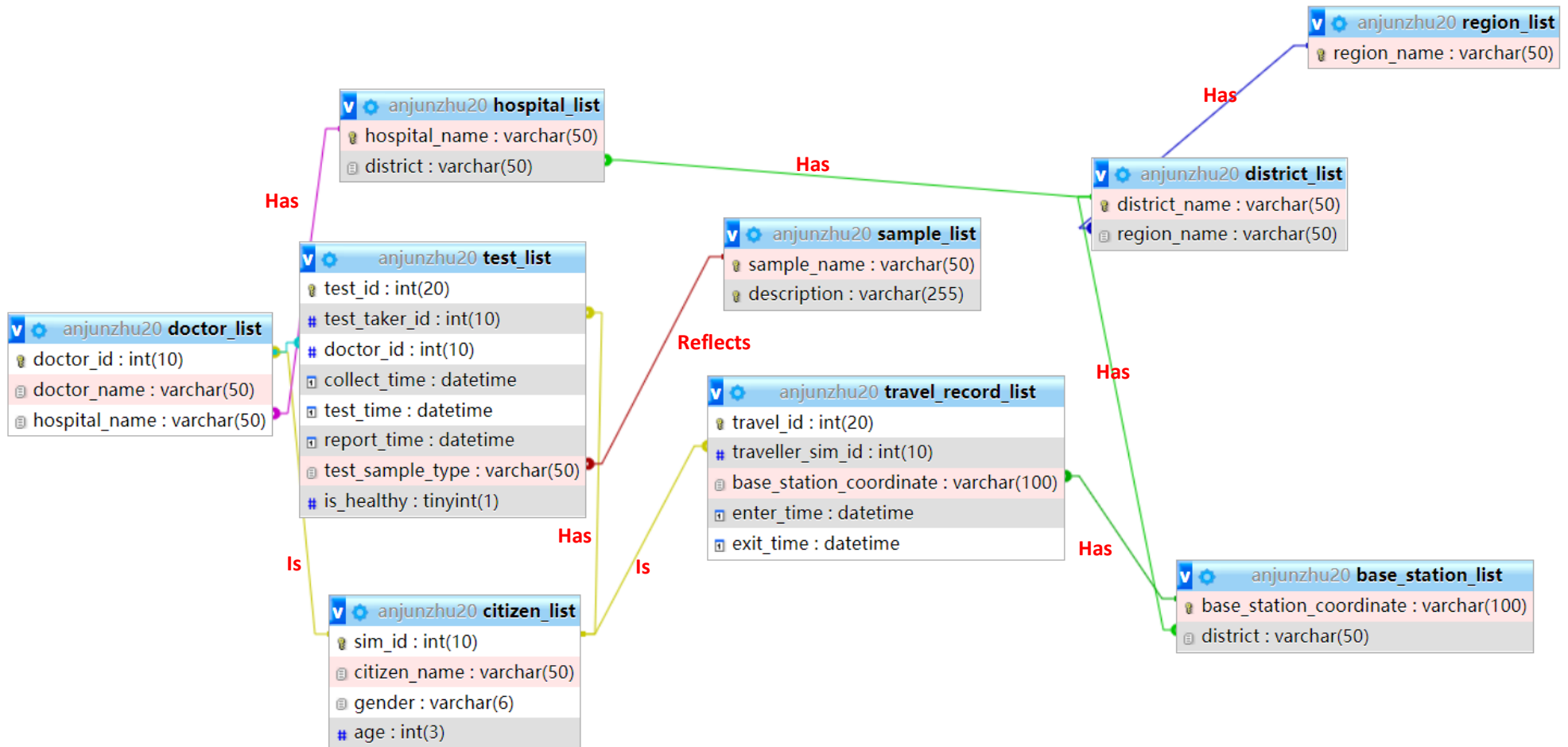


Your ER Diagram here.

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



Database Design Details

Tables

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

Table name: citizen_list

Table design explanation:

The citizen_list table is used to store the information of all citizens in the kingdom. The primary key is sim_id. Because it represents the phone number of each citizen and is unique for everybody. Name, gender and age are all fundamental and important pieces of information for describing and identifying each citizen in the kingdom, as well as the real life. Each row can be regard as a simplified version of citizen ID card.

| Column Definition | Domain | Explanation |
|--------------------------------------|---|--|
| sim_id int(10) PRIMARY KEY | A unique 10-digit long sequence of numbers, such as '8008101818'. | Citizen's phone number must be 10 digits long. It strictly follows the format of TTT-R-DDD-SSS, such as 800-8-101-818. 'T' stands for the telecom operator of the number. 'R' stands for region, 'D' for district and 'S' for sequence. The first 3 digits can represent different telecom operators, which is the same as real life, where China Telecom, China Unicom and China Mobile offer different phone numbers with different starting digits for customers respectively. The fourth digit refer to the region the number belongs, followed by 3 digits representing the belonging district. Then next 3 digits represent the sequence code of the phone number. |
| citizen_name varchar(50) NOT null | All valid names are acceptable, such as 'Anjun Zhu'. | Citizens' name, which can be up to 50 characters long. Since people can share a same name, this column is not unique. |
| gender varchar(6) NOT null | Both 'male' and 'female' can be filled in. | Citizens' gender. 'Female' is 6-character long and 'male' is 4, so the length can be up to 6, but should not be longer than 6. |
| age int(3) NOT null | The range:0-999. In case there are some elder citizens, whose is over 99 years old. | Citizens' age. Since people can share one age, this column is not unique. |

Foreign keys and reasons: No foreign key in this table.

Table name: base_station_list

Table design explanation: The base_station_list stores all information of each base station in the kingdom. The primary key is base_station_coordinate. Because it stores the unique GPS location of each

base station, and can be referred by the table travel_record_list. The column district stores the name of each district every base station belongs to.

| Column Definition | Domain | Explanation |
|--|---|---|
| base_station_coordinate varchar(100) PRIMARY KEY | All valid coordinates are acceptable, and they should follow the format 'latitude, longitude', such as '10101.1,2098.84'. | This column stores the unique GPS location of each base station. The total digits can be up to 100, which is enough to represent the GPS locations of all base stations in the kingdom. |
| district varchar(50) NOT null | All valid names are acceptable, such as 'Centre Lukewarm Hillside'. | The name of each district every base station belongs to. |

Foreign keys and reasons: No foreign key.

Table name: district_list

Table design explanation: The district_list table is used to store the information of all districts of all regions in the kingdom. The primary key is district_name. Because it is unique and represents the name of each district.

| Column Definition | Domain | Explanation |
|--|---|---|
| district_name varchar(50) NOT null PRIMARY KEY | All valid names are acceptable, such as 'Centre Lukewarm Hillside'. | The name of each district, which can be up to 50 characters long. |
| region_name varchar(50) NOT null | All valid names are acceptable, such as 'Central Region'. | The name of each region the district belongs to, which can be up to 50 characters long. |

Foreign keys and reasons: The column region_name refers to region_list.region_name, which makes sure that region_name are valid names that reflect existing regions of the kingdom. It corresponds to the Region-includes-districts (1:M) relationship in the ER diagram.

Table name: doctor_list

Table design explanation: The doctor_list table is used to store the information of all current doctors in the kingdom. The primary key is doctor_id. Because it represents the phone number of each doctor, which is unique. Frequent update should be carried out in case hospitals recruit new doctors or resign doctors. Since in real life, doctors can transfer to different hospitals. Each hospital has more than one doctors, whereas doctors can have working experience in more than one hospital. It is true in Lukewarm Kingdom as well, thus there is an M:M relationship between the doctors and hospitals, and this doctor_list is needed to convert this M:M relationship into two 1:M relationships.

| Column Definition | Domain | Explanation |
|--|---|---|
| doctor_id int(10) NOT null PRIMARY KEY | A unique 10-digit long sequence of numbers, such as '8008101818'. | This number refers to the phone number of each doctor. Since each citizen of the kingdom has and only has one phone number, it can help identify the identification of the doctor. |
| doctor_name varchar(50) NOT null | All valid names are acceptable, such as 'Anjun Zhu'. | The name of each doctor. Since some people may share one name, the column cannot be set as primary key. However, in most of the circumstances in real life, it is doctor's name that is recorded. |
| hospital_name varchar(50) NOT null | All valid names are acceptable, such as 'Centre Lukewarm Hillside'. | The name of each hospital each doctor belongs to. One hospital can have many doctors, hence it should not be a unique column. |

Foreign keys and reasons: There are two foreign keys in this table. The column doctor_id refers to citizen_list.sim_id, which makes sure that doctor_id are valid and unique phone numbers that identify only one single doctor respectively. As a constraint for valid references, it corresponds to the Citizen-is-doctors (1:1) relationship in the ER diagram. The column hospital_name refers to hospital_list.hospital_name, which makes sure that hospital_name are valid and unique names that identify only one single hospital respectively. This is because in real life the name of each hospital is unique. As a constraint for valid references, it corresponds to the Hospital-has-name (1:1) relationship in the ER diagram.

Table name: hospital_list

Table design explanation: The hospital_list table is used to store the information of all hospitals in the kingdom. The primary key is hospital_name. Because it represents the name of each hospital, which is unique.

| Column Definition | Domain | Explanation |
|--|--|---|
| hospital_name varchar(50) NOT null PRIMARY KEY | All valid names are acceptable, such as 'Central Lukewarm Kingdom Hospital'. | The name of each hospital, which can be up to 50 characters long. Since each name of the hospital is unique, which is the same as real life hospital names. It can help identify the exact hospital |
| district varchar(50) NOT null | All valid names are acceptable, such as 'Centre Lukewarm Hillside'. | The name of each district every hospital belongs to. |

Foreign keys and reasons: The column district refers to district_list.district_name, which makes sure that 'district' are valid and reflect existing districts of the kingdom. It corresponds to the District-has-hospitals (1:M) relationship in the ER diagram.

Table name: region_list

Table design explanation: The hospital_list table is used to store the information of all hospitals in the kingdom and provide constraint for other involving tables. The primary key is region_name. Because it represents the name of each region, which is unique.

| Column Definition | Domain | Explanation |
|--|---|---|
| region_name varchar(50) NOT null PRIMARY KEY | All valid names are acceptable, such as 'Central Region'. | The name of each region, which can be up to 50 characters long. Since each name of the region is unique, which is the same as real life region names. It can help identify the exact region |

Foreign keys and reasons: No foreign key.

Table name: sample_list

Table design explanation: The sample_list table is used to store the information of all sample types found in the kingdom and provide constraint for other involving tables. The primary key is sample_name. Because it represents the name of each sample type, which is unique.

| Column Definition | Domain | Explanation |
|--|---|--|
| sample_name varchar(50) NOT null PRIMARY KEY | All valid names are acceptable, such as 'Coughid-19'. | This column is designed to describe the name of the sample types, which can be up to 50 characters long. Since each name of the sample type is unique, which is the same as real life sample type names. It can help identify the exact sample type. |
| description varchar(255) NOT null UNIQUE | All valid descriptions are acceptable. For example, 'Coughid-21' has a unique description 'Coughid-21 is a newly identified type of virus this year, all patients tested to e positive should rest well and avoid going outside'. | This column can store the description of each sample type. Since some descriptions can be very long, the length of this column is 255. |

Foreign keys and reasons: No foreign key.

Table name: test_list

Table design explanation: The report_list table is used to store the information of all tests of all citizens in the kingdom. The primary key is test_id. Because it represents the sequence of each test, which is unique.

| Column Definition | Domain | Explanation |
|---|---|--|
| test_id int(20) NOT null PRIMARY KEY AUTO_INCREMENT | A unique 20-digit long sequence of numbers with auto increment. | This column can record and identify every test of all test takers. Since there may be many tests in the future, the test id is 20-digit long. |
| test_taker_id int(10) NOT null | A unique 10-digit long sequence of numbers, such as '8008101818'. | This column can help identify the test taker's identification throughout the phone number. |
| doctor_id int(10) NOT null | A unique 10-digit long sequence of numbers, such as '8008101818'. | This number refers to the phone number of each doctor. Since the doctor is responsible for the test, it can help identify the identification of the doctor. The reason for not using doctor name instead is that sometimes two or more doctors can share a same name, even in one hospital. Hence it might be difficult to identify the exact doctor simply by searching the name. Only the phone number can one exact doctor be identified. |
| collect_time datetime NOT null | All valid datetime are acceptable, such as '2021-10-03 00:00'. | The column stores the sample collect time of every test. |
| test_time datetime NOT null | All valid datetime are acceptable, such as '2021-10-03 00:00'. | The column stores the test time of every test. |
| report_time datetime NOT null | All valid datetime are acceptable, such as '2021-10-03 00:00'. | The column stores the report time of every test. |
| test_sample_tyle varchar(50) NOT null | All valid names are acceptable, such as 'Coughid-19'. | This column is designed to describe the name of the test sample type. |
| is_healthy tinyint(1) NOT null | Only '0' or '1' is accepted | This column refers to whether the test taker has the test sample type. '1' indicates that the test taker is healthy, while '0' indicates not. |

Foreign keys and reasons: There are several foreign keys in this table. The column patient_id refers to citizen_list.sim_id, which makes sure that patient_id are valid and reflect existing phone number of all citizens in the kingdom. As a constraint for valid references, it corresponds to the Patient-is-citizen (1:1) relationship in the ER diagram. It corresponds to the Tests-take place in-hospitals (M: 1) relationship in the ER diagram. The column doctor_id refers to doctor_list.doctor_id, which makes sure that doctor_id are valid and can reflect all current doctors in the kingdom. As a constraint for valid references, it corresponds to the Doctor-is-citizen (1:1) relationship in the ER diagram. The column result refers to sample_list.sample_name, which makes sure that the column results are valid and can reflect all existing sample types. As a constraint for valid references, it corresponds to the Result-reflects-sample type (1:1) relationship in the ER diagram.

Table name: travel_record_list

Table design explanation: The travel_record_list table is used to store the travelling information of all citizens in the kingdom. The primary key is test_id. Because it represents the sequence of each test, which is unique. Whenever a traveller enters a new base station area, one travel_id is generated. This

ensures the whole travelling record of every citizen in the kingdom can be tracked and traced when needed.

| Column Definition | Domain | Explanation |
|---|---|---|
| travel_id int(20) NOT null PRIMARY KEY AUTO_INCREMENT | A unique 20-digit long sequence of numbers with auto increment. | This column can record and identify every travelling of all citizens. Since there may be many travelling records in the future, the id number is 20-digit long. |
| traveller_sim_id int(10) NOT null | A unique 10-digit long sequence of numbers, such as '8008101818'. | This column can help identify the traveller's identification throughout the phone number. |
| base_station_coordinate varchar(100) NOT null | All valid coordinates are acceptable, and they should follow the format 'latitude, longitude', such as '10101.1,2098.84'. | All valid coordinates are acceptable, and they should follow the format 'latitude, longitude', such as '10101.1,2098.84'. |
| enter_time DATETIME NOT null | All valid datetime are acceptable, such as '2021-10-03 00:00'. | The column stores the enter time to the base station of every travelling record. |
| exit_time DATETIME | All valid datetime are acceptable, such as '2021-10-03 00:00'. | The column stores the exit time from the base station of every travelling record. This information can be null, if the traveller stays in the same area of the same base station. |

Foreign keys and reasons: There are two foreign keys in this table. The column traveller_sim_id refers to citizen_list.sim_id, which makes sure that traveller_sim_id are valid and reflect existing phone number of all citizens in the kingdom. As a constraint for valid references, it corresponds to the Traveller-is-citizen (1:1) relationship in the ER diagram. The column base_station_coordinate refers to base_station_list.base_station_coordinate, which makes sure that base_station_coordinate are valid and can reflect all existing base station of the kingdom. It corresponds to the Travel record-has-base station coordinate(1:1) relationship in the ER diagram.

[add more blocks if needed]

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 positive 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:

{Name of the test taker, Sex of the test taker, Age of the test taker, Mobile number of the test taker, Sample type of the test, Sample result, Sample description, Sample collect time, Sample test time, Report time, Doctor name, Doctor phone number*, Hospital name}

For example, in the test report above: {Jianjun Chen, Male, 70, 8008101818, Coughid-21, Positive, Coughid-21 is a newly identified type of virus this year, all patients tested to be positive should rest well and avoid going outside, 2020/10/1 13:27, 2020/10/1 14:50, 2020/10/1 17:20, Jun Qi, not mentioned, Central Lukewarm Kingdom Hospital}

*The reason for 'Doctor phone number' has been mention above. To learn more detail, please find the explanation in 'doctor_id' of test_list in the table section.

FDs (Indicate partial or transitive dependencies):

{Mobile number of the test taker} can imply -> {Name of the test taker, Sex of the test taker, Age of the test taker}

{Sample type of the test} can imply -> {Sample description}

{Doctor phone number} can imply -> {Doctor name}

Normalised tables and which normal form they are currently in:

{Mobile number of the test taker, Name of the test taker, Sex of the test taker, Age of the test taker} 3NF

{Sample type of the test, Sample description} 3NF

{Doctor phone number, Doctor name} 3NF

{Mobile number of the test taker, Sample type of the test, Sample result, Sample collect time, Sample test time, Report time, Doctor phone number, Hospital name} 3NF

Stage 2

Attributes:

{Mobile number of the test taker, Name of the test taker, Sex of the test taker, Age of the test taker}

{Sample type of the test, Sample description}

{Doctor phone number, Doctor name}

{Mobile number of the test taker, Sample type of the test, Sample result, Sample collect time, Sample test time, Report time, Doctor phone number, Hospital name}

FDs (Indicate partial or transitive dependencies):

{Doctor phone number} can imply -> {Hospital name}

Normalised tables and which normal form they are currently in:

{Mobile number of the test taker, Name of the test taker, Sex of the test taker, Age of the test taker} 3NF

{Sample type of the test, Sample description} 3NF

{Doctor phone number, Doctor name} 3NF

{Mobile number of the test taker, Sample type of the test, Sample collect time, Sample test time, Report time, Doctor phone number, } 2NF

Use Cases

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

Important Use Cases

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

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 distinct traveller_sim_id FROM ((travel_record_list LEFT OUTER JOIN base_station_list ON
travel_record_list.base_station_coordinate=base_station_list.base_station_coordinate) LEFT OUTER JOIN
district_list ON base_station_list.district=district_list.district_name)
WHERE(((base_station_list.district='Lenny town')AND((exit_time>='2021-10-07
19:30:00')AND(enter_time<='2021-10-08 21:50:00'))
AND(traveller_sim_id<>233636))OR((base_station_list.district='Centre Lukewarm
Hill')AND((exit_time>='2021-10-08 21:50:00')AND(enter_time<='2021-10-09 19:30:00'))
AND(traveller_sim_id<>233636))
```

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

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

| Sim_id | Citizen_name |
|-----------|-------------------|
| 800810000 | Michail Antonio |
| 800810001 | Andriy Yarmolenko |
| 800810002 | Said Benrahma |
| 800810003 | Nikla Vlasic |
| 800810004 | Declan Rice |
| 800810005 | Tomas Soucek |
| 233636 | Mark |
| 800810020 | Alphonse Areola |

The following information is added to the table travel_record_list:

| Travel_id | Traveller_sim_id | Base_station_coordinate | Enter_time | Exit_time |
|-----------|------------------|-------------------------|------------------------|------------------------|
| 1 | 800810000 | 10101.1,2098.84 | 2021-10-06 19:30:00 | 2021-10-07 19:40:00 |

| | | | | |
|----|-----------|-----------------|------------------------|------------------------|
| 2 | 800810000 | 22304.4,7338 | 2021-10-07 19:40:00 | 2021-10-09 19:40:00 |
| 3 | 800810001 | 22304.4,7338 | 2021-10-07 19:20:00 | 2021-10-09 19:40:00 |
| 4 | 800810001 | 10101.1,2098.84 | 2021-10-09 19:40:00 | 2021-10-09 19:50:00 |
| 5 | 800810002 | 10101.1,2098.84 | 2021-10-06 19:35:00 | 2021-10-06 19:40:00 |
| 6 | 800810002 | 22304.4,7338 | 2021-10-06 19:40:00 | 2021-10-09 19:35:00 |
| 7 | 800810003 | 22304.4,7338 | 2021-10-06 19:20:00 | 2021-10-07 19:50:00 |
| 8 | 800810003 | 10101.1,2098.84 | 2021-10-07 19:50:00 | 2021-10-07 20:50:00 |
| 9 | 800810003 | 60204.4,7798.84 | 2021-10-07 20:50:00 | 2021-10-08 20:50:00 |
| 10 | 800810003 | 22304.4,7338 | 2021-10-08 20:50:00 | 2021-10-09 19:40:00 |
| 11 | 800810004 | 22304.4,7338 | 2021-10-06 10:50:00 | 2021-10-08 22:50:00 |
| 12 | 800810004 | 60204.4,7798.84 | 2021-10-08 22:50:00 | 2021-10-09 22:50:00 |
| 13 | 800810004 | 10101.1,2098.84 | 2021-10-09 22:50:00 | |
| 14 | 800810005 | 22304.4,7338 | 2021-10-06 12:50:00 | 2021-10-06 13:50:00 |
| 15 | 800810005 | 10101.1,2098.84 | 2021-10-06 13:50:00 | 2021-10-09 19:50:00 |
| 16 | 800810005 | 22304.4,7338 | 2021-10-09 19:50:00 | 2021-10-09 19:55:00 |
| 17 | 233636 | 22304.4,7338 | 2021-10-06 13:50:00 | 2021-10-07 19:30:00 |
| 18 | 233636 | 10101.1,2098.84 | 2021-10-07 19:30:00 | 2021-10-07 19:50:00 |
| 19 | 233636 | 10201.1,2098.84 | 2021-10-07 19:50:00 | 2021-10-08 21:50:00 |
| 20 | 233636 | 60104.4,7798.84 | 2021-10-08 21:50:00 | 2021-10-08 22:50:00 |
| 21 | 233636 | 60204.4,7798.84 | 2021-10-08 22:50:00 | 2021-10-09 13:50:00 |
| 22 | 233636 | 60304.4,7798.84 | 2021-10-09 13:50:00 | 2021-10-09 19:30:00 |
| 24 | 800810020 | 63615.5,8964.2 | 2021-10-07 19:30:00 | 2021-10-12 19:30:00 |

The following information is added to the table travel_record_list:

| Base_station_coordinate | district |
|-------------------------|--------------------------|
| 60304.4,7798.84 | Centre Lukewarm Hillside |
| 60204.4,7798.84 | Centre Lukewarm Hillside |
| 60104.4,7798.84 | Centre Lukewarm Hillside |

| | |
|-----------------|--------------------|
| 10201.1,2098.84 | Lenny town |
| 10101.1,2098.84 | Lenny town |
| 22304.4,7338 | Glow Sand district |
| 63615.5,8964.2 | Raspberry town |

The following information is added to the table region_list:

| Region_name |
|----------------|
| North Region |
| South Region |
| East Region |
| West Region |
| Central Region |

The following information is added to the table district_list:

| District_name | Region_name |
|--------------------------|----------------|
| Centre Lukewarm Hillside | Central Region |
| Glow Sand district | East Region |
| Lenny town | East Region |
| Raspberry town | West Region |

This data set contains people with travelling records around 2021-10-09 19:30:00, which is the exact time when Mark was tested positive. To better illustrate this, the following graph shows the travel sequence of Mark and these people.



In the past 48 hours before his positive report was published, Mark travelled Lenny town to Centre Lukewarm Hill. And in Lukewarm Hill, he was found positive.

Michail Antonio (ID 800810000) left Lenny town later than Mark's arrival, and then Michail moved to a safe district.

Andriy Yarmolenko (ID 800810001) stayed in safe districts. And after Mark's positive report published, Andriy then entered Lenny town.

Said Benrahma (ID 800810002) left Lenny town earlier than Mark's arrival, and then he moved to a safe district.

Nikola Vlasic (ID 800810003) entered Lenny town later than Mark's arrival, exited and entered Centre Lukewarm Hill. He exited there and went to a safe district before Mark's arrival.

Declan Rice (ID 800810004) entered Centre Lukewarm Hill later than Mark's arrival, and exited after Mark's positive record was published.

Tomas Soucek (ID 800810005) entered Lenny town earlier than Mark's arrival, and exited after Mark's positive record was published.

Alphonse Areola (ID 800810020) stayed in the Raspberry town all the time, which was a safe district.

It is required that if someone is tested to positive, all people in the same district as Mark in the past 48 hours (before the positive report is published) need to take viral tests. Since Mark was tested to be positive at 1930 on 09-Oct-2021, and assume he went to Lenny town and Centre Lukewarm Hill in these 48 hours, anyone left after Mark's arrival and arrived earlier than Mark's exit in the same district as him should take viral test. In this case, Michail Antonio (ID 800810000), Nikola Vlasic (ID 800810003) and Tomas Soucek (ID 800810005) should take the test.

The result of the SELECT statement (screenshot):



```
SELECT distinct traveller_sim_id FROM ((travel_record_list LEFT OUTER JOIN base_station_list ON travel_record_list.base_station_coordinate=base_station_list.base_station_coordinate) LEFT OUTER JOIN district_list ON base_station_list.district=district_list.district_name) WHERE ((base_station_list.district='Lenny tom') AND ((exit_time>='2021-10-07 19:30:00') AND (enter_time<='2021-10-08 21:50:00')) AND (traveller_sim_id<>'233636')) OR ((base_station_list.district='Centre Lukewarm Hill') AND ((exit_time>='2021-10-08 21:50:00') AND (enter_time<='2021-10-09 19:30:00')) AND (traveller_sim_id<>'233636'));
```

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0003 秒。)

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

☐ 显示全部 | 行数: 25 | 过滤行: 在表中搜索

+ 选项

| traveller_sim_id |
|------------------|
| 800810000 |
| 800810003 |
| 800810005 |

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):

The GPS format is (latitude, longitude) stored in the format of varchar. This follows the first normalization since in this project the specific search requirement of latitude and longitude respectively is not needed.

<https://pretagteam.com/question/gps-radius-search-with-php-5-and-mysql>

Your SQL statement(s) for travel record insertion:

```
INSERT INTO travel_record_list VALUES(23,800810000,'22304.4,7338','2021-12-18 19:30:00','2021-12-18 20:30:00');
```

This statement shows one citizen with his sim id 800810000 enters the rage of the base station with its GPS location 22304.4 in latitude and 7338 in longitude on 2021-12-18 19:30:00, and exits one hour later. The result of all SQL statements (screenshot):

```

✓ 插入了 1 行。(查询花费 0.0220 秒。)

INSERT INTO travel_record_list VALUES (23,800810000,'22304.4,7338','2021-12-18 19:30:00','2021-12-18 20:30:00')

```

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 hospital_name, AVG(UNIX_TIMESTAMP(report_time) - UNIX_TIMESTAMP(test_time)) AS
avg_time FROM (test_list RIGHT OUTER JOIN doctor_list ON
test_list.doctor_id=doctor_list.doctor_id)GROUP BY doctor_list.hospital_name ORDER BY avg_time ASC
LIMIT 1

```

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

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

| Sim_id | Citizen_name |
|-----------|----------------|
| 800810006 | Jarrod Bowen |
| 800810007 | Pablo Fornals |
| 800810008 | Manuel Lanzini |
| 800810009 | Arthur Masuaku |
| 800810010 | Alex Kral |

The following citizen information is added to the table hospital_list:

| Hospital_name | district |
|-----------------------------------|--------------------------|
| Central Lukewarm Kingdom Hospital | Centre Lukewarm Hillside |
| Lenny Town Hospital | Lenny town |
| Glow Sand Hospital | Glow Sand district |
| Raspberry Hospital | Raspberry town |

The following citizen information is added to the table doctor_list:

| Doctor_id | Doctor_name | Hospital_name |
|-----------|--------------|-----------------------------------|
| 800810006 | Jarrod Bowen | Central Lukewarm Kingdom Hospital |

| | | |
|-----------|----------------|---------------------|
| 800810007 | Pablo Fornals | Lenny Town Hospital |
| 800810008 | Manuel Lanzini | Glow Sand Hospital |
| 800810009 | Arthur Masuaku | Raspberry Hospital |

The following citizen information is added to the table sample_list:

| sample_name | description |
|-------------|--|
| Coughid-21 | Coughid-21 is a newly identified type of virus this year, all patients tested to be positive should rest well and avoid going outside. |

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|------------------------|------------------------|------------------------|------------------|------------|
| 1 | 800810010 | 800810006 | 2021-10-01 13:50:00 | 2021-10-01 14:50:00 | 2021-10-01 15:50:00 | Coughid-21 | 1 |
| 2 | 800810010 | 800810007 | 2021-10-02 13:50:00 | 2021-10-02 14:50:00 | 2021-10-02 16:50:00 | Coughid-21 | 1 |
| 3 | 800810010 | 800810008 | 2021-10-03 13:50:00 | 2021-10-03 14:50:00 | 2021-10-03 17:50:00 | Coughid-21 | 1 |
| 4 | 800810010 | 800810009 | 2021-10-04 13:50:00 | 2021-10-04 14:50:00 | 2021-10-04 18:50:00 | Coughid-21 | 1 |
| 5 | 800810010 | 800810006 | 2021-10-05 13:50:00 | 2021-10-05 14:50:00 | 2021-10-05 15:50:00 | Coughid-21 | 1 |
| 6 | 800810010 | 800810007 | 2021-10-06 13:50:00 | 2021-10-06 14:50:00 | 2021-10-06 16:50:00 | Coughid-21 | 1 |
| 7 | 800810010 | 800810008 | 2021-10-07 13:50:00 | 2021-10-07 14:50:00 | 2021-10-07 17:50:00 | Coughid-21 | 1 |
| 8 | 800810010 | 800810009 | 2021-10-08 13:50:00 | 2021-10-08 14:50:00 | 2021-10-08 18:50:00 | Coughid-21 | 1 |
| 9 | 800810010 | 800810006 | 2021-10-09 13:50:00 | 2021-10-09 14:50:00 | 2021-10-09 15:50:00 | Coughid-21 | 1 |
| 10 | 800810010 | 800810007 | 2021-10-10 13:50:00 | 2021-10-10 14:50:00 | 2021-10-10 16:50:00 | Coughid-21 | 1 |
| 11 | 800810010 | 800810008 | 2021-10-11 13:50:00 | 2021-10-11 14:50:00 | 2021-10-11 17:50:00 | Coughid-21 | 1 |
| 12 | 800810010 | 800810009 | 2021-10-12 13:50:00 | 2021-10-12 14:50:00 | 2021-10-12 18:50:00 | Coughid-21 | 1 |

| | | | | | | | |
|----|-----------|-----------|---------------------|---------------------|---------------------|------------|---|
| 13 | 800810010 | 800810006 | 2021-10-13 13:50:00 | 2021-10-13 14:50:00 | 2021-10-13 15:50:00 | Coughid-21 | 1 |
| 14 | 800810010 | 800810007 | 2021-10-14 13:50:00 | 2021-10-14 14:50:00 | 2021-10-14 16:50:00 | Coughid-21 | 1 |
| 15 | 800810010 | 800810008 | 2021-10-15 13:50:00 | 2021-10-15 14:50:00 | 2021-10-15 17:50:00 | Coughid-21 | 1 |
| 16 | 800810010 | 800810009 | 2021-10-16 13:50:00 | 2021-10-16 14:50:00 | 2021-10-16 18:50:00 | Coughid-21 | 1 |
| 17 | 800810010 | 800810006 | 2021-10-17 13:50:00 | 2021-10-17 14:50:00 | 2021-10-17 15:50:00 | Coughid-21 | 1 |
| 18 | 800810010 | 800810007 | 2021-10-18 13:50:00 | 2021-10-18 14:50:00 | 2021-10-18 16:50:00 | Coughid-21 | 1 |
| 19 | 800810010 | 800810008 | 2021-10-19 13:50:00 | 2021-10-19 14:50:00 | 2021-10-19 17:50:00 | Coughid-21 | 1 |
| 20 | 800810010 | 800810009 | 2021-10-20 13:50:00 | 2021-10-20 14:50:00 | 2021-10-20 18:50:00 | Coughid-21 | 1 |

This test data set contains several hospitals with different efficiency to handle with the test reports. The expected result of the query should show that the Central Lukewarm Kingdom Hospital is the most efficient hospital to do viral tests.

The result of the SELECT statement (screenshot):

正在显示第 0 - 0 行 (共 1 行, 查询花费 0.0050 秒。)[avg_time: 3600.0000... - 3600.0000...]

```
SELECT hospital_name, AVG(UNIX_TIMESTAMP(report_time) - UNIX_TIMESTAMP(test_time)) AS avg_time FROM (test_list RIGHT OUTER JOIN doctor_list ON test_list.doctor_id=doctor_list.doctor_id) GROUP BY doctor_list.hospital_name ORDER BY avg_time ASC LIMIT 1
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

+ 选项

| hospital_name | avg_time |
|-----------------------------------|-----------|
| Central Lukewarm Kingdom Hospital | 3600.0000 |

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 test_taker_id FROM `test_list` WHERE collect_time>='2021-10-3
00:00:00' AND collect_time<='2021-10-5
00:00:00' GROUP BY test_taker_id HAVING ((UNIX_TIMESTAMP(MAX(collect_time))
- UNIX_TIMESTAMP(MIN(collect_time))) >=24*60*60) AND (COUNT(test_taker_id)
=2)
```


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

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

| Sim_id | Citizen_name |
|-----------|-----------------|
| 800810011 | Kurt Zouma |
| 800810012 | Aaron Cresswell |
| 800810013 | Vladimir Coufal |
| 800810014 | Issa Diop |
| 800810015 | Angelo Ogbonna |

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|------------------------|------------------------|------------------------|------------------|------------|
| 21 | 800810011 | 800810006 | 2021-10-03 19:00:00 | 2021-10-03 19:50:00 | 2021-10-03 20:50:00 | Coughid-21 | 1 |
| 22 | 800810012 | 800810006 | 2021-10-03 19:00:00 | 2021-10-03 19:50:00 | 2021-10-03 20:50:00 | Coughid-21 | 1 |
| 23 | 800810013 | 800810006 | 2021-10-03 19:00:00 | 2021-10-03 19:50:00 | 2021-10-03 20:50:00 | Coughid-21 | 1 |
| 24 | 800810014 | 800810006 | 2021-10-03 19:00:00 | 2021-10-03 19:50:00 | 2021-10-03 20:50:00 | Coughid-21 | 1 |
| 25 | 800810011 | 800810006 | 2021-10-04 20:00:00 | 2021-10-04 20:50:00 | 2021-10-04 21:50:00 | Coughid-21 | 1 |
| 26 | 800810012 | 800810006 | 2021-10-04 20:00:00 | 2021-10-04 20:50:00 | 2021-10-04 21:50:00 | Coughid-21 | 1 |
| 27 | 800810013 | 800810006 | 2021-10-04 13:00:00 | 2021-10-04 19:50:00 | 2021-10-04 20:50:00 | Coughid-21 | 1 |
| 28 | 800810014 | 800810006 | 2021-10-04 14:00:00 | 2021-10-04 19:50:00 | 2021-10-04 20:50:00 | Coughid-21 | 1 |
| 29 | 800810015 | 800810006 | 2021-10-04 14:00:00 | 2021-10-04 19:50:00 | 2021-10-04 20:50:00 | Coughid-21 | 1 |
| 30 | 800810015 | 800810006 | 2021-10-03 11:00:00 | 2021-10-03 19:50:00 | 2021-10-03 20:50:00 | Coughid-21 | 1 |
| 31 | 800810015 | 800810006 | 2021-10-04 15:00:00 | 2021-10-04 19:50:00 | 2021-10-04 20:50:00 | Coughid-21 | 1 |

| | | | | | | | |
|----|-----------|-----------|---------------------|---------------------|---------------------|------------|---|
| 39 | 233636 | 800810006 | 2021-10-09 17:00:00 | 2021-10-09 18:00:00 | 2021-10-09 19:00:00 | Coughid-21 | 0 |
| 40 | 800810020 | 800810006 | 2021-10-09 17:00:00 | 2021-10-09 18:00:00 | 2021-10-09 19:00:00 | Coughid-21 | 1 |

This test data set contains several citizens with different frequency of taking viral tests. Kurt Zouma (ID: 800810011) and Aaron Cresswell (ID: 800810012) took two tests from 2021-10-03 00:00 to 2021-10-05 00:00, and their two tests have a gap time of 25 hours. Vladimir Coufal (ID: 800810013) and Issa Diop (ID: 800810014) took two tests from 2021-10-03 00:00 to 2021-10-05 00:00, yet their tests don't satisfy the time gap (23 hours in between). Angelo Ogbonna (ID: 800810015) took three tests within the time window, and the time gap between first two tests is larger than 24 hours. Mark (ID 233636) took one test during the time window, as well as Alphonse Areola (ID 800810020). It is required that the phone numbers of all citizens who did two viral tests should be listed, and as my understanding, it means the predicate is based on the collect time. Because only the collect time depends on citizen's action. Considering that in Use Case 4, I happened to insert that Alex Kral (ID 800810010) took two tests with in the time window of Use Case 5. Therefore, according to the statement, only the phone numbers of Alex Kral, Kurt Zouma and Aaron Cresswell, which are 800810010, 800810011 and 800810012 should be shown after the SELECT statement.

The result of the SELECT statement (screenshot):

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0028 秒。)

```
SELECT test_taker_id FROM `test_list` WHERE collect_time>='2021-10-3 00:00:00' AND collect_time<='2021-10-5 00:00:00' GROUP BY test_taker_id HAVING ((UNIX_TIMESTAMP(MAX(collect_time)) - UNIX_TIMESTAMP(MIN(collect_time)))>=24*60*60) AND (COUNT(test_taker_id)=2)
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

☐ 显示全部 | 行数: 25 | 过滤行: 在表中搜索 | 按索引排序: 无

+ 选项

| | test_taker_id |
|-----------------------------------|---------------|
| <input type="checkbox"/> 编辑 复制 删除 | 800810010 |
| <input type="checkbox"/> 编辑 复制 删除 | 800810011 |
| <input type="checkbox"/> 编辑 复制 删除 | 800810012 |

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

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

Your SQL statement: (This statement assumes the current time is 2021-10-14 19:00:00)

SELECT DISTINCT district AS district_name, (

CASE

```

        WHEN(SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) -
UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-10-14 19:00:00' AND
enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)>0)THEN 'high'

        WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) -
UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-14 19:00:00'AND
enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)>0) AND (SUM(CASE WHEN is_healthy=0 AND
(UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-
10-14 19:00:00'AND enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)=0)THEN 'mid'

        WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) -
UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-10-14 19:00:00' AND
enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)=0) AND (SUM(CASE WHEN is_healthy=0 AND
(UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-
14 19:00:00'AND enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)=0)THEN 'low'

END)risk_level FROM ((test_list RIGHT OUTER JOIN travel_record_list ON
travel_record_list.traveller_sim_id=test_list.test_taker_id)NATURAL JOIN base_station_list) GROUP BY
district ORDER BY CASE

        WHEN(SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) -
UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-10-14 19:00:00' AND
enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)>0)THEN 1

        WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) -
UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-14 19:00:00'AND
enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)>0) AND (SUM(CASE WHEN is_healthy=0 AND
(UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-
10-14 19:00:00'AND enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)=0)THEN 2

        WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) -
UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-10-14 19:00:00' AND
enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)=0) AND (SUM(CASE WHEN is_healthy=0 AND
(UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-
14 19:00:00'AND enter_time>='2021-10-07 19:00:00'THEN 1 ELSE 0 END)=0)THEN 3

END;

```

The result of the SELECT statement (screenshot):

正在显示第 0 - 3 行 (共 4 行, 查询花费 0.0007 秒。)

```

SELECT DISTINCT district AS district_name, (CASE WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-10-14 19:00:00' AND enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)>0) THEN 'high' WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-14 19:00:00' AND enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)>0) AND (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) >=24*60*60 AND report_time<='2021-10-14 19:00:00' AND enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)=0) THEN 'mid' WHEN (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-14 19:00:00' AND enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)=0) AND (SUM(CASE WHEN is_healthy=0 AND (UNIX_TIMESTAMP(exit_time) - UNIX_TIMESTAMP(enter_time)) <24*60*60 AND report_time<='2021-10-14 19:00:00' AND enter_time>='2021-10-07 19:00:00' THEN 1 ELSE 0 END)=0) THEN 3

```

☐ 性能分析 [编辑] [刷新]

☐ 显示全部 | 行数: 25 | 过滤行: 在表中搜索

+ 选项

| district_name | risk_level |
|--------------------------|------------|
| Glow Sand district | high |
| Lenny town | high |
| Centre Lukewarm Hillside | mid |
| Raspberry town | 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 test_taker_id, citizen_name FROM ((test_list LEFT OUTER JOIN citizen_list ON
test_list.test_taker_id=citizen_list.sim_id) LEFT OUTER JOIN doctor_list ON
test_list.doctor_id=doctor_list.doctor_id ) LEFT OUTER JOIN hospital_list ON
doctor_list.hospital_name=hospital_list.hospital_name WHERE (district='Centre Lukewarm
Hillside')AND(is_healthy=0)AND(report_time>='2021-10-04 00:00:00')AND(report_time<='2021-10-04
23:59:59');
```

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

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

| Sim_id | Citizen_name |
|-----------|--------------|
| 800810016 | Ben Johnson |
| 800810017 | Craig Dawson |

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|---------------------|---------------------|---------------------|------------------|------------|
| 32 | 800810016 | 800810006 | 2021-10-04 20:00:00 | 2021-10-04 20:50:00 | 2021-10-04 21:50:00 | Coughid-21 | 0 |
| 33 | 800810017 | 800810007 | 2021-10-04 20:00:00 | 2021-10-04 20:50:00 | 2021-10-04 21:50:00 | Coughid-21 | 0 |

This test data set together with test data set in Use Case 3 and Use Case 4 contains several test records with different test results and different test takers in different hospitals of different districts. Ben Johnson (ID 800810016) was tested positive on 2021-10-04 in Centre Lukewarm Hillside. Hence according to the statement, only him and his phone number should be selected.

The result of the SELECT statement (screenshot):

正在显示第 0 - 0 行 (共 1 行, 查询花费 0.0030 秒。)

```
SELECT test_taker_id, citizen_name FROM ((test_list LEFT OUTER JOIN citizen_list ON test_list.test_taker_id=citizen_list.sim_id) LEFT OUTER JOIN doctor_list ON
test_list.doctor_id=doctor_list.doctor_id ) LEFT OUTER JOIN hospital_list ON doctor_list.hospital_name=hospital_list.hospital_name WHERE (district='Centre Lukewarm
Hillside')AND(is_healthy=0)AND(report_time>='2021-10-04 00:00:00')AND(report_time<='2021-10-04 23:59:59')
```

性能分析 [编辑内嵌] [编辑] [解析 SQL]

☐ 显示全部

行数: 25

过滤行: 在表中搜索

+ 选项

| test_taker_id | citizen_name |
|---------------|--------------|
| 800810016 | Ben Johnson |

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

Your SQL statement:

```

SELECT (
(
SELECT COUNT(test_taker_id) FROM (test_list NATURAL JOIN doctor_list) NATURAL JOIN hospital_list
WHERE(report_time>='2021-10-05 00:00:00')AND(report_time<='2021-10-05
23:59:59')AND(is_healthy=0)AND(district='Centre Lukewarm Hillside')
)
-
(
SELECT COUNT(test_taker_id) FROM (test_list NATURAL JOIN doctor_list) NATURAL JOIN hospital_list
WHERE(report_time>='2021-10-04 00:00:00')AND(report_time<='2021-10-04
23:59:59')AND(is_healthy=0)AND(district='Centre Lukewarm Hillside')
)
) AS increase

```

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

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

| Sim_id | Citizen_name |
|-----------|------------------|
| 800810018 | Ryan Fredericks |
| 800810019 | Lukasz Fabianski |

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|---------------------|---------------------|---------------------|------------------|------------|
| 34 | 800810018 | 800810006 | 2021-10-04 20:00:00 | 2021-10-04 20:50:00 | 2021-10-04 21:50:00 | Coughid-21 | 0 |
| 35 | 800810019 | 800810006 | 2021-10-05 20:00:00 | 2021-10-05 20:50:00 | 2021-10-05 21:50:00 | Coughid-21 | 0 |

This test data set together with test data set in Use Case 3, Use Case 4 and Use Case 5 contains several test records with different test results and different test takers in different hospitals of different districts on 2021-10-04 and 2021-10-05. Ben Johnson (ID 800810016) and Ryan Fredericks (ID 800810018) were tested positive on 2021-10-04 in Centre Lukewarm Hillside, and Lukasz Fabianski (ID 800810019) was tested positive on 2021-10-05 in Centre Lukewarm Hillside. Therefore, according to the statement, -1 should be shown after the SELECT statement.

The result of the SELECT statement (screenshots):

您的 SQL 语句已成功运行。

```
SELECT ( ( SELECT COUNT(test_taker_id) FROM (test_list NATURAL JOIN doctor_list) NATURAL JOIN hospital_list WHERE (report_time>='2021-10-05 00:00:00')AND (report_time<='2021-10-05 23:59:59')AND (is_healthy=0)AND (district='Centre Lukewarm Hillside') ) ) - ( SELECT COUNT(test_taker_id) FROM (test_list NATURAL JOIN doctor_list) NATURAL JOIN hospital_list WHERE (report_time>='2021-10-04 00:00:00')AND (report_time<='2021-10-04 23:59:59')AND (is_healthy=0)AND (district='Centre Lukewarm Hillside') ) ) AS increase
```

☐ 性能分析 [\[编辑内嵌\]](#) [\[编辑\]](#) [\[解析 SQL\]](#) [\[创建 PHP 代码\]](#) [\[刷新\]](#)

+ 选项

increase

-1

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(
(
SELECT COUNT( traveller_sim_id) FROM ((travel_record_list LEFT OUTER JOIN base_station_list ON
travel_record_list.base_station_coordinate=base_station_list.base_station_coordinate) LEFT OUTER JOIN
district_list ON base_station_list.district=district_list.district_name)
WHERE(((base_station_list.district='Lenny town')AND((exit_time>='2021-10-07
19:30:00')AND(enter_time<='2021-10-08 21:50:00'))
AND(traveller_sim_id<>233636))OR((base_station_list.district='Centre Lukewarm
Hill')AND((exit_time>='2021-10-08 21:50:00')AND(enter_time<='2021-10-09 19:30:00'))
AND(traveller_sim_id<>233636))
)
/
(
SELECT COUNT(test_taker_id) FROM (((((travel_record_list NATURAL JOIN base_station_list ) ) LEFT
OUTER JOIN district_list ON district=district_list.district_name) )LEFT OUTER JOIN test_list ON
traveller_sim_id=test_list.test_taker_id) WHERE ( ( (base_station_list.district='Lenny
town')AND(exit_time>='2021-10-07 19:30:00')AND(enter_time<='2021-10-08 21:50:00') )
OR( (base_station_list.district='Centre Lukewarm Hill')AND(exit_time>='2021-10-08
21:50:00')AND(enter_time<='2021-10-09 19:30:00') ) )AND (is_healthy=0)AND(report_time>='2021-10-09
19:00:00')AND(report_time<='2021-10-23 19:00:00')AND(traveller_sim_id<>233636)
)
) as spread_rate;
```

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

The following citizen information is added to the table test_list:

| Test_i d | Test_taker_ id | Doctor_id | Collect_ti me | Test_tim e | Report_ti me | Test_sample_ty pe | Is_healt hy |
|-------------|-------------------|-----------|------------------|---------------|-----------------|----------------------|----------------|
|-------------|-------------------|-----------|------------------|---------------|-----------------|----------------------|----------------|

| | | | | | | | |
|----|-----------|-----------|---------------------|---------------------|---------------------|------------|---|
| 36 | 800810000 | 800810006 | 2021-10-12 20:00:00 | 2021-10-12 20:50:00 | 2021-10-12 21:50:00 | Coughid-21 | 0 |
| 37 | 800810003 | 800810006 | 2021-10-25 20:00:00 | 2021-10-25 20:50:00 | 2021-10-25 21:50:00 | Coughid-21 | 0 |
| 38 | 800810005 | 800810006 | 2021-10-15 21:50:00 | 2021-11-05 21:50:00 | 2021-10-15 21:50:00 | Coughid-21 | 1 |

In Use Case 1, Michail Antonio (ID 800810000), Nikola Vlasic (ID 800810003) and Tomas Soucek (ID 800810005) were in the same district as the positive case with 48 hours. Michail Antonio finally was tested positive within 14 days, Nikola Vlasic was tested positive after 14 days, whereas Tomas Soucek was tested negative. According to the statement, the spread rate should be calculated by dividing the total number of people that were in the same district as the positive case with 48 hours, which is three (Michail Antonio (ID 800810000), Nikola Vlasic (ID 800810003) and Tomas Soucek (ID 800810005)), by the total number of people among them that later confirmed to be infected in 14 days, which is one (only Michail Antonio). The final result is three.

The result of the SELECT statement (screenshots):

```

您的 SQL 语句已成功运行。

select( ( SELECT COUNT( traveller_sim_id) FROM ((travel_record_list LEFT OUTER JOIN base_station_list ON travel_record_list.base_station_coordinate=base_station_list.base_station_coordinate) LEFT OUTER JOIN district_list ON base_station_list.district=district_list.district_name) WHERE ((base_station_list.district='Lenny town') AND ((exit_time>='2021-10-07 19:30:00') AND (enter_time<='2021-10-08 21:50:00')) AND (traveller_sim_id<>233636)) OR ((base_station_list.district='Centre Lukewarm Hill') AND ((exit_time>='2021-10-07 21:50:00') AND (enter_time<='2021-10-09 19:30:00')) AND (traveller_sim_id<>233636))) / ( SELECT COUNT( test_taker_id) FROM (((travel_record_list NATURAL JOIN base_station_list) ) LEFT OUTER JOIN district_list ON district=district_list.district_name) LEFT OUTER JOIN test_list ON traveller_sim_id=test_list.test_taker_id WHERE ( ( (base_station_list.district='Lenny town') AND (exit_time>='2021-10-07 19:30:00') AND (enter_time<='2021-10-08 21:50:00')) OR ( (base_stat[...]

[ 编辑 ]

+ 选项
spread_rate
3.0000

```

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: Doctor Jarrod Bowen (ID 800810006) was tested positive by himself on 2021-10-17 18:00:00. As usual, all people in the same district as him in the past 48 hours (before the positive report is published) need to take viral tests. However, by the hospital regulation, once tested positive, all his collected patients in the 48 hours based on his test time should take viral tests as well. Please write a query that contains all phone numbers of the patients checked by Dr Bowen in the past 48 hours.

Your SQL statement:

```

SELECT test_taker_id FROM test_list WHERE doctor_id=800810006 AND collect_time>='2021-10-15 18:00:00' AND collect_time<='2021-10-17 18:00';

```

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

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|---------------------|---------------------|---------------------|------------------|------------|
| 41 | 800810006 | 800810006 | 2021-10-17 17:00:00 | 2021-10-17 18:00:00 | 2021-10-17 19:00:00 | Coughid-21 | 0 |

This data means Doctor Jarrod Bowen (ID 800810006) was tested positive by himself on 2021-10-17 18:00:00, and all other data has been inserted in Use Case section.

The result of the SELECT statement (screenshot):

The screenshot shows a database query result interface. At the top, a green status bar indicates '正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0089 秒。)' (Showing rows 0 - 2 of 3, query cost 0.0089 seconds). Below this, the SQL query is displayed: `SELECT test_taker_id FROM test_list WHERE doctor_id=800810006 AND collect_time>='2021-10-15 18:00:00' AND collect_time<='2021-10-17 18:00:00';`. A toolbar below the query includes options like '性能分析', '编辑内嵌', '编辑', '解析 SQL', '创建 PHP 代码', and '刷新'. Below the toolbar, there are filters for '显示全部', '行数: 25', '过滤行: 在表中搜索', and '按索引排序: 无'. The main area shows a table with one row:

| test_taker_id |
|---------------|
| 800810006 |

. Each row has options for '编辑' (edit), '复制' (copy), and '删除' (delete).

Use case 2: In order to keep the public health system running as efficient as possible, and avoid over working of all doctors, the government requires that the doctors who has checked the most patients during a week should take a rest at home for at least two days. Please write a query that contains the name and hospital of the doctor who has collected the most patient samples during the week from 2021-10-11 to 2021-10-17.

Your SQL statement:

```
SELECT doctor_name ,COUNT(test_taker_id) AS num FROM test_list NATURAL JOIN
doctor_list WHERE collect_time>='2021-10-11
00:00:00'AND collect_time<='2021_10_17
23:59:59' GROUP BY doctor_id ORDER BY num DESC LIMIT 1;
```

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

All doctors' information has been inserted in the Use Case section, as well as all test records. With the data already in the database, only Jarrod Bowen should be selected out, and his number is 5.

The result of the SELECT statement (screenshot):

您的 SQL 语句已成功运行。

```
SELECT doctor_name ,COUNT(test_taker_id) AS num FROM test_list NATURAL JOIN doctor_list WHERE collect_time>='2021-10-11 00:00:00' AND collect_time<='2021-10-17 23:59:59' GROUP BY doctor_id ORDER BY num DESC LIMIT 1;
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

+ 选项

| doctor_name | num |
|---------------|-----|
| Jarrold Bowen | 5 |

Use case 3: The cost of viral test fee of all students from primary schools to universities is included in the medical insurance for students, which is in the end covered by the government. Assume people younger than 23 are all students, and it takes 50 Lukewarm dollars to take a test. Please write a query that calculates the total amount of money the government should spend on the tests of all students.

Your SQL statement:

```
SELECT COUNT(test_id)*50 AS total_amount_of_money FROM test_list INNER JOIN citizen_list ON test_list.test_taker_id=citizen_list.sim_id WHERE age<23;
```

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

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

| Sim_id | Citizen_name | Age |
|-----------|----------------------|-----|
| 800810021 | Kamarai Swyer | 19 |
| 800810022 | Freddie Potts | 18 |
| 800810023 | Jamal Baptiste | 18 |
| 800810024 | Harrison Ashby | 20 |
| 800810025 | Ajibola-Joshua Alese | 20 |
| 800810026 | Emmanuel Longelo | 20 |

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|---------------------|---------------------|---------------------|------------------|------------|
| 42 | 800810021 | 80081007 | 2021-10-23 17:00:00 | 2021-10-23 18:00:00 | 2021-10-23 19:00:00 | Coughid-21 | 1 |
| 43 | 800810022 | 80081007 | 2021-10-24 17:00:00 | 2021-10-24 18:00:00 | 2021-10-24 19:00:00 | Coughid-21 | 1 |
| 44 | 800810023 | 80081007 | 2021-10-25 17:00:00 | 2021-10-25 18:00:00 | 2021-10-25 19:00:00 | Coughid-21 | 1 |
| 45 | 800810024 | 80081007 | 2021-10-26 17:00:00 | 2021-10-26 18:00:00 | 2021-10-26 19:00:00 | Coughid-21 | 1 |
| 46 | 800810025 | 80081007 | 2021-10-27 17:00:00 | 2021-10-27 18:00:00 | 2021-10-27 19:00:00 | Coughid-21 | 1 |

| | | | | | | | |
|----|-----------|-----------|------------------------|------------------------|------------------------|------------|---|
| 47 | 800810026 | 800810007 | 2021-10-28 17:00:00 | 2021-10-28 18:00:00 | 2021-10-28 19:00:00 | Coughid-21 | 1 |
| 48 | 800810022 | 800810007 | 2021-10-29 17:00:00 | 2021-10-29 18:00:00 | 2021-10-29 19:00:00 | Coughid-21 | 1 |
| 49 | 800810023 | 800810007 | 2021-10-30 17:00:00 | 2021-10-30 18:00:00 | 2021-10-30 19:00:00 | Coughid-21 | 1 |
| 50 | 800810024 | 800810007 | 2021-10-31 17:00:00 | 2021-10-31 18:00:00 | 2021-10-31 19:00:00 | Coughid-21 | 1 |
| 51 | 800810025 | 800810007 | 2021-11-01 17:00:00 | 2021-11-01 18:00:00 | 2021-11-01 19:00:00 | Coughid-21 | 1 |
| 52 | 800810026 | 800810007 | 2021-11-02 17:00:00 | 2021-11-02 18:00:00 | 2021-11-02 19:00:00 | Coughid-21 | 1 |
| 53 | 800810021 | 800810007 | 2021-11-03 17:00:00 | 2021-11-03 18:00:00 | 2021-11-03 19:00:00 | Coughid-21 | 1 |
| 54 | 800810021 | 800810007 | 2021-11-04 17:00:00 | 2021-11-04 18:00:00 | 2021-11-04 19:00:00 | Coughid-21 | 1 |
| 55 | 800810022 | 800810007 | 2021-11-05 17:00:00 | 2021-11-05 18:00:00 | 2021-11-05 19:00:00 | Coughid-21 | 1 |
| 56 | 800810023 | 800810007 | 2021-11-06 17:00:00 | 2021-11-06 18:00:00 | 2021-11-06 19:00:00 | Coughid-21 | 1 |
| 57 | 800810024 | 800810007 | 2021-11-07 17:00:00 | 2021-11-07 18:00:00 | 2021-11-07 19:00:00 | Coughid-21 | 1 |
| 58 | 800810025 | 800810007 | 2021-11-08 17:00:00 | 2021-11-08 18:00:00 | 2021-11-08 19:00:00 | Coughid-21 | 1 |

All citizen added in this case are under 23, which can get 50 Lukewarm dollars. Some citizen added in the Use Case section are also under 23, such as Declan Rice and Ben Johnson. Hence the total number of test times of the students is 18, and the amount of money covered by the Lukewarm Kingdom government is $50 \times 18 = 900$ Lukewarm dollars

The result of the SELECT statement (screenshot):

您的 SQL 语句已成功运行。

```
SELECT COUNT(test_id)*50 AS total_amount_of_money FROM test_list INNER JOIN citizen_list ON test_list.test_taker_id=citizen_list.sim_id WHERE age<23;
```

☐ 性能分析 [\[编辑内嵌 \]](#) [\[编辑 \]](#) [\[解析 SQL \]](#) [\[创建 PHP 代码 \]](#) [\[刷新 \]](#)

+ 选项

| | |
|-----------------------|-----|
| total_amount_of_money | 900 |
|-----------------------|-----|

Use case 4: According to the latest research by one scientist in the Lukewarm Kingdom, the acquisition of Coughid-21 may have a bad impact on the infants. Hence, it is more dangerous for all potential pregnant and already pregnant women to have Coughid-21. The government decided to send text messages to all women aging from 25 to 35 to inform them to keep themselves safe and sound. write a query that contains all phone numbers of all women aging from 25 to 35.

Your SQL statement:

```
SELECT sim_id FROM `citizen_list` WHERE gender='Female' AND age>=25 AND age<= 35;
```

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

The following citizen information is added to the table citizen_list:

| Sim_id | Citizen_name | Gender | Age |
|-----------|----------------------|--------|-----|
| 800810027 | Mackenzie Arnold | Female | 24 |
| 800810028 | Zaneta Wyne | Female | 25 |
| 800810029 | Abbey-Leigh Stringer | Female | 28 |
| 800810030 | Gilly Flaherty | Female | 35 |
| 800810031 | Lisa Evans | Female | 36 |
| 800810032 | Claudia Walker | Female | 42 |

The data set contains the information of all female citizens aging from 24 to 42 from the current database. According to the statement, the phone numbers of Zaneta Wyne (ID 800810028), Zaneta Wyne (ID 800810029) and Zaneta Wyne (ID 800810030) should be selected.

The result of the SELECT statement (screenshot):

正在显示第 0 - 2 行 (共 3 行, 查询花费 0.0002 秒。)

```
SELECT sim_id FROM `citizen_list` WHERE gender='Female' AND age>=25 AND age<= 35;
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

☐ 显示全部 | 行数: 25 | 过滤行: 在表中搜索 | 按索引排序: 无

+ 选项

← T →

sim_id

☐ 编辑 复制 删除 800810028

☐ 编辑 复制 删除 800810029

☐ 编辑 复制 删除 800810030

Use case 5: When this pandemic emerged, no people paid much attention to it. They thought at most the spread of Coughid-21 was like that of a usual flu. However, as the situation got worse and worse, they soon realized that they were wrong. Whereas nobody took it serious in the beginning, so the vast majority of the public places only recorded people's name. It turned out that there are two people called

Aaron Cresswell. One is mentioned before in the Use Case4, whose phone number is 800810012. He entered Lenny town on 2021-09-01 12:00:00, and left for Glow Sand district on 2021-09-02 15:00:00. And then his entered Centre Luke warm Hill on 2021-09-03 14:00:00, and stayed there until 2021-09-05 17:00:00. The other Aaron entered Raspberry town on 2021-09-01 12:00:00, exited on 2021-09-03 18:00:00 and he went to Lenny town, and stayed there until 2021-09-05 14:00:00. Please write a query that shows all travelling record of them respectively.

Your SQL statement:

```
SELECT * FROM travel_record_list INNER JOIN citizen_list ON travel_record_list.traveller_sim_id=citizen_list.sim_id WHERE citizen_name='Aaron Cresswell' ORDER BY sim_id;
```

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

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

| Sim_id | Citizen_name |
|-----------|----------------|
| 800810033 | Aaron Creswell |

The following information is added to the table travel_record_list:

| Travel_id | Traveller_sim_id | Base_station_coordinate | Enter_time | Exit_time |
|-----------|------------------|-------------------------|---------------------|---------------------|
| 25 | 800810012 | 10101.1,2098.84 | 2021-09-01 12:00:00 | 2021-09-02 15:00:00 |
| 26 | 800810012 | 22304.4,7338 | 2021-09-02 15:00:00 | 2021-09-03 14:00:00 |
| 27 | 800810012 | 60304.4,7798.84 | 2021-09-03 14:00:00 | 2021-09-05 17:00:00 |
| 28 | 800810033 | 63615.5,8964.2 | 2021-09-01 12:00:00 | 2021-09-03 18:00:00 |
| 29 | 800810033 | 10101.1,2098.84 | 2021-09-03 18:00:00 | 2021-09-05 14:00:00 |

The result of the SELECT statement (screenshot):

正在显示第 0 - 4 行 (共 5 行, 查询花费 0.0003 秒。)

```
SELECT * FROM travel_record_list INNER JOIN citizen_list ON travel_record_list.traveller_sim_id=citizen_list.sim_id WHERE citizen_name='Aaron Cresswell' ORDER BY sim_id;
```

☐ 性能分析 [\[编辑内嵌 \]](#) [\[编辑 \]](#) [\[解析 SQL \]](#) [\[创建 PHP 代码 \]](#) [\[刷新 \]](#)

☐ 显示全部 | 行数: 25 | 过滤行: 在表中搜索 | 按索引排序: 无

| travel_id | traveller_sim_id | base_station_coordinate | enter_time | exit_time | sim_id | citizen_name | gender | age |
|-----------|------------------|-------------------------|---------------------|---------------------|-----------|-----------------|--------|-----|
| 25 | 800810012 | 10101.1,2098.84 | 2021-09-01 12:00:00 | 2021-09-02 15:00:00 | 800810012 | Aaron Cresswell | Male | 31 |
| 26 | 800810012 | 22304.4,7338 | 2021-09-02 15:00:00 | 2021-09-03 14:00:00 | 800810012 | Aaron Cresswell | Male | 31 |
| 27 | 800810012 | 60304.4,7798.84 | 2021-09-03 14:00:00 | 2021-09-05 17:00:00 | 800810012 | Aaron Cresswell | Male | 31 |
| 28 | 800810033 | 63615.5,8964.2 | 2021-09-01 12:00:00 | 2021-09-03 18:00:00 | 800810033 | Aaron Cresswell | Male | 35 |
| 29 | 800810033 | 10101.1,2098.84 | 2021-09-03 18:00:00 | 2021-09-05 14:00:00 | 800810033 | Aaron Cresswell | Male | 35 |

Use case 6: A scientific research shows that people above 65 years old are more vulnerable than others. Not only they have a weaker immune system, but they will also experience a longer recovery. Please write a query that filters out the phone numbers of these elder citizens.

Your SQL statement:

```
SELECT sim_id FROM citizen_list WHERE age>=65;
```

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

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

| Sim_id | Citizen_name | Age |
|-----------|-----------------|-----|
| 800810034 | Roger Waters | 65 |
| 800810035 | Freddie Mercury | 67 |
| 800810036 | Brian May | 78 |
| 800810037 | Roger Taylor | 89 |
| 800810038 | John Deacon | 76 |
| 800810039 | David Gilmour | 75 |

The data set is made up of the information of people over 65 years old. With the data set in the previous sections and cases together, only this set should be selected out.

The result of the SELECT statement (screenshot):

正在显示第 0 - 5 行 (共 6 行, 查询花费 0.0002 秒。)

```
SELECT sim_id FROM citizen_list WHERE age>=65;
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

☐ 显示全部

行数: 25

过滤行: 在表中搜索

按索引排序: 无

+ 选项

← T →

sim_id

☐

编辑

复制

删除

800810034

☐

编辑

复制

删除

800810035

☐

编辑

复制

删除

800810036

☐

编辑

复制

删除

800810037

☐

编辑

复制

删除

800810038

☐

编辑

复制

删除

800810039

Use case 7: Each base station has a chunk of chip for the storage and delivery of travel records, and every memory has a lifespan. However, too much travel records can deteriorate this. To prevent the potential

overflow of the travel records, the government decides to replace some most used chips with new ones. Please write a query that selects the most used base station and the number of travel records.

Your SQL statement:

```
SELECT base_station_coordinate, COUNT(travel_id) AS num FROM travel_record_list GROUP BY base_station_coordinate ORDER BY num DESC LIMIT 1;
```

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

All the information of the base stations has been inserted in the Use Case section, as well as all test records. With the data already in the database, only the base station with the coordinate 22304.4,7338 should be selected out.

The result of the SELECT statement (screenshot):



Use case 8: It is important to keep doctors, nurses and all other hospital workers safe and sound during this pandemic. However, there are some doctors unfortunately got infected by the Coughid-21. The government decides compensate those doctors 20,000 Lukewarm dollars each. Please write a query that calculates the total amount of money spent in this compensation.

Your SQL statement:

```
SELECT COUNT(test_taker_id)*20000 AS compensation FROM test_list INNER JOIN doctor_list ON test_taker_id=doctor_list.doctor_id WHERE is_healthy=0;
```

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

The following citizen information is added to the table test_list:

| Test_id | Test_taker_id | Doctor_id | Collect_time | Test_time | Report_time | Test_sample_type | Is_healthy |
|---------|---------------|-----------|---------------------|---------------------|---------------------|------------------|------------|
| 59 | 800810007 | 800810007 | 2021-10-11 13:50:00 | 2021-10-11 14:50:00 | 2021-10-11 15:50:00 | Coughid-21 | 0 |

The data set indicates that doctor Pable Fornals was once infected. Apart from him, with the current data, only one doctor, Jarrod Bowen, was once infected. Hence the total amount of money spent in this compensation is $20,000 \times 2 = 40,000$ Lukewarm dollars.

The result of the SELECT statement (screenshot):

您的 SQL 语句已成功运行。

```
SELECT COUNT(test_taker_id)*20000 AS compensation FROM test_list INNER JOIN doctor_list ON test_taker_id=doctor_list.doctor_id WHERE is_healthy=0;
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

+ 选项

| compensation |
|--------------|
| 40000 |

Use case 9: There is a famous radio programme called Morning Lukewarm. It focuses on daily news usually, whereas in the pandemic, it focuses on the outbreak of Coughid-21 as well. The speaker tells the number of cumulative confirmed cases in the programme every morning. Assume now is 2021-10-20 07:00:00, the speaker needs to know the number of cumulative confirmed cases. Please write a query that counts this number.

Your SQL statement:

```
SELECT COUNT(test_id) AS positive_cases FROM `test_list` WHERE is_healthy=0  
AND report_time<='2021-10-20 07:00:00';
```

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

With the current data, the total number of positive cases is 8, which should be selected and printed out.

The result of the SELECT statement (screenshot):

您的 SQL 语句已成功运行。

```
SELECT COUNT(test_id) AS positive_cases FROM `test_list` WHERE is_healthy=0 AND report_time<='2021-10-20 07:00:00';
```

☐ 性能分析 [编辑内嵌] [编辑] [解析 SQL] [创建 PHP 代码] [刷新]

+ 选项

| positive_cases |
|----------------|
| 8 |

Use case 10: The ratio of doctor is calculated by the number of all doctors dividing by the number of all citizens. Please write a query to calculate this ratio.

Your SQL statement:

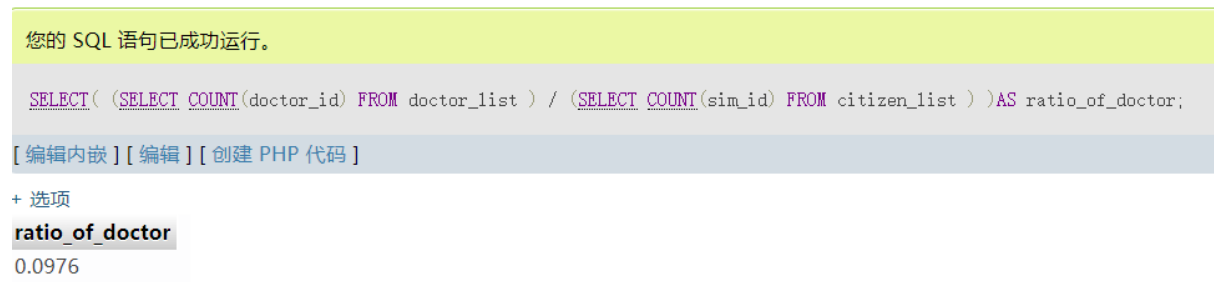
```
SELECT(  
(SELECT COUNT(doctor_id) FROM doctor_list )  
/  
(SELECT COUNT(sim_id) FROM citizen_list )
```

```
)AS ratio_of_doctor;
```

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

With the current data, the total number of doctors is 4, and that of citizens is 41. Hence the ratio is about 0.1, which should be selected and printed out.

The result of the SELECT statement (screenshot):



The screenshot shows a SQL execution interface. At the top, a green bar indicates the SQL statement was successfully executed. Below this, the SQL query is displayed: `SELECT((SELECT COUNT(doctor_id) FROM doctor_list) / (SELECT COUNT(sim_id) FROM citizen_list))AS ratio_of_doctor;`. Underneath the query, there are links for "[编辑内嵌]", "[编辑]", and "[创建 PHP 代码]". A "+ 选项" (Options) link is also present. The result is shown as a table with one column, **ratio_of_doctor**, and one row containing the value 0.0976.

```
您的 SQL 语句已成功运行。
```

```
SELECT( (SELECT COUNT(doctor_id) FROM doctor_list ) / (SELECT COUNT(sim_id) FROM citizen_list ) )AS ratio_of_doctor;
```

[编辑内嵌] [编辑] [创建 PHP 代码]

+ 选项

| ratio_of_doctor |
|-----------------|
| 0.0976 |