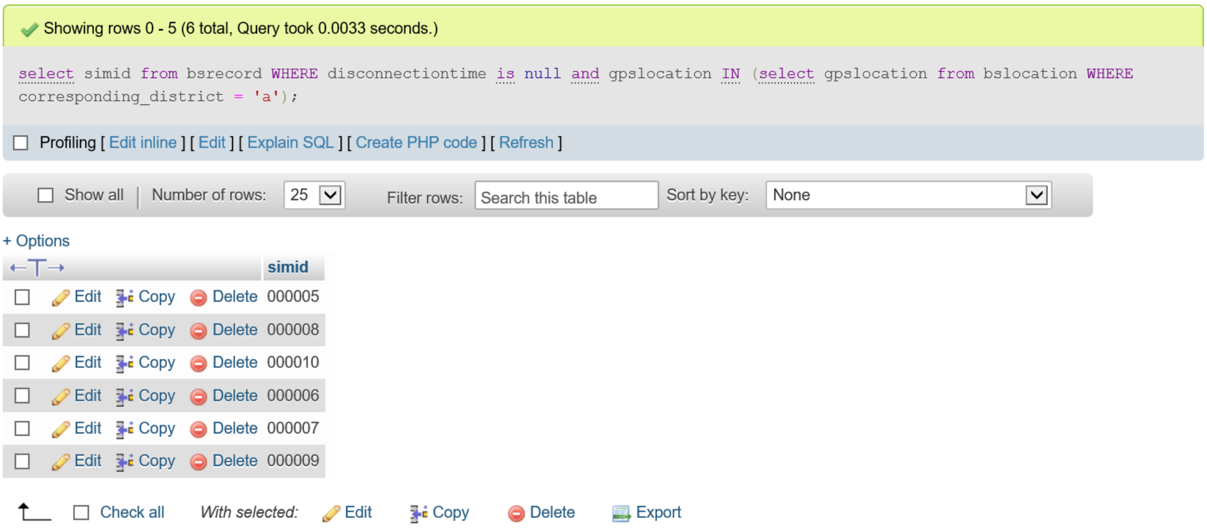
**select simid from bsrecord WHERE**

**disconnectiontime is null and gpslocation IN**

**(select gpslocation from bslocation WHERE**

**corresponding_district = 'a')**

The result of the SELECT statement (screenshot):



Showing rows 0 - 5 (6 total, Query took 0.0033 seconds.)

```
select simid from bsrecord WHERE disconnectiontime is null and gpslocation IN (select gpslocation from bslocation WHERE
corresponding_district = 'a');
```

| simid |
|---|
| 000005 |
| 000008 |
| 000010 |
| 000006 |
| 000007 |
| 000009 |

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]
□ Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None
+ Options
Edit Copy Delete
□ Check all With selected: Edit Copy Delete Export

***Use case 10***:

Mark is one who loves travelling. And he went to north region in Lukewarm Kingdom for travel. According to the policy related to epidemic prevention, a traveler needs to do the viral test both before he went to another place and after went to another place. Mark has already come to the north region, he needs to find a hospital capable of doing the viral test. Please write a query to list all hospitals capable of doing viral test in north region.

Your SQL statement:

**select hospitalname from testhospital WHERE**

**districtname in**

**(select districtname from risklevel WHERE**

**regionname = 'north')**

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

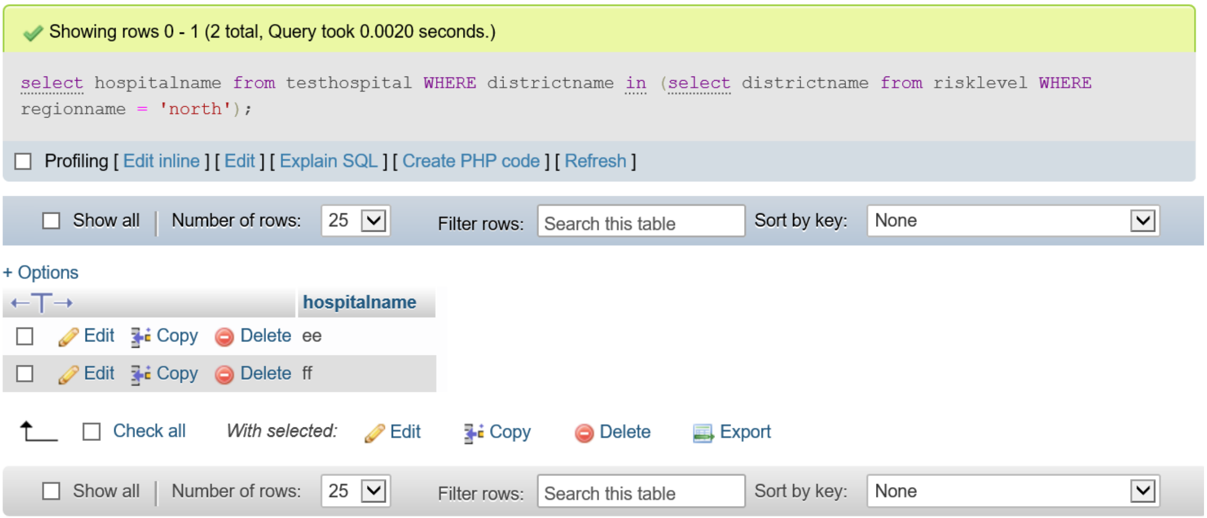
**The following information of hospital capable of doing viral test is added to the testhospital table.**

| **hospitalname** | **districtname** |
|---|---|
| **‘aa’** | **‘a’** |
| **‘bb’** | **‘a’** |
| **‘cc’** | **‘b’** |
| **‘dd’** | **‘b’** |
| **‘ee’** | **‘c’** |
| **‘ff’** | **‘d’** |
| **‘gg’** | **‘e’** |
| **‘hh’** | **‘f’** |
| **‘ii’** | **‘g’** |
| **‘jj’** | **‘h’** |
| **‘kk’** | **‘i’** |
| **‘ll’** | **‘j’** |

**The corresponding district names data above have already been added to the risklevel table. Some other information has already added to the other corresponding tables as there are related constraints of the foreign keys, but these data are hidden as they are not related to this task.**

**This data set contains all hospitals capable of doing the viral test in Lukewarm Kingdom, which means the set contains hospitals in district ‘a’ and hospitals not in district ‘a’. the expected result of the query should only contain hospitals in north region (one of the five region in Lukewarm Kingdom). The hospitals in other regions should be properly filtered out.**

The result of the SELECT statement (screenshot):



✓ Showing rows 0 - 1 (2 total, Query took 0.0020 seconds.)

```
select hospitalname from testhospital WHERE districtname in (select districtname from risklevel WHERE
regionname = 'north');
```

□ Profiling [ Edit inline ] [ Edit ] [ Explain SQL ] [ Create PHP code ] [ Refresh ]

□ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

+ Options

| ←T→ | hospitalname |
|---|---|
| □ Edit Copy Delete | ee |
| □ Edit Copy Delete | ff |

□ Check all  With selected: Edit Copy Delete Export

□ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None