Coursework Report

Must Read

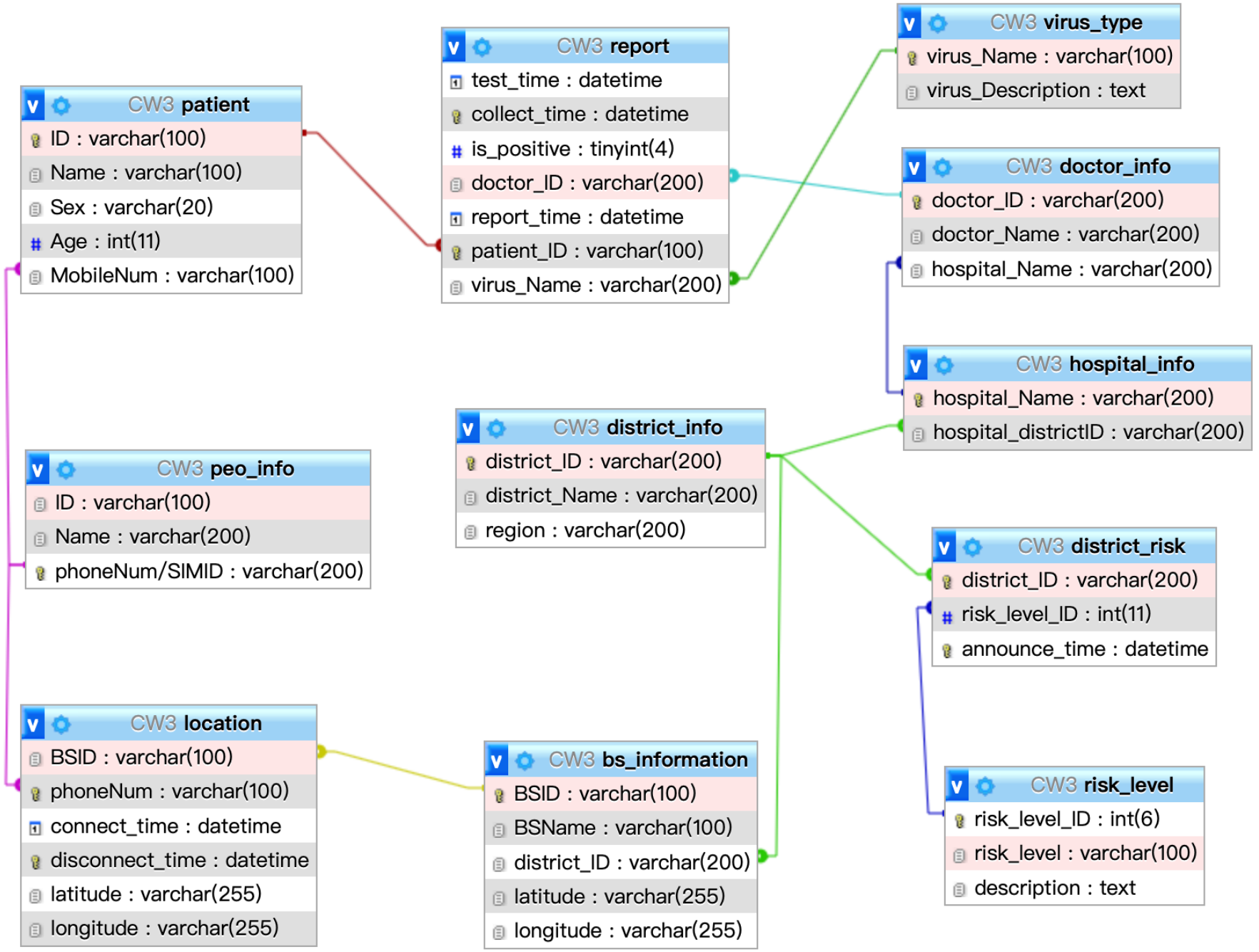
Your report and database design must be your own work. You should not copy any code from others or let anyone develop this PMMS. ***Plagiarism and collusion lead to a zero mark for this coursework***.

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

Name: Junru Jiang

Your ER Diagram here.

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



Database Design Details

Tables

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

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

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

Table name: Staff

Table design general explanation:

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

Attributes:

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

Foreign keys:

The column Branch_no references branch.branch_no, this is to make sure that branch_no are valid numbers that reflect existing branches of the company. It corresponds to the XXX relationship in the ER diagram.

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

Table name: patient

Table design explanation:

This patient table is used to store the information of all patients who came to do the viral tests. The primary key is ID as it is unique for every patient. The reason why I don't use the mobileNum as a primary key is that MobileNum may get changed but usually one person's ID will never change. And there are cases when some people discard one MobileNum and another one began to use it, then maybe some lagging information will match with the wrong person.

Column Definition	Domain	Explanation
ID varchar(100) primary key	All valid IDs are acceptable. For example, '32010719990819'.	The unique ID number of the citizens who did the viral test.
Name varchar(100)	All valid names are acceptable. For example, 'Monica'.	The name of patients
Sex varchar(20)	Male, female or other gender	The gender of patients. The value should be restricted to Male, Female and Others which can be checked using "check (sex IN('Male','Female', 'Others'))".
Age int(11)	integer from 0 - 150	The age of patients
MobileNum varchar(100)	Valid Mobile Number. For example, '8008101818'	The mobile number of patients. So that, if the patient is tested to be positive, we can immediately contact the person.

Foreign keys and reasons:

In the foreign key which is called 'MobileNum_fk', the column MobileNum references peo_info.phoneNum/SIMID, this is to make sure that the mobile numbers are valid that reflecting the people in the Lukewarm Kingdom. It corresponds to the relationship from peo_info to patient.

Table name: virus_type

Table design explanation

The virus_type table is used to store the information of all viruses that can be tested. The primary key is the virus_Name as usually it is unique for every virus.

Column Definition	Domain	Explanation
virus_Name varchar(255) primary key	All possible virus that can be tested from table virus_type, For example, 'Covid-19'.	Not null. The name of virus that tested.
virus_Description text	Some description about the virus.	Not null. The description that related to the virus characteristic.

Foreign keys and reasons:

No foreign keys.

Table name: doctor_info

Table design explanation

The doctor_info is used to store the information of doctors. The primary key is doctor's ID as it is unique for every doctor. But the doctor's name is likely to duplicate.

Column Definition	Domain	Explanation
doctor_ID varchar(200) primary key	All valid IDs from table doctor_info. For example, '12345'.	The doctor ID number which should be unique.
doctor_Name varchar(200)	All valid names are acceptable. For example, 'Richard'.	The name of doctors.
hospital_Name varchar(200)	All valid names are acceptable. For example, 'Centre Lukewarm Hillside Hospital'.	The hospital name which usually is unique from other hospitals.

Foreign keys and reasons:

In the foreign key which is called 'hospital_Name_fk', the column hospital_Name references hospital_info. hospital_Name, this is to make sure all doctors are from the hospitals in the Lukewarm Kingdom. It corresponds to the relationship from hospital_info to doctor_info.

Table name: hospital_info

Table design explanation

The hospital_info is used to store the information of hospital. The primary key is hospital's Name as it is unique for every hospital.

Column Definition	Domain	Explanation
hospital_Name varchar(200) primary key	All valid names are acceptable. For example, 'Centre Lukewarm Hillside Hospital'.	The hospital name which usually is unique from other hospitals.
hospital_districtID varchar(200)	Valid hospital district ID. For example, 'A123'	The location of the hospital. The hospital_districtID must have length of 4. The district ID must start with a letter to represent the region. Then follows with 3 numbers to represent the district. In this part, we assume all the hospitals are in the Lukewarm Kingdom. So, we don't need to specify the country of the hospital.

Foreign keys and reasons:

In the foreign key which is called ‘hospital_district_fk’, the column hospital_districtID references district_info. district_ID, this is to make sure all hospitals are located at valid districts in the Lukewarm Kingdom. It corresponds to the relationship from district_info to hospital_info.

Table name: peo_info

Table design explanation

The table peo_info is used to store the information of all citizens in Lukewarm Kingdom. The primary key is phoneNum/SIMID as it is unique for every person.

Column Definition	Domain	Explanation
ID varchar(100)	All valid IDs. For example, ‘123’.	Not Null
Name varchar(200)	All valid names. For example, ‘Chandler’	Not Null
phoneNum/SIMID varchar (200) primary Key	All valid phone numbers. For example, ‘567890’	Not Null this column is unique and not null because everyone has a unique mobile phone number

Foreign keys and reasons:

No foreign keys.

Table name:bs_information

Table design explanation

The table bs_information is used to store the information of basic station. The primary key is BSID as it is unique for every BS.

Column Definition	Domain	Explanation
BSID varchar(100) primary key	All valid IDs. For example, ‘1’	Not Null
BSName varchar(100)	All valid names with three capital letters. For example, ‘AAA’	Not Null The BSName need to be three capital letters. The first letter represents the order of BSs in this district, the second letter refers to the order of the districts and the last one represents the region.
district_ID varchar(200)	All valid district ID numbers with one capital letter and three digits after. For example, ‘A123’	Not Null the district_ID need to one capital letter to represent the region and the rest three digits to represent the order of districts in that region.
latitude varchar(255)	All valid GPS format are allowed. For example, ‘40°41'54.67’’N’	Not Null this is the latitude of BS location
longitude varchar(255)	All valid GPS format are allowed. For example, ‘74° 0'21.50’’W’	Not Null this is the longitude of the BS location

Foreign keys and reasons:

In the foreign key which is called ‘district_fk’, the column district_ID references district_info. district_ID, this is to make sure that the district_ID are valid that reflecting the districts of the BSs. It corresponds to the relationship from district_info to bs_information.

Table name: report

Table design explanation

This table report is used to store the information of all report. The combined primary keys are collect_time and patient_ID. Because one patient can have different reports at different collect_time. But one patient can only have one report at one time. Besides, the reason why I choose collect_time is patient can only know the collect_time before the report. If they want to query some information afterwards, there would be a duplicate mistake since database will check automatically.

Column Definition	Domain	Explanation
test_time datetime	All valid datetime type. For example, ‘2021-12-14 02:20:00’. We use 24-hour system	Not Null when you insert the data, check manually that the test_time is later than collect_time and earlier than report_time.
collect_time datetime primary key	All valid datetime type. For example, ‘2021-12-14 14:20:00’. We use 24-hour system	Not Null when you insert the data, check manually that the collect_time is earliest than any other time in this table.
is_positive tinyint(4)	1 or 0	Not Null We use 1 or 0 to represent the result. 1 refer to positive and 0 refer to negative.
doctor_ID varchar(200)	All valid IDs from table doctor_info. For example, ‘12345’	Not Null One doctor ID number of every report.
report_time datetime	All valid datetime type. For example, ‘2021-12-14 16:20:00’. We use 24-hour system	Not Null when you insert the data, check manually that the report_time is latest than any other time in this table.
patient_ID varchar(100) primary key	All valid IDs are acceptable from patient table. For example, ‘32010719990819’.	Not Null
virus_Name varchar(200)	All possible virus that can be tested from table virus_type, For example, ‘Covid-19’.	Not Null Usually every virus have unique name.

Foreign keys and reasons:

In the foreign key 1 which is called ‘doctor_ID_fk’, the column doctor_ID references doctor_info. doctor_ID, this is to make sure that the doctor_ID are valid that reflecting the existing doctors. It corresponds to the relationship from doctor_info to report.

In the foreign key 2 which is called ‘patient_ID_fk’, the column patient_ID references patient. ID, this is to make sure that the patient_ID are valid that reflecting the existing ID number of every patient. It corresponds to the relationship from patient to report.

In the foreign key 3 which is called ‘virus_name_fk’, the column virus_name references virus_type.virus_Name, this is to make sure that the virus_Name are valid that reflecting the existing viruses . It corresponds to the relationship from virus_type to report.

Table name:district_info

Table design explanation

This table is used to store the information of all district. The primary key is district_ID since although district name may get repeated but district ID number will never duplicated.

Column Definition	Domain	Explanation
district_ID varchar(200) primary key	All valid district ID numbers with one capital letter and three digits after. For example, ‘A123’	Not Null the district_ID need to one capital letter to represent the region and the rest three digits to represent the order of districts in that region.
district_Name varchar(200)	All valid district names are allowed. For example, ‘Centre Lukewarm Hillside’	Not Null The district names can be duplicated, but they are in different region.
region varchar(200)	All valid region names are allowed. For example, ‘NY’	Not Null Usually the region names need to be unique.

Foreign keys and reasons:

No foreign key.

Table name: district_risk

Table design explanation

The table district_risk is used to store the information of the risk level of districts. The primary key is district_ID as it is unique for every district.

Column Definition	Domain	Explanation
district_ID varchar(200) primary key	All valid district ID numbers with one capital letter and three digits after. For example, ‘A123’	Not Null the district_ID need to one capital letter to represent the region and the rest three digits to represent the order of districts in that region.
risk_level_ID int(11)	0 or 1 or 2	Default Null the database will check whether your input is 0 or 1or 2.
announce_time datetime primary key	All valid datetime type. For example, ‘2021-12-14 15:20:00’. We use 24-hour system	Not Null

Foreign keys and reasons:

In the foreign key 1 which is called ‘district_ID_fk’, the column district_ID references district_info. district_ID, this is to make sure that the district_ID are valid that reflecting the districts. It corresponds to the relationship from district_info to district_risk.

n the foreign key 2 which is called ‘risk_level_fk’, the column risk_level_ID references risk_level.risk_level_ID, this is to make sure that the risk_level_ID are valid that reflecting the existing risk levels. It corresponds to the relationship from risk_level to district_risk.

Table name:location

Table design explanation

The table location is used to store the location of every person. The combined primary key is phoneNum and disconnect_time. The reason why I choose the phoneNum as a part of primary key is everyone has a mobile number in the Lukewarm Kingdom. In addition, since the disconnect time will update with the current time. If someone haven’t left the BS, the disconnect time is the current time when we do query. Also, I have to mention that one phoneNum and one disconnect_time lead to a specific GPS location. But if we choose phoneNum and connect_time as combined primary key, there are the case that two people come into the BS at the same time but leave in the same GPS location with different disconnect_time, then that will cause the problem.

Column Definition	Domain	Explanation
BSID varchar(100)	All valid IDs. For example, ‘1’	Not Null
phoneNum varchar(100) primary key	All valid phone numbers. For example, ‘567890’	Not Null this column is unique and not null because everyone has a unique mobile phone number
connect_time datetime	All valid datetime type. For example, ‘2021-12-14 02:20:00’. We use 24-hour system	Not Null The connect_time refers to the time when a person came into the covered area of BS for the first time.
disconnect_time datetime primary key	All valid datetime type. For example, ‘2021-12-14 02:20:00’. We use 24-hour system	Not Null Default current_timestamp The disconnect_time refers to the time when a person appear in the covered area of BS for the last time.
latitude varchar(255)	All valid GPS format are allowed. For example, ‘40°41'54.67’’N’	Not Null this is the latitude of people’s real time location
longitude varchar(255)	All valid GPS format are allowed. For example, ‘74° 0'21.50’’W’	Not Null this is the longtitude of people’s real time location

Foreign keys and reasons:

In the foreign key 1 which is called ‘phoneNum_fk’, the column phoneNum references peo_info. phoneNum/SIMI, this is to make sure that the phoneNum are valid that reflecting the phoneNum of every person. It corresponds to the relationship from peo_info to location.

In the foreign key 2 which is called ‘BSID_fk’, the column BSID references bs_information. BSID, this is to make sure that the BSID are valid that reflecting the existing BSs. It corresponds to the relationship from bs_information to location.

Table name: risk_level

Table design explanation

The table is used to store the information of risk level. The primary key is risk_level_ID as it is unique for every risk level.

Column Definition	Domain	Explanation
risk_level_ID int(6) primary key	0 or 1 or 2	Not Null the database will check whether your input is 0 or 1or 2.
risk_level varchar(100)	high or mid or low	Not Null the database will check whether your input is valid or not.
description text	Some description about the risk level.	Not null. The description that how can a district be judged as high risk, mid-risk and low-risk.

Foreign keys and reasons:

No foreign key.

[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 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: ID, Name, Sex, Age, MobileNum, test_time, collect_time, is_positive, report_time, virus_Name,virus_Description, doctor_Name, doctor_ID, Hospital_Name, Hospital_district_ID

FDs (Indicate partial or transitive dependencies):

No FDs needed to be considered at this stage.

Normalised tables and which normal form they are currently in:

Now they are in the first normal form.

ID(primary key), Name, Sex, Age, MobileNum, test_time, collect_time (primary key), is_positive, report_time, virus_Name,virus_Description, doctor_Name, doctor_ID, Hospital_Name, Hospital_district_ID

Explanation: Therefore, one ID and one collect time will match one specific record.

Stage 2

Attributes: ID(primary key), Name, Sex, Age, MobileNum, test_time, collect_time (primary key), is_positive, report_time, virus_Name,virus_Description, doctor_Name, doctor_ID, Hospital_Name, Hospital_district_ID

FDs (Indicate partial or transitive dependencies):

{ID, collect_time}→{ Name, Sex, Age, MobileNum, test_time, is_positive, report_time, virus_Name,virus_Description, doctor_Name, doctor_ID, Hospital_Name, Hospital_district_ID}

{ID}→{ Name, Sex, Age, MobileNum}

Normalised tables and which normal form they are currently in:

Now they are in the second normal form.

ID(primary key), Name, Sex, Age, MobileNum

Patient_ID (primary key), collect_time (primary key), test_time, is_positive, report_time, virus_Name,virus_Description, doctor_Name, doctor_ID, Hospital_Name, Hospital_district_ID

Stage 3

Attributes: ID(primary key), Name, Sex, Age, MobileNum

Patient_ID (primary key), collect_time (primary key), test_time, is_positive, report_time, virus_Name,virus_Description, doctor_Name, doctor_ID, Hospital_Name, Hospital_district_ID

FDs (Indicate partial or transitive dependencies):

{Patient_ID, connect_time}→ { virus_Name } →{ virus_Description }

{Patient_ID, connect_time}→ { doctor_Name, doctor_ID } →{ Hospital_Name, Hospital_district_ID }

Normalised tables and which normal form they are currently in:

Now they are in the third normal form.

ID(primary key), Name, Sex, Age, MobileNum

Patient_ID (primary key), collect_time (primary key), test_time, is_positive, report_time, virus_Name,doctor_ID

virus_Name(primary key), virus_Description

doctor_ID(primary key), doctor_Name, Hospital_Name, Hospital_district_ID

Use Cases

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

Important Use Cases

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

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

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

Your SQL statement:

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

Test data and explanation:

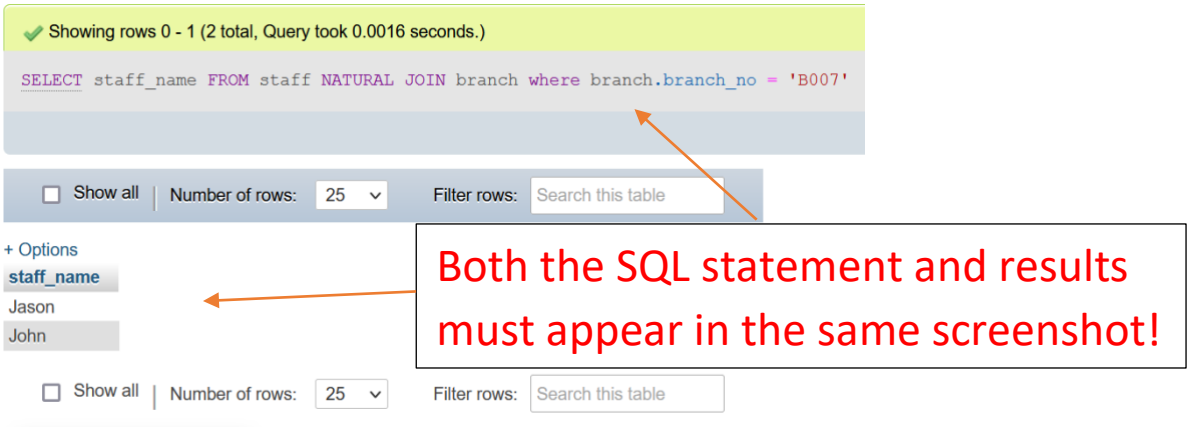
The following staff member information is added to the staff table (some attributes are hidden as they are not related to this task)

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

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

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

The result of the SELECT statement (screenshot):



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

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

Your SQL statement:

SELECT DISTINCT B.phoneNum FROM (location A INNER JOIN bs_information B1 USING(BSID)), (location B INNER JOIN bs_information B2 USING (BSID))

WHERE B1.district_ID = B2.district_ID AND A.phoneNum = '233636' AND B.phoneNum != '233636'

AND NOT (B.disconnect_time < A.connect_time)

AND NOT (B.connect_time > A.disconnect_time)

AND A.connect_time BETWEEN '2021-10-07 19:30:00' AND '2021-10-09 19:30:00';

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

This is the data added into table district_info.

district_ID	district_Name	region
A123	Centre Lukewarm Hillside	NY
A456	Lenny town	NY
B123	Glow Sand district	LA
C123	Raspberry town	KF
C456	Bunny Tail district	KF

This is the data added into table location.

(Some attributes are hidden as they are not related to this task)

BSID	phoneNum	connect_time	disconnect_time
1	13678566	2021-10-08 19:45:00	2021-10-09 14:20:00
1	233636	2021-10-06 19:20:00	2021-10-06 23:55:00
1	233636	2021-10-08 23:30:00	2021-10-09 19:30:00
6	233636	2021-10-08 14:30:00	2021-10-08 19:30:34
1	4576788	2021-10-06 22:20:00	2021-10-07 14:30:00
6	567890	2021-10-08 19:30:00	2021-10-09 21:13:00
1	777222	2021-10-01 19:30:00	2021-10-05 19:30:00
2	777222	2021-10-08 19:30:00	2021-10-09 16:30:00
1	777222	2021-10-10 19:30:00	2021-10-11 19:30:00

The corresponding phone numbers '233636' and some other phone numbers like '13678566' and '777222' have been inserted into location table. As you can see from the table location, the corresponding connect_time and disconnect_time have also put into the table.

The test data contains the BSID, phoneNum, connect_time and disconnect_time for each person. The test data contains people who were in the same district with Mark in 48 hours and people who were not. As shown, Mark who has the phone number '233636' went into the district A123 from 2021-10-06 19:20:00 to 2021-10-06 23:55:00. However, this shouldn't be selected because it is not the 48 hours before 2021-10-09 19:30:00. Therefore, the other two records of Mark should be filtered out from table location A. In comparison, the records of '13678566' and '567890' should be taken since they have time intersection with the positive case Mark in the same districts in 48 hours. In the same time, the second record of '777222' is also qualified. The other records are all unqualified. The first record of '777222' left the BS-1 before Mark came. The third record of '777222' went BS-1 after Mark left.

The expected result of the query should only contain the phone number '13678566' and '567890'.

The result of the SELECT statement (screenshot):

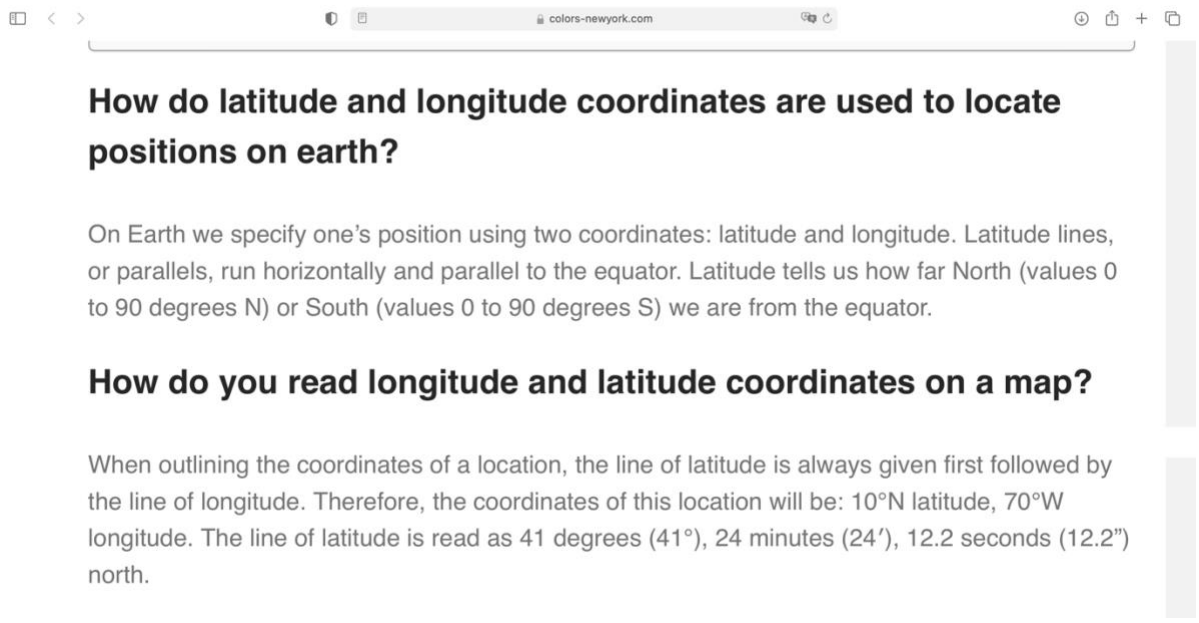
Use case 2: Please first clearly describe the format of GPS locations. The format must be a valid format that is used in real life. Then mimic what happens to your database when a user moves into the range of a base station and then moves out one hour later by listing all SQL statements involved in the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

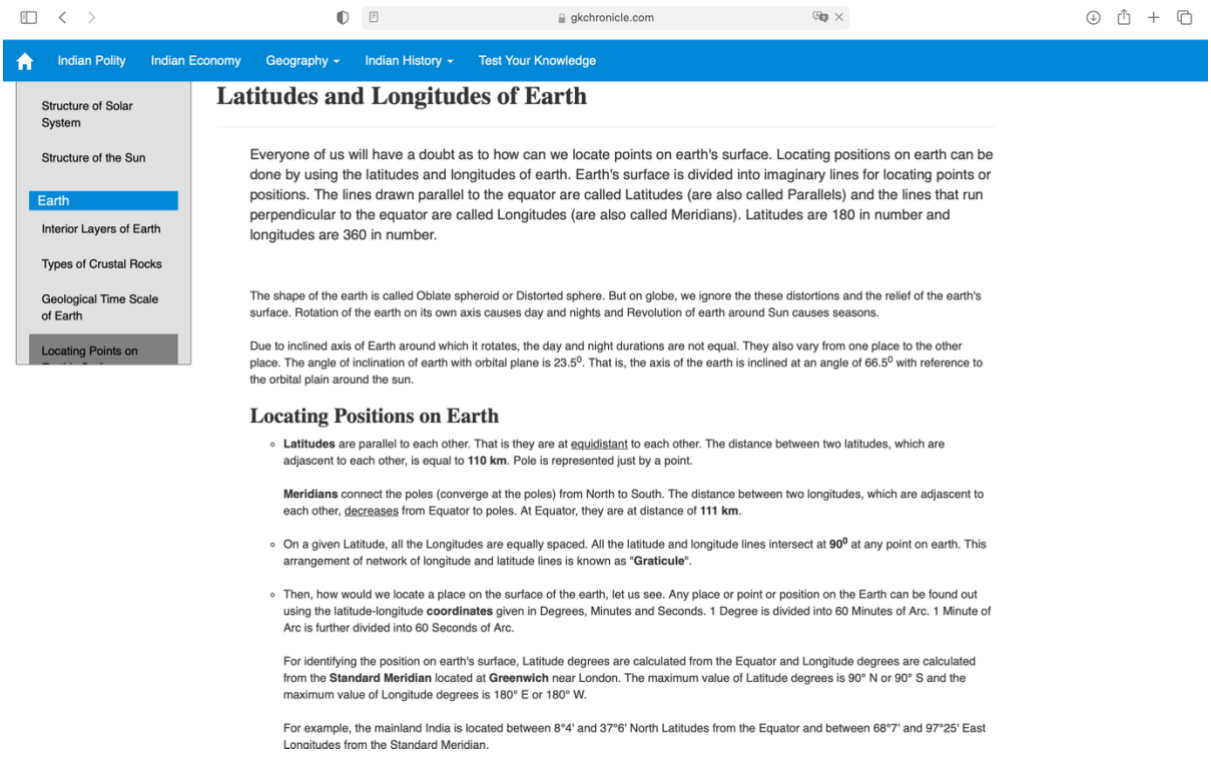
The GPS format and where did you learn it from (show the website link or the screenshot of the book):

As you can see from the screenshot below, when locating a certain position on the Earth, we can simply use longitude and latitude to verify the location of a person in one moment. The latitude and longitude both have three parts which are degrees, minutes, seconds.

For further information, the first material is available at: https://colors-newyork.com/how-do-latitude-and-longitude-coordinates-are-used-to-locate-positions-on-earth/ (Accessed: 03 December 2021)

The second material is available at: https://gkchronicle.com/world-geography/Locating-points-on-earths-surface.php (Accessed: 07 December 2021)





Your SQL statement(s) for travel record insertion:

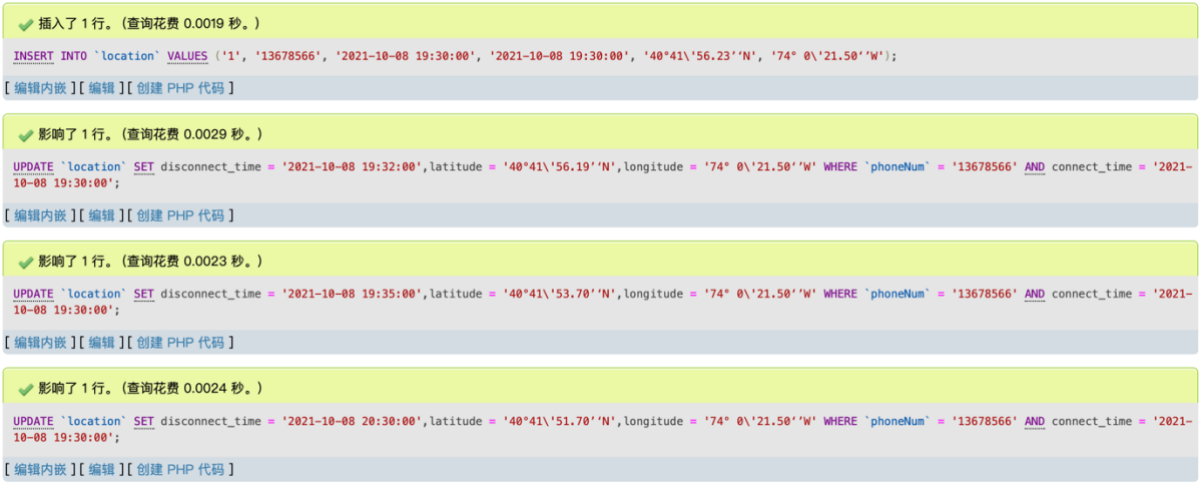
INSERT INTO `location` VALUES ('1', '13678566', '2021-10-08 19:30:00', '2021-10-08 19:30:00', '40°41\'56.23''N', '74° 0\'21.50''W');

UPDATE `location` SET disconnect_time = '2021-10-08 19:32:00',latitude = '40°41\'56.19''N',longitude = '74° 0\'21.50''W' WHERE `phoneNum` = '13678566' AND connect_time = '2021-10-08 19:30:00';

UPDATE `location` SET disconnect_time = '2021-10-08 19:35:00',latitude = '40°41\'53.70''N',longitude = '74° 0\'21.50''W' WHERE `phoneNum` = '13678566' AND connect_time = '2021-10-08 19:30:00';

UPDATE `location` SET disconnect_time = '2021-10-08 20:30:00',latitude = '40°41\'51.70''N',longitude = '74° 0\'21.50''W' WHERE `phoneNum` = '13678566' AND connect_time = '2021-10-08 19:30:00';

The result of the SELECT statements (screenshot):



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

MIN(generation_time),hospital_Name

FROM

(SELECT AVG(UNIX_TIMESTAMP(report_time) - UNIX_TIMESTAMP(test_time)) AS generation_time, hospital_Name FROM report INNER JOIN doctor_info USING (doctor_ID) GROUP BY hospital_name) AS `result`;

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

This is the information that inserted into table doctor_info.

doctor_ID	doctor_Name	hospital_Name
12345	Richard	London Centre Hospital
23456	Dick	California Centre Hospital
34567	Watson	London Centre Hospital

This is the information that inserted into table report.

(Some attributes are hidden as they are not related to this task).

test_time	collect_time	is_positive	doctor_ID	report_time	virus_Name

2021-10-03 18:09:00	2021-10-03 05:30:00	0	23456	2021-10-03 20:09:00	Covid-19
2021-10-04 12:20:00	2021-10-03 19:30:00	1	34567	2021-10-04 18:20:00	Covid-19
2021-10-09 15:34:00	2021-10-04 15:30:00	1	12345	2021-10-10 15:34:00	Covid-19
2021-10-14 14:30:00	2021-10-04 19:34:00	1	34567	2021-10-14 15:30:00	Covid-19
2021-10-04 22:20:00	2021-10-04 19:45:00	0	23456	2021-10-05 07:20:00	Covid-19

The test data contains the report_time and collect_time of two different hospitals. The expected result of the query is to figure out the minimum value of average report generation time. From the table doctor_info, you can calculate the average report time of California Centre Hospital is 19800s while the average report time of London Centre Hospital is 37200s. Obviously, the minimum value of average report generation time should be 19800s and California Centre Hospital has the fastest report generation time.

For this test to work, all existing data in report and doctor_info need to be deleted first.

The result of the SELECT statement (screenshot):

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

```
        MobileNum
FROM
        patient AS p
        INNER JOIN (
        SELECT
                *
        FROM
                report
        WHERE
                collect_time <= '2021-10-05 00:00:00' AND collect_time >= '2021-10-03 00:00:00'
        GROUP BY
                patient_ID
        HAVING
                COUNT( collect_time ) = '2'
                AND UNIX_TIMESTAMP(MAX( collect_time ))-
UNIX_TIMESTAMP(MIN( collect_time )) > 86400
        ) AS N ON p.ID = N.patient_ID;
```

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

This is the data I inserted into table patient about the information of patients who did the viral tests.

ID	Name	Sex	Age	MobileNum
123	Chandler	Male	31	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	31	1678438
688	Mark	Male	22	233636

This is the data we inserted into table report about the information of different reports.

(Some attributes are hidden as they are not related to this task).

test_time	collect_time	is_positive	doctor_ID	report_time	patient_ID
2021-10-03 18:09:00	2021-10-03 05:30:00	0	23456	2021-10-03 20:09:00	123
2021-10-03 19:30:00	2021-10-03 15:30:00	0	23456	2021-10-03 22:30:00	134

2021-10-04 12:20:00	2021-10-03 19:30:00	1	34567	2021-10-04 18:20:00	688
2021-10-04 22:20:00	2021-10-04 19:45:00	0	23456	2021-10-05 07:30:00	123
2021-10-14 14:30:00	2021-10-09 19:34:00	0	34567	2021-10-14 14:30:00	688
2021-10-03 22:30:00	2021-10-03 19:30:00	0	34567	2021-10-04 22:30:00	134

The test data set the reports of 4 patients in different collect_time.

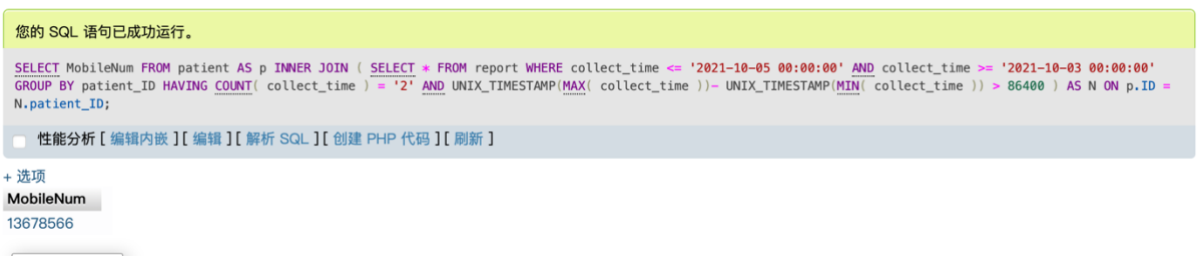
As we can see from the table report, when we first select two viral tests from 2021-10-03 00:00 to 2021-10-05 00:00, patient 123 and patient 134 did two viral tests, same as the result in the screenshot below.

But time internal of two viral tests on patient 134 was less than 24 hours. So, patient 134's records should be deleted from the result. In contrast, time internal of two viral tests on patient 123 was more than 24 hours. So, patient 123's records should be kept in the result.

Because there are only two viral tests which is selected before, so we can use MIN to represent the record that was done previously and MAX to represent the record the was done later.

For this test to work, all existing data in report and patient need to be deleted first.

The result of the SELECT statement (screenshot):



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

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

Your SQL statement:

SELECT

district_Name,

risk_level

FROM

(district_info INNER JOIN district_risk USING (district_ID))INNER JOIN risk_level USING (risk_level_ID)

WHERE announce_time = '2021-10-09 00:00:00'

ORDER BY risk_level_ID DESC, district_Name ASC;

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

This is the data I inserted into table district_info.

district_ID	district_Name	region
A123	Centre Lukewarm Hillside	NY
A456	Lenny town	NY
B123	Glow Sand district	LA
C123	Raspberry town	KF
C456	Bunny Tail district	KF

This is the information that I added to table district_risk.

district_ID	announce_time	risk_level_ID
A123	2021-10-09 00:00:00	2
A123	2021-10-09 19:30:00	1
A456	2021-10-09 00:00:00	2
B123	2021-10-09 00:00:00	1
C123	2021-10-09 00:00:00	0
C456	2021-10-09 00:00:00	0

This is the information that I put into the table risk_level.

risk_level	description	risk_level_ID
low	no positive cases within 1 week	0
mid	here are positive cases within 1 week	1
high	There are one or more positive case staying more than 24 hours.	2

It can be seen from the tables, there are three risk levels. There are basic district information and districts' risk levels on serval specific times. The districts should only be displayed on the ranking only if they are at same announce time. For example, in the sql statement, I want to list the district of different risk levels on 2021-10-09 00:00:00. Therefore, the record of A123 on 2021-10-09 19:30:00 should not be included.

The result of the SELECT statement (screenshot):

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`,
        MobileNum
FROM
        report
        INNER JOIN patient ON report.patient_ID = patient.ID
    INNER JOIN doctor_info USING ( doctor_ID )
    INNER JOIN hospital_info USING (hospital_Name)
    INNER JOIN district_info ON hospital_info.hospital_districtID = district_info.district_ID
WHERE
```

collect_Time BETWEEN '2021-10-04 00:00:00' AND '2021-10-04 23:59:59'

AND is_Positive = '1'

AND district_Name = 'Centre Lukewarm Hillside';

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

This is the information put in the table hospital_info.

hospital_Name	hospital_districtID
London Centre Hospital	A123
NY Centre Hospital	A123
California Centre Hospital	B123

This is the information put in the table patient.

ID	Name	Sex	Age	MobileNum
123	Chandler	Male	31	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	31	1678438
688	Mark	Male	22	233636

This is the data inserted into table report.

test_time	collect_time	is_positive	doctor_ID	report_time	patient_ID
2021-10-09 23:34:00	2021-10-04 15:30:00	0	34567	2021-10-10 15:30:00	145
2021-10-03 18:09:00	2021-10-03 05:30:00	0	23456	2021-10-03 20:09:00	145
2021-10-04 12:20:00	2021-10-03 19:30:00	1	34567	2021-10-04 18:20:00	123
2021-10-14 14:30:00	2021-10-04 19:34:00	1	34567	2021-10-14 14:30:00	688
2021-10-04 22:20:00	2021-10-04 19:45:00	0	23456	2021-10-05 07:30:00	134

This is the data inserted into table doctor_info.

doctor_ID	doctor_Name	hospital_Name
12345	Richard	NY Centre Hospital
23456	Dick	California Centre Hospital

34567	Watson	London Centre Hospital

This is the data inserted into table district_info.

district_ID	district_Name	region
A123	Centre Lukewarm Hillside	NY
A456	Lenny town	NY
B123	Glow Sand district	LA

The test data contains the information about all positive or negative cases in different districts at different time.

In order to fulfil the purpose, we separate the whole query into four parts. First, as the screenshot below posted, we need to put table report and patient together using ID for future checks on is_positive and the restraints on collect_time.

test_time	collect_time	is_positive	doctor_ID	report_time	patient_ID	virus_Name	ID	Name	Sex	Age	MobileNum
2021–10–03 18:09:00	2021–10–03 05:30:00	0	23456	2021–10–03 20:09:00	145	Covid–19	145	Ross	Male	31	777222
2021–10–04 12:20:00	2021–10–03 19:30:00	1	34567	2021–10–04 18:20:00	123	Covid–19	123	Chandler	Male	31	13678566
2021–10–09 23:34:00	2021–10–04 15:30:00	0	34567	2021–10–10 15:30:00	145	Covid–19	145	Ross	Male	31	777222
2021–10–14 14:30:00	2021–10–04 19:34:00	1	34567	2021–10–14 14:30:00	688	Covid–19	688	Mark	Male	22	233636
2021–10–04 22:20:00	2021–10–04 19:45:00	0	23456	2021–10–05 07:30:00	134	Covid–19	134	Moncia	Female	29	4576788

Furthermore, we connect the existing table with table doctor_info to get the information of hospital name using doctor_ID preparing for inquiring the hospital location.

doctor_ID	test_time	collect_time	is_positive	report_time	patient_ID	virus_Name	ID	Name	Sex	Age	MobileNum	doctor_Na
23456	2021–10–03 18:09:00	2021–10–03 05:30:00	0	2021–10–03 20:09:00	145	Covid–19	145	Ross	Male	31	777222	Dick
34567	2021–10–04 12:20:00	2021–10–03 19:30:00	1	2021–10–04 18:20:00	123	Covid–19	123	Chandler	Male	31	13678566	Watson
34567	2021–10–09 23:34:00	2021–10–04 15:30:00	0	2021–10–10 15:30:00	145	Covid–19	145	Ross	Male	31	777222	Watson
34567	2021–10–14 14:30:00	2021–10–04 19:34:00	1	2021–10–14 14:30:00	688	Covid–19	688	Mark	Male	22	233636	Watson
23456	2021–10–04 22:20:00	2021–10–04 19:45:00	0	2021–10–05 07:30:00	134	Covid–19	134	Moncia	Female	29	4576788	Dick

After that, we connect the existing table with the table hospital_info to get the information of hospital district ID number for further query on hospital location to be 'Centre Lukewarm Hillside'.

正在显示第 0 – 4 行 (共 5 行, 查询花费 0.0024 秒。)

```
SELECT * FROM report INNER JOIN patient ON report.patient_ID = patient.ID INNER JOIN doctor_info USING ( doctor_ID ) INNER JOIN hospital_info USING
(hospital_Name);
```

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

hospital_Name	doctor_ID	test_time	collect_time	is_positive	report_time	patient_ID	virus_Name	ID	Name	Sex	Age	Mobile
California Centre Hospital	23456	2021–10–03 18:09:00	2021–10–03 05:30:00	0	2021–10–03 20:09:00	145	Covid–19	145	Ross	Male	35	777222
London Centre Hospital	34567	2021–10–04 12:20:00	2021–10–03 19:30:00	1	2021–10–04 18:20:00	123	Covid–19	123	Chandler	Male	40	136785
London Centre Hospital	34567	2021–10–09 23:34:00	2021–10–04 15:30:00	0	2021–10–10 15:30:00	145	Covid–19	145	Ross	Male	35	777222
London Centre Hospital	34567	2021–10–14 14:30:00	2021–10–04 19:34:00	1	2021–10–14 14:30:00	688	Covid–19	688	Mark	Male	15	23363
California Centre Hospital	23456	2021–10–04 22:20:00	2021–10–04 19:45:00	0	2021–10–05 07:30:00	134	Covid–19	134	Moncia	Female	29	45767

0021 秒。)

```
N report.patient_ID = patient.ID INNER JOIN doctor_info USING ( doctor_ID ) INNER JOIN hospital_info USING
```

[创建 PHP 代码] [刷新]

过滤行: 在表中搜索 按索引排序: 无

	collect_time	is_positive	report_time	patient_ID	virus_Name	ID	Name	Sex	Age	MobileNum	doctor_Name	hospital_districtID
:09:00	2021–10–03 05:30:00	0	2021–10–03 20:09:00	145	Covid–19	145	Ross	Male	35	777222	Dick	B123
:20:00	2021–10–03 19:30:00	1	2021–10–04 18:20:00	123	Covid–19	123	Chandler	Male	40	13678566	Watson	A123
3:34:00	2021–10–04 15:30:00	0	2021–10–10 15:30:00	145	Covid–19	145	Ross	Male	35	777222	Watson	A123
:30:00	2021–10–04 19:34:00	1	2021–10–14 14:30:00	688	Covid–19	688	Mark	Male	15	233636	Watson	A123
2:20:00	2021–10–04 19:45:00	0	2021–10–05 07:30:00	134	Covid–19	134	Moncia	Female	29	4576788	Dick	B123

Finally, we use inner join to relate the existing table with the table district_info using district_ID, so that we can select the positive cases in the district “Centre Lukewarm Hillside” on 2021-10-04.

正在显示第 0 – 4 行 (共 5 行, 查询花费 0.0033 秒。)

```
SELECT * FROM report INNER JOIN patient ON report.patient_ID = patient.ID INNER JOIN doctor_info USING ( doctor_ID ) INNER JOIN hospital_info USING
(hospital_Name) INNER JOIN district_info ON hospital_info.hospital_districtID = district_info.district_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

+ 选项

hospital_Name	doctor_ID	test_time	collect_time	is_positive	report_time	patient_ID	virus_Name	ID	Name	Sex	Age	Mobile
California Centre Hospital	23456	2021–10–03 18:09:00	2021–10–03 05:30:00	0	2021–10–03 20:09:00	145	Covid–19	145	Ross	Male	35	777222
London Centre Hospital	34567	2021–10–04 12:20:00	2021–10–03 19:30:00	1	2021–10–04 18:20:00	123	Covid–19	123	Chandler	Male	40	136785
London Centre Hospital	34567	2021–10–09 23:34:00	2021–10–04 15:30:00	0	2021–10–10 15:30:00	145	Covid–19	145	Ross	Male	35	777222
London Centre Hospital	34567	2021–10–14 14:30:00	2021–10–04 19:34:00	1	2021–10–14 14:30:00	688	Covid–19	688	Mark	Male	15	23363
California Centre Hospital	23456	2021–10–04 22:20:00	2021–10–04 19:45:00	0	2021–10–05 07:30:00	134	Covid–19	134	Moncia	Female	29	45767

```
JOIN doctor_info USING ( doctor_ID ) INNER JOIN hospital_info USING
= district_info.district_ID;
```

无

report_time	patient_ID	virus_Name	ID	Name	Sex	Age	MobileNum	doctor_Name	hospital_districtID	district_ID	district_Name	region
2021-10-03 20:09:00	145	Covid-19	145	Ross	Male	35	777222	Dick	B123	B123	Glow Sand district	LA
2021-10-04 18:20:00	123	Covid-19	123	Chandler	Male	40	13678566	Watson	A123	A123	Centre Lukewarm Hillside	NY
2021-10-10 15:30:00	145	Covid-19	145	Ross	Male	35	777222	Watson	A123	A123	Centre Lukewarm Hillside	NY
2021-10-14 14:30:00	688	Covid-19	688	Mark	Male	15	233636	Watson	A123	A123	Centre Lukewarm Hillside	NY
2021-10-05 07:30:00	134	Covid-19	134	Moncia	Female	29	4576788	Dick	B123	B123	Glow Sand district	LA

For this test to work, all existing data in the patient, doctor_info, hospital_info, report and district_info need to deleted first.

The result of the SELECT statement (screenshot):

正在显示第 0 – 0 行 (共 1 行, 查询花费 0.0077 秒。)

```
SELECT `Name`, MobileNum FROM report INNER JOIN patient ON report.patient_ID = patient.ID INNER JOIN doctor_info USING ( doctor_ID ) INNER JOIN hospital_info
USING (hospital_Name) INNER JOIN district_info ON hospital_info.hospital_districtID = district_info.district_ID WHERE collect_Time BETWEEN '2021-10-04
00:00:00' AND '2021-10-04 23:59:59' AND is_Positive = '1' AND district_Name = 'Centre Lukewarm Hillside';
```

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

Name	MobileNum
Mark	233636

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(collect_Time)

FROM report INNER JOIN doctor_info USING (doctor_ID) INNER JOIN hospital_info USING (hospital_Name)

INNER JOIN district_info ON hospital_info.hospital_districtID = district_info.district_ID

WHERE district_Name = 'Centre Lukewarm Hillside' AND report.is_positive = '1' AND collect_Time BETWEEN '2021-10-05 00:00:00' AND '2021-10-05 23:59:59')

- (SELECT COUNT(collect_Time)

FROM report INNER JOIN doctor_info USING (doctor_ID) INNER JOIN hospital_info USING (hospital_Name)

INNER JOIN district_info ON hospital_info.hospital_districtID = district_info.district_ID

WHERE district_Name = 'Centre Lukewarm Hillside' AND report.is_positive = '1' AND collect_Time BETWEEN '2021-10-04 00:00:00' AND '2021-10-04 23:59:59')

AS 'Increment';

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

Both two cases are only different with the data in table report.

The data in table doctor_info is the same.

doctor_ID	doctor_Name	hospital_Name
12345	Richard	NY Centre Hospital
23456	Dick	California Centre Hospital
34567	Watson	London Centre Hospital

The data in table hospital_info is the same.

hospital_Name	hospital_districtID
London Centre Hospital	A123
NY Centre Hospital	A123
California Centre Hospital	B123

For case 1: When the increment is negative, this is the data we inserted into our table report.

(Some attributes are hidden as they are not related to this task).

test_time	collect_time	is_positive	doctor_ID	report_time
2021-10-03 18:09:00	2021-10-03 05:30:00	0	23456	2021-10-03 20:09:00
2021-10-04 12:20:00	2021-10-03 19:30:00	1	34567	2021-10-04 18:20:00
2021-10-09 23:34:00	2021-10-04 15:30:00	1	34567	2021-10-10 15:30:00
2021-10-14 14:30:00	2021-10-04 19:34:00	1	34567	2021-10-14 14:30:00
2021-10-04 22:20:00	2021-10-04 19:45:00	0	23456	2021-10-05 07:30:00
2021-10-06 13:48:00	2021-10-05 13:38:00	0	34567	2021-10-07 16:23:00
2021-10-05 13:23:00	2021-10-05 17:38:00	1	34567	2021-10-07 13:38:00

As you can see from case 1 report table, there was only one positive case on 2021-10-05 while there were 2 positive cases on 2021-10-04. Then the increment can be calculated as 1-2 = -1. The expected result is -1 and the final answer is -1 too.

For this test to work, all existing data in the patient, doctor_info, hospital_info, report need to deleted first.

The reason why we need to let the patient table to be updated, but I don' t list the detail information of patients here is that in case 7 we don't care about who actually get infected but care about the number. However, you still have to record the patient information to make the report complete in real life. In other words, although the accurate patient information is not related in the case 7, it should be inserted into report.

For case 2: When the increment is positive, this is the data we inserted into our table report.

(Some attributes are hidden as they are not related to this task).

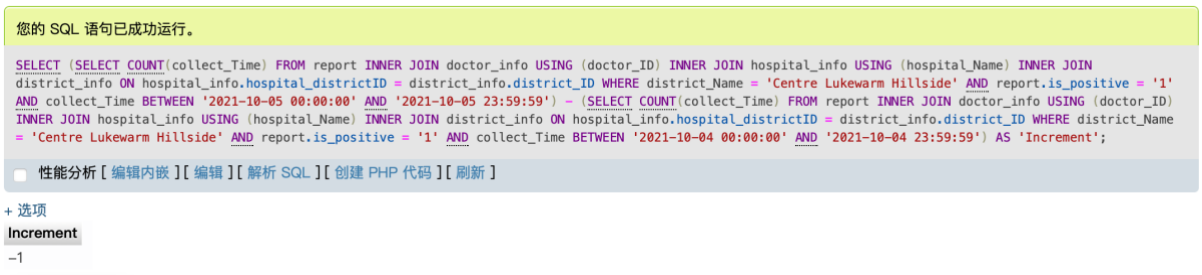
test_time	collect_time	is_positive	doctor_ID	report_time
2021-10-03 18:09:00	2021-10-03 05:30:00	0	23456	2021-10-03 20:09:00
2021-10-04 12:20:00	2021-10-03 19:30:00	1	34567	2021-10-04 18:20:00
2021-10-09 23:34:00	2021-10-04 15:30:00	1	34567	2021-10-10 15:30:00
2021-10-14 14:30:00	2021-10-04 19:34:00	0	34567	2021-10-14 14:30:00
2021-10-04 22:20:00	2021-10-04 19:45:00	0	23456	2021-10-05 07:30:00
2021-10-06 13:48:00	2021-10-05 13:38:00	0	34567	2021-10-07 16:23:00
2021-10-05 13:23:00	2021-10-05 17:38:00	1	34567	2021-10-07 13:38:00
2021-10-05 19:45:00	2021-10-05 23:45:00	1	34567	2021-10-06 19:45:00

As you can see from case 2 report table, there were two positive cases on 2021-10-05 while there was only one positive case on 2021-10-04. Then the increment can be calculated as 2-1 = 1. The expected result is 1 and the final answer is 1 too. For this test to work, all existing data in the patient, doctor_info, hospital_info and report need to deleted first. Although the accurate patient information is not related in the case 7, it should be inserted into report.

The result of the SELECT statement (screenshots):

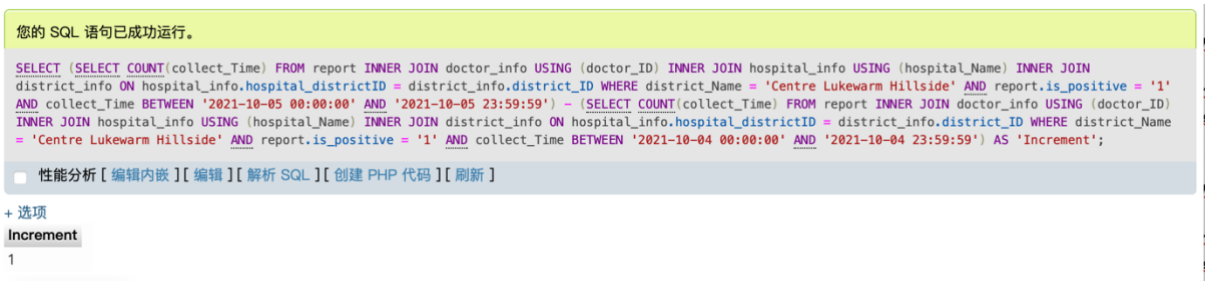
Case 1: Negative Increment

test_time	collect_time	is_positive	doctor_ID	report_time
2021-10-03	2021-10-03 05	0	23456	2021-10-03 2(
2021-10-04	2021-10-03 19:	1	34567	2021-10-04 18
2021-10-09	2021-10-04 15:	1	34567	2021-10-10 15
2021-10-14	2021-10-04 19:	1	34567	2021-10-14 14
2021-10-04	2021-10-04 19:	0	23456	2021-10-05 07
2021-10-06	2021-10-05 13:	0	34567	2021-10-07 16
2021-10-05	2021-10-05 17:	1	34567	2021-10-07 13



Case 2: Positive Increment

test_time	collect_time	is_positive	doctor_ID	report_time
2021-10-03	2021-10-03 05	0	23456	2021-10-03 2(
2021-10-04	2021-10-03 19:	1	34567	2021-10-04 18
2021-10-09	2021-10-04 15:	1	34567	2021-10-10 15
2021-10-14	2021-10-04 19:	0	34567	2021-10-14 14
2021-10-04	2021-10-04 19:	0	23456	2021-10-05 07
2021-10-06	2021-10-05 13:	0	34567	2021-10-07 16
2021-10-05	2021-10-05 17:	1	34567	2021-10-07 13
2021-10-05	2021-10-05 23	1	34567	2021-10-06 19



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 (telephone number is 233636) was tested to be positive at 19:30 on 09-Oct-2021 and he is the only person in the country that has coughid-19. Please write a query that calculates the spread rate of the virus.

Your SQL statement:

```
SELECT
   (SELECT COUNT(DISTINCT B.phoneNum) FROM (location A INNER JOIN bs_information B1
USING(BSID)), (location B INNER JOIN bs_information B2 USING (BSID))
WHERE B1.district_ID = B2.district_ID AND A.phoneNum = '233636' AND B.phoneNum != '233636'
AND NOT (B.disconnect_time < A.connect_time)
AND NOT (B.connect_time > A.disconnect_time)
AND A.connect_time BETWEEN '2021-10-07 19:30:00' AND '2021-10-09 19:30:00')
/
(SELECT
            COUNT(DISTINCT patient_ID) AS 'Infected People'
        FROM
            report
        WHERE
            is_positive = '1'
            AND collect_time BETWEEN '2021-10-09 19:30:00' AND '2021-10-23 19:30:00'
          AND patient_ID IN (
                    SELECT patient.ID FROM patient
       WHERE MobileNum IN (
         SELECT DISTINCT B.phoneNum FROM (location A INNER JOIN bs_information B1 USING(BSID)),
(location B INNER JOIN bs_information B2 USING (BSID))
WHERE B1.district_ID = B2.district_ID
   AND A.phoneNum = '233636' AND B.phoneNum != '233636'
        AND NOT (B.disconnect_time < A.connect_time)
        AND NOT (B.connect_time > A.disconnect_time)
        AND A.connect_time BETWEEN '2021-10-07 19:30:00' AND '2021-10-09 19:30:00'))) AS 'rate';
```

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

This is the information that I inserted into table report.

(Some attributes are hidden as they are not related to this task).

collect_time	is_positive	doctor_ID	patient_ID	virus_Name
2021-10-08 05:30:00	0	23456	123	Covid-19
2021-10-08 19:30:00	1	34567	145	Covid-19
2021-10-09 15:30:00	1	34567	156	Covid-19

2021-10-09 19:34:00	0	34567	688	Covid-19
2021-10-10 13:38:00	0	34567	156	Covid-19
2021-10-10 19:45:00	0	23456	123	Covid-19
2021-10-12 17:38:00	1	34567	156	Covid-19
2021-10-24 23:45:00	1	34567	145	Covid-19

This is the information that I inserted into table patient.

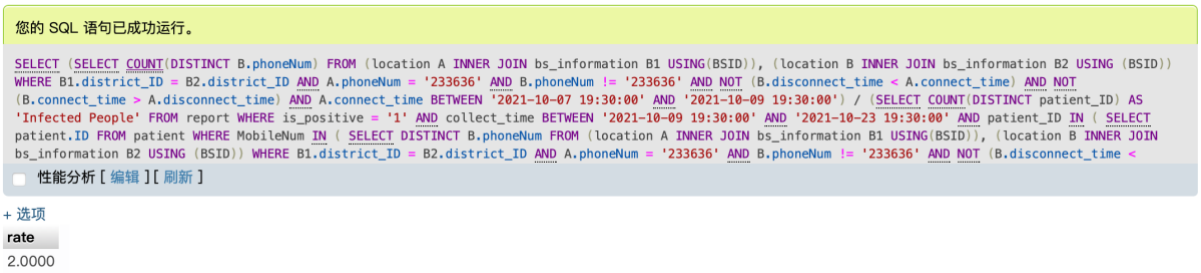
ID	Name	Sex	Age	MobileNum
123	Chandler	Male	31	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	31	777222
156	Rachel	Female	26	567890
555	Phoebe	Female	23	555556
688	Mark	Male	22	233636

This is the information that I inserted into table location which is same as user case 1.

BSID	phoneNum	connect_time	disconnect_time	latitude	longitude
1	13678566	2021-10-08 19:45:00	2021-10-09 14:20:00	40°41'56.67’‘N	74° 0'21.50’‘W
1	233636	2021-10-06 19:20:00	2021-10-06 23:55:00	40°41'58.67’‘N	74° 0'21.50’‘W
6	233636	2021-10-08 14:30:00	2021-10-08 19:30:34	67°41'.67’‘N	74° 0'21.50’‘W
1	233636	2021-10-08 23:30:00	2021-10-09 19:30:00	40°41'59.67’‘N	74° 0'21.50’‘W
1	4576788	2021-10-06 22:20:00	2021-10-07 14:30:00	40°46'56.67’‘N	74° 0'21.50’‘W
6	567890	2021-10-08 19:30:00	2021-10-09 21:13:00	68°41'56.67’‘N	74° 0'21.50’‘W
1	777222	2021-10-01 19:30:00	2021-10-05 19:30:00	39°41'56.67’‘N	74° 0'21.50’‘W
1	777222	2021-10-10 19:30:00	2021-10-11 19:30:00	41°41'56.67’‘N	74° 0'21.50’‘W

As shown in the table location and illustrated in the user case 1, there are two people who are at same districts with Mark in 48 hours. They are Chandler and Rachel. During the 14 days, as we can see from report table, Chandler whose patient ID number is 123 is negative. Rachel whose patient ID number is 156 was tested negative in 2021-10-10 but was positive in 2021-10-12. Therefore, only Rachel was caught the Covid-19 virus. The spread rate of the virus is expected to be 2/1 = 2. For this test to work, all existing data in patient, location and report should be deleted first.

The result of the SELECT statement (screenshots):



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: Owing to the epidemic background, the best way to avoid the infection is staying at home without hanging around. Therefore, the government would like to award those people who stay in one area of BSs for more than 24 hours. Please write a query that select the phone number of those people who meet the rewarding condition.

Your SQL statement:

SELECT DISTINCT phoneNum FROM location

WHERE UNIX_TIMESTAMP(disconnect_time)- UNIX_TIMESTAMP(connect_time) >= 86400;

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

This is the information that I inserted into the table location.

(Some attributes are hidden as they are not related to this task).

BSID	phoneNum	connect_time	disconnect_time
1	13678566	2021-10-08 19:45:00	2021-10-09 14:20:00
1	233636	2021-10-06 19:20:00	2021-10-06 23:55:00
6	233636	2021-10-08 14:30:00	2021-10-08 19:30:34
1	233636	2021-10-08 23:30:00	2021-10-09 19:30:00
1	4576788	2021-10-06 22:20:00	2021-10-07 14:30:00

6	567890	2021-10-08 19:30:00	2021-10-09 21:13:00
1	777222	2021-10-01 19:30:00	2021-10-05 19:30:00
1	777222	2021-10-10 19:30:00	2021-10-11 19:30:00

As the data shown in the table, 567890 spent more than 24 hours in the BS-1. 777222 spent exactly 1 hour in the BS-1. They should be selected as winners. For this test to work, all existing data in location should be deleted first.

The result of the SELECT statement (screenshot):

Use case 2: Due to the epidemic background, people in the high-risk districts are required to stay at home. However, citizens staying in the mid-risk and low-risk districts are allowed to go out. Therefore, government has set a QR-code for all citizens, people in the high-risk districts whose risk_level ID number is 2 will get a red code while people in the mid-risk and low-risk districts will get a green code based on the travel history yesterday. List all the citizens who have red code on 2021-10-09. Assume the risk level of one district announce at every day 00:00:00 P.M.

Your SQL statement:

SELECT DISTINCT phoneNum FROM location INNER JOIN bs_information USING (BSID) INNER JOIN district_risk USING (district_ID)

WHERE announce_time = '2021-10-09 00:00:00' AND district_risk.risk_level_ID = 2 AND connect_time BETWEEN '2021-10-08 00:00:00' AND '2021-10-08 23:59:59';

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

This is the data that we inserted into table district_risk.

district_ID	risk_level_ID	announce_time
C123	0	2021-10-09 00:00:00
C456	0	2021-10-09 00:00:00
A123	1	2021-10-10 00:00:00
B123	1	2021-10-09 00:00:00
A123	2	2021-10-09 00:00:00
A456	2	2021-10-09 00:00:00

This is the information that we inserted into table bs_information

BSID	BSName	district_ID	latitude	longitude
1	AAA	A123	40°41'54.67‘‘N	74° 0'21.50‘‘W
15	AAB	B123	34°03‘52.53‘‘N	118°15'23.50‘‘W
2	BAA	A123	40°42'54.67‘‘N	74° 0'23.50‘‘W
3	CAA	A123	40°42'51.67‘‘N	74° 0'22.50W
56	AAC	C123	67°15′43.34‘‘S	109°12′21.37‘‘E
6	ABA	A456	40°42'58.67‘‘N	74° 0'25.50‘‘W

This is the test data that I inserted into table location.

(Some attributes are hidden as they are not related to this task).

BSID	phoneNum	connect_time	disconnect_time
2	13678566	2021-10-08 19:45:00	2021-10-09 14:20:00
3	233636	2021-10-06 19:20:00	2021-10-06 23:55:00
6	233636	2021-10-08 14:30:00	2021-10-08 19:30:34
1	233636	2021-10-08 23:30:00	2021-10-09 19:30:00
1	4576788	2021-10-06 22:20:00	2021-10-07 14:30:00
15	567890	2021-10-08 19:30:00	2021-10-09 21:13:00
2	777222	2021-10-01 19:30:00	2021-10-05 19:30:00
6	777222	2021-10-10 19:30:00	2021-10-11 19:30:00

From the table district_risk, we can get that on 2021-10-09 00:00:00, A123 and A456 were high risk district. From the table bs_information, we can get that BS-1, BS-2, BS-3, BS-6 were all in high-risk level.

From the table location, we are able to get the information that there are four records between 2021-10-08 00:00:00 and 2021-10-08 23:59:59. 567890 went to BS-15 which is in the mid-level. So, this record should be filtered out. Besides, the phone number only need to appear once. Key word DISTINCT will filter out the repeated ones. Consequently, there should be two phone numbers left which are 13678566 and 233636.

For this test to work, all existing data in location, bs_information and district_risk should be deleted first.

The result of the SELECT statement (screenshot):

Use case 3: Based on the risk level announced in 2021-10-09 00:00:00, please find out the risk levels of the districts the citizen whose phone number is '233636' went on 2021-10-08.

Your SQL statement:

SELECT phoneNum, risk_level FROM location INNER JOIN bs_information USING (BSID) INNER JOIN district_risk USING (district_ID) INNER JOIN risk_level USING(risk_level_ID)

WHERE phoneNum = '233636' AND connect_time BETWEEN '2021-10-08 00:00:00' AND '2021-10-08 23:59:59'AND announce_time = '2021-10-09 00:00:00'

ORDER BY risk_level_ID DESC;

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

This is the test data that I inserted into table risk_level.

(Some attributes are hidden as they are not related to this task).

risk_level_ID	risk_level
0	low
1	mid
2	high

This is the data that we inserted into table district_risk.

district_ID	risk_level_ID	announce_time
C123	0	2021-10-09 00:00:00
C456	0	2021-10-09 00:00:00
A123	1	2021-10-10 00:00:00
B123	1	2021-10-09 00:00:00
A123	2	2021-10-09 00:00:00
A456	2	2021-10-09 00:00:00

This is the information that we inserted into table bs_information

BSID	BSName	district_ID	latitude	longitude
1	AAA	A123	40°41'54.67‘‘N	74° 0'21.50‘‘W
15	AAB	B123	34°03‘52.53‘‘N	118°15'23.50‘‘W
2	BAA	A123	40°42'54.67‘‘N	74° 0'23.50‘‘W
3	CAA	A123	40°42'51.67‘‘N	74° 0'22.50W
56	AAC	C123	67°15′43.34‘‘S	109°12′21.37‘‘E
6	ABA	A456	40°42'58.67‘‘N	74° 0'25.50‘‘W

This is the test data that I inserted into table location.

(Some attributes are hidden as they are not related to this task).

BSID	phoneNum	connect_time	disconnect_time
2	13678566	2021-10-08 19:45:00	2021-10-09 14:20:00
3	233636	2021-10-06 19:20:00	2021-10-06 23:55:00
6	233636	2021-10-08 14:30:00	2021-10-08 19:30:34
15	233636	2021-10-08 23:30:00	2021-10-09 19:30:00
1	4576788	2021-10-06 22:20:00	2021-10-07 14:30:00
15	567890	2021-10-08 19:30:00	2021-10-09 21:13:00

From the table location, we can get the information that phone number 233636 have three travelling records. Two of them is in 2021-10-08. He or she went to BS-6 and BS-15. From table bs_information, BS-6 is in district A456, and BS-15 is in B123. Based on the data announced in 2021-10-09 00:00:00, A456 was in high risk level and B123 was in mid risk level. So, the result should be high and mid.

For this test to work, all existing data in location, bs_information, district_risk and risk_level should be deleted first.

The result of the SELECT statement (screenshot):

Use case 4: After the outbreak of pandemic, lots of medical staff have made great efforts to save patients' lives. Now the government would like to award the doctor who have collected the most viral tests so far. List the information of the winner.

Your SQL statement:

SELECT doctor_Name,doctor_ID, COUNT(doctor_ID) AS 'times'

FROM doctor_info LEFT OUTER JOIN report USING (doctor_ID)

GROUP BY doctor_ID

HAVING COUNT(doctor_ID) =(

SELECT MAX(temp.amount)

FROM(

SELECT COUNT(doctor_ID) amount

FROM report

GROUP BY doctor_ID)temp);

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

This is the data that I inserted into table doctor_info.

doctor_ID	doctor_Name	hospital_Name
12345	Richard	NY Centre Hospital
23456	Dick	California Centre Hospital
34567	Watson	London Centre Hospital

From the table doctor_info, we are able to know the whole information about the doctor through his or her doctor ID number.

This is the information that I inserted into table report.

(Some attributes are hidden as they are not related to this task).

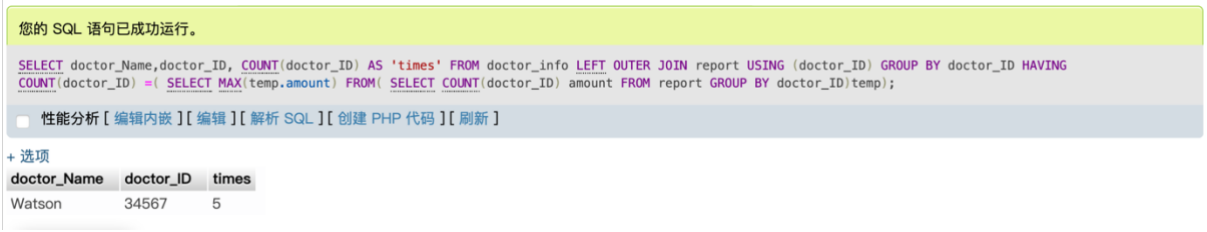
collect_time	is_positive	doctor_ID	patient_ID	virus_Name
2021-10-08 05:30:00	1	23456	123	Covid-19
2021-10-08 19:30:00	1	34567	145	Covid-19
2021-10-09 15:30:00	0	12345	156	Covid-19
2021-10-09 19:34:00	0	34567	688	Covid-19
2021-10-09 23:45:00	0	34567	337	Covid-19
2021-10-10 13:38:00	1	12345	156	Covid-19
2021-10-10 19:21:00	1	23456	156	Covid-19

2021-10-10 19:45:00	1	23456	555	Covid-19
2021-10-12 17:38:00	1	34567	134	Covid-19
2021-10-12 23:45:00	1	12345	688	Covid-19
2021-10-24 23:45:00	1	34567	337	Covid-19

It can be calculated from the previous table report, doctor 12345 collected 3 viral tests. Doctor 23456 collected 3 viral tests. Doctor 34567 collected 5 viral tests. So, doctor 34567 did the most viral tests.

For this test to work, all existing data in report and doctor_info should be deleted first.

The result of the SELECT statement (screenshot):



Use case 5: To make a deep understanding of the situation on different virus. Government now hires you to list all the virus name which has at least one positive cases.

Your SQL statement:

SELECT virus_Name FROM patient INNER JOIN report ON patient.ID = report.patient_ID

WHERE report.is_positive = 1

GROUP BY virus_Name

HAVING COUNT(DISTINCT patient_ID) >= 1;

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

This is the information about patients in the table patient.

ID	Name	Sex	Age	MobileNum
123	Chandler	Male	40	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	35	777222
156	Rachel	Female	26	567890
337	Joey	Male	40	222222
555	Phoebe	Female	23	555556

688	Mark	Male	15	233636

This the information I inserted into table report.

(Some attributes are hidden as they are not related to this task).

collect_time	is_positive	patient_ID	virus_Name
2021-10-08 05:30:00	1	123	Coughid-21
2021-10-08 19:30:00	1	145	Covid-19
2021-10-09 15:30:00	0	156	MERS
2021-10-09 19:34:00	0	688	SARS
2021-10-09 23:45:00	0	337	Coughid-21
2021-10-10 13:38:00	1	156	Covid-19
2021-10-10 19:21:00	1	156	MERS
2021-10-10 19:45:00	1	555	Coughid-21
2021-10-12 17:38:00	1	134	Covid-19
2021-10-12 23:45:00	1	688	Coughid-21
2021-10-24 23:45:00	1	337	Coughid-21

It is clear that there are five viral tests on Coughid-21. Four of them are positive cases. There are three positive cases on Covid-19. Two MERS viral test and, one is positive. One SARS test but it was negative.

So Coughid-21, Covid-19 and MERS should be kept.

For this test to work, all existing data in report and patient should be deleted first.

The result of the SELECT statement (screenshot):

Use case 6: To better fight against the pandemic, researchers would like to explore the influence of virus on different gender. Assume the rate is the male positive cases dividing by the total positive cases so far. Please write a query that calculates the rate of the virus on male.

Your SQL statement:

SELECT (SELECT COUNT(DISTINCT patient_ID)

FROM report INNER JOIN patient ON report.patient_ID = patient.ID

WHERE report.is_positive = '1'

AND patient.Sex = 'Male'

)/

(SELECT COUNT(DISTINCT patient_ID)

FROM report

WHERE report.is_positive = '1') AS 'res';

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

This is the information about patients in the table patient.

ID	Name	Sex	Age	MobileNum
123	Chandler	Male	40	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	35	777222
156	Rachel	Female	26	567890
337	Joey	Male	40	222222
555	Phoebe	Female	23	555556
688	Mark	Male	15	233636

It is clear from the table patient that among 7 patients, four of them are male while others are female. No other gender is included.

This the information I inserted into table report.

(Some attributes are hidden as they are not related to this task).

collect_time	is_positive	patient_ID	virus_Name
2021-10-08 05:30:00	1	123	Covid-19
2021-10-08 19:30:00	1	145	Covid-19
2021-10-09 15:30:00	0	156	Covid-19
2021-10-09 19:34:00	0	688	Covid-19
2021-10-10 13:38:00	0	156	Covid-19
2021-10-10 19:45:00	1	555	Covid-19
2021-10-12 17:38:00	1	134	Covid-19
2021-10-24 23:45:00	1	337	Covid-19

As table report shown, 5 patients were infected, and 2 patients were not. Among the positive cases, three of the positive cases were male. So, the expected rate should be 3/5 = 0.6 and the final answer is also 0.6000. For this test to work, all existing data in patient and report should be deleted first.

The result of the SELECT statement (screenshot):

Use case 7: To better fight against the pandemic, researchers would like to explore the influence of virus on different demographical groups. Assume the rate is the positive cases who are at their 20s dividing by the total positive cases so far. Please write a query that calculates the rate of the virus on people in their 20s.

Your SQL statement:

```
SELECT (SELECT COUNT(DISTINCT patient_ID)
        FROM report INNER JOIN patient ON report.patient_ID = patient.ID
        WHERE report.is_positive = '1'
        AND patient.Age >= 20
        AND patient.Age < 30
)/
(SELECT COUNT(DISTINCT patient_ID)
        FROM report INNER JOIN patient ON report.patient_ID = patient.ID
        WHERE report.is_positive = '1') AS 'res';
```

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

This is the information about patients in the table patient.

ID	Name	Sex	Age	MobileNum
123	Chandler	Male	40	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	35	777222
156	Rachel	Female	26	567890
337	Joey	Male	40	222222
555	Phoebe	Female	23	555556
688	Mark	Male	15	233636

It is clear from the table patient that among 7 patients, three of them are at their 20s while others are not.

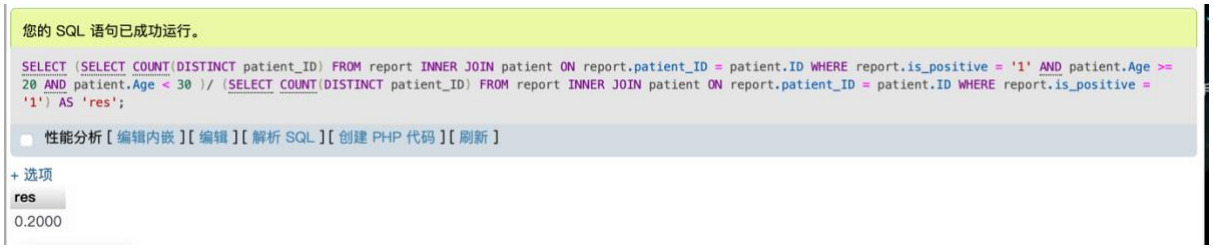
This the information I inserted into table report.

(Some attributes are hidden as they are not related to this task).

collect_time	is_positive	patient_ID	virus_Name
2021-10-08 05:30:00	1	123	Covid-19
2021-10-08 19:30:00	1	145	Covid-19
2021-10-09 15:30:00	0	156	Covid-19
2021-10-09 19:34:00	1	688	Covid-19
2021-10-10 13:38:00	0	156	Covid-19
2021-10-10 19:45:00	0	555	Covid-19
2021-10-12 17:38:00	1	134	Covid-19
2021-10-24 23:45:00	1	337	Covid-19

As table report shown, 5 patients were infected, and 2 patients were not. Among the positive cases, only one case which is 134 Monica were positive and at her 20s. So, the expected rate should be 1/5 = 0.2 and the final answer is also 0.2000. For this test to work, all existing data in patient and report should be deleted first.

The result of the SELECT statement (screenshot):



Use case 8: One couple would like to travel to Lukewarm Kingdom a few days later. They want to stay in the hotel of Centre Lukewarm Hillside. But they are afraid of the virus. Please list the risk level of Centre Lukewarm Hillside and announce time accordingly.

Your SQL statement:

SELECT risk_level, announce_time FROM district_risk INNER JOIN risk_level USING(risk_level_ID) INNER JOIN district_info USING (district_ID)

WHERE district_info.district_Name = 'Centre Lukewarm Hillside';

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

This is the data added into table district_info.

district_ID	district_Name	region

A123	Centre Lukewarm Hillside	NY
A456	Lenny town	NY
B123	Glow Sand district	LA
C123	Raspberry town	KF
C456	Bunny Tail district	KF

This is the test data that I inserted into table risk_level.

(Some attributes are hidden as they are not related to this task).

risk_level_ID	risk_level
0	low
1	mid
2	high

This is the test data that I inserted into table district_risk.

district_ID	risk_level_ID	announce_time
A123	0	2021-10-05 00:00:00
A123	0	2021-10-06 00:00:00
C123	0	2021-10-09 00:00:00
C456	0	2021-10-09 00:00:00
A123	1	2021-10-07 00:00:00
A123	1	2021-10-08 00:00:00
A123	1	2021-10-09 00:00:00
B123	1	2021-10-09 00:00:00
A123	2	2021-10-10 00:00:00
A123	2	2021-10-11 00:00:00
A456	2	2021-10-09 00:00:00

From the table district_info, we know that Centre Lukewarm Hillside's ID number is A123. Then we can get from table district_risk that there are 7 records about Centre Lukewarm Hillside. Connect the table district_risk with the table risk_level, we can know the risk_level. For this test to work, all existing data in risk_level, district_info and district_risk should be deleted first.

The result of the SELECT statement (screenshot):

Use case 9: Some BSs in region NY have countered some issues this week. The workers would like to check the problem one by one. So, they need a list of all information of BSs in region NY. Please write a query to list the details.

Your SQL statement:

SELECT BSID, district_info.district_Name,district_info.region FROM bs_information INNER JOIN district_info USING (district_ID)

WHERE region = 'NY';

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

This is the data added into table district_info.

district_ID	district_Name	region
A123	Centre Lukewarm Hillside	NY
A456	Lenny town	NY
B123	Glow Sand district	LA
C123	Raspberry town	KF
C456	Bunny Tail district	KF

This is the information that we inserted into table bs_information.

(Some attributes are hidden as they are not related to this task).

BSID	BSName	district_ID
1	AAA	A123
15	AAB	B123
2	BAA	A123
3	CAA	A123
56	AAC	C123
6	ABA	A456

As you can see from table district_info, district A123 and A456 are in region NY. From table bs_information, we can get that BS-1, BS-2, BS-3 are from district A123 and BS-6 are from district A456.

For this test to work, all existing data in district_info and bs_infomation should be deleted first.

The result of the SELECT statement (screenshot):

Use case 10: Government would like to know which kind of virus cause the panic of the public by viewing the total number of citizens who did the viral tests from 2021-10-08 to 2021-10-12. Please write a query that list the result from highest to lowest.

Your SQL statement:

```
SELECT virus_Name ,COUNT(DISTINCT patient_ID) AS 'times'
                FROM report
    WHERE collect_time BETWEEN '2021-10-08 00:00:00' AND '2021-10-12 23:59:59'
    GROUP BY virus_Name
    ORDER BY COUNT(DISTINCT patient_ID) DESC;
```

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

This is the data we inserted into table patient.

ID	Name	Sex	Age	MobileNum
123	Chandler	Male	40	13678566
134	Moncia	Female	29	4576788
145	Ross	Male	35	777222
156	Rachel	Female	26	567890
337	Joey	Male	40	222222
555	Phoebe	Female	23	555556
688	Mark	Male	15	233636

This is the table we inserted into table report.

(Some attributes are hidden as they are not related to this task).

collect_time	is_positive	patient_ID	virus_Name

2021-10-08 05:30:00	1	123	Coughid-21
2021-10-08 19:30:00	1	145	Covid-19
2021-10-09 15:30:00	0	156	MERS
2021-10-09 19:34:00	0	688	SARS
2021-10-09 23:45:00	0	337	Coughid-21
2021-10-10 13:38:00	1	156	Covid-19
2021-10-10 19:21:00	1	555	MERS
2021-10-10 19:45:00	1	555	Coughid-21
2021-10-12 17:38:00	1	134	Covid-19
2021-10-12 23:45:00	0	688	Coughid-21
2021-10-24 23:45:00	1	337	Coughid-21

It is clear that there were 10 viral tests from 2021-10-08 to 2021-10-12. Four of them are Coughid-21 and three of them are Covid-19. Two of them are MERS and the left one is SARS. So the order of occurence should be Coughid-21, Covid-19, MERS, SARS. For this test to work, all existing data in patient and report should be deleted first.

The result of the SELECT statement (screenshot):