CPT103 Coursework Report

Must Read

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

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

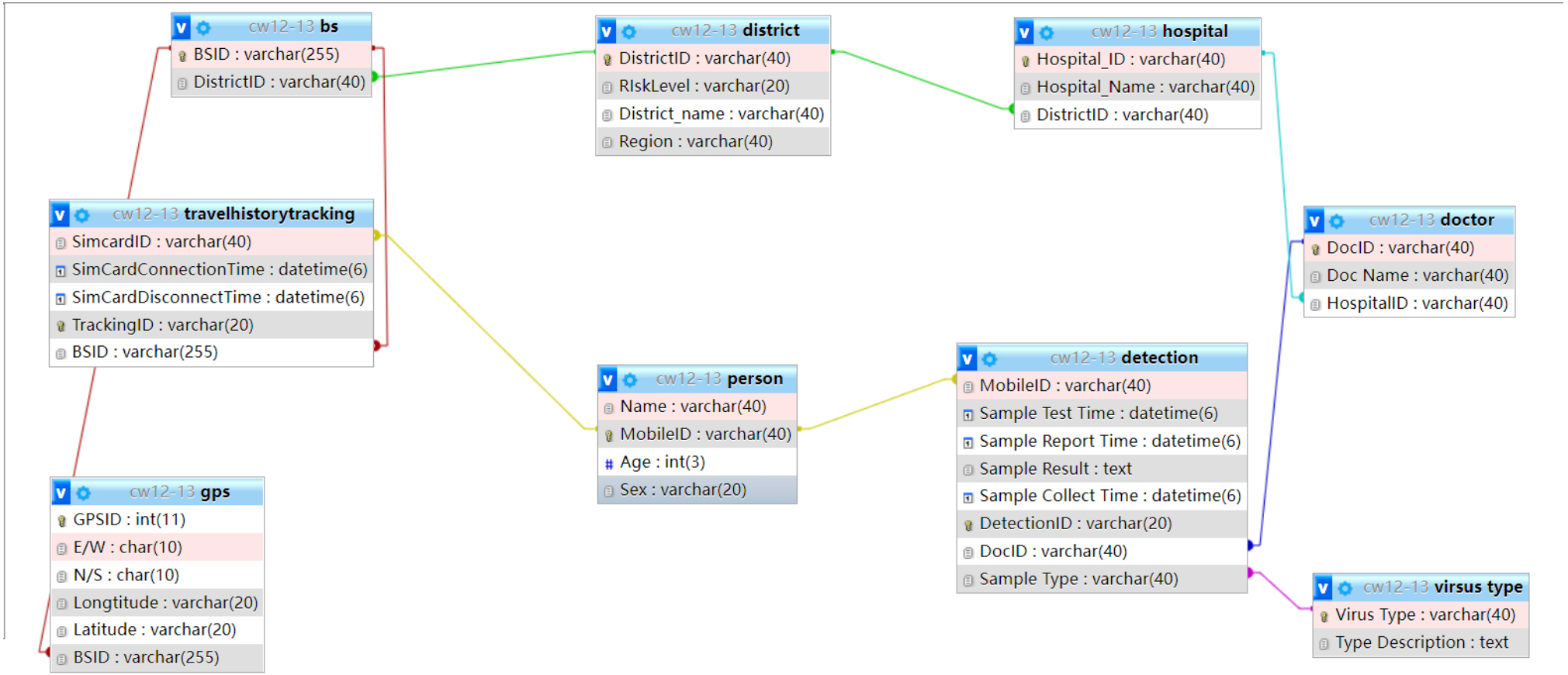
Your ID:2036501

Name: SHUCHEN_YUAN

Email Address: SHUCHEN.YUAN20@xjtlu.edu.cn

Your ER Diagram here.

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



Database Design Details

Tables

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

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

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

Table name: Staff

Table design general explanation:

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

Attributes:

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

Foreign keys:

The column Branch_no 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: district

Table design explanation

The district table is used to store the data of the district of the Lukewarm Kingdom. The primary key is the districtID because every district has a unique districtID in real life and the district have district ID, district name, corresponding risk level and the region it belongs to. Thus, we set those as the attribute of this table. And no more data needs to be referenced in this case.

Column Definition	Domain	Explanation
districtID varchar (40)	Any valid form of the combination of the letters and numbers is acceptable such as '213 ' or'C999'	The ID of a district. This ID can be flexible when it comes to different country or area thus, the type is varchar and the length is 40 in case it meets different situation. And it is the primary key of this table thus it is unique
District_name varchar (40)	All valid names are acceptable. For example: LA/ Los Angeles	The name of the district and different district can have the same name like what's in real-life situation. In addition, the length can't be so long corresponding the real-life situations so we set it 40
RIsklevel Varchar (20)	High, mid, low	As defined, the risk is high, mid, low, we set it Varchar (20) because it might be a special risklevel for the area that there is none virus such as 'safe', which couldn't be so long.
Region Varchar (40)	Central, East, West, North, South	The region must be one of the Central, East, West, North, South

Foreign keys and reasons:

No foreign keys

Table name: person

Table design explanation

The district table is used to store the data of the person of the Lukewarm Kingdom who already have done the virus test. We assume that all the person in the Lukewarm Kingdom have done the virus test just like in the real life everyone comes to China will provide a negative report and all the citizen have done to test so they are allowed to go to different place. The primary key is the MobileID because every person has a phone with a unique phone number in real life and a person has a phone number which is the mobile ID in this case, name, sex, and age. Thus, we set those as the attribute of this table. And no more data needs to be referenced in this case.

Column Definition	Domain	Explanation
MobileID Varchar (40)	Any valid form of MobileID is acceptable such as 1-800-273-8255 or +86 15199998888, which must have character representing	The MobileID can be flexible such as 1-800-273-8255 or +86 1519998888 which must contain the number that represent it region or district (some character may be inserted) thus we use the Varchar to handle possible situation and 40 length of the data is enough for almost all the cases we might meet in daily life. And everyone has a phone

	the place that the sim card belongs to.	corresponding to a mobileID thus it is unique and we set it as a primary key.
Name Varchar (40)	Any valid form of name is acceptable. For example 'Mark' or 'zhang san'	The name of a person and it can be duplicate. Every person has a name.
Age Int (3)	From 000 to 999	The age of the person, and to handle possible solution like age of 100+ we use length of 3. Every person has a age.
Sex Varchar (20)	Any valid form of Sex is acceptable such as male, female, None-binary gender.	Every person has a sex and it can be flexible nowadays, the sex can be complicate and not too long thus we use varchar (20) to handle it.

Foreign keys and reasons:

No foreign key.

Table name: GPS

Table design explanation

The district table is used to store the data of the GPS information of the bs station. As the question requires, we assume that the GPSID reflects the location of the base station and with the detect range of the base station we can imply the rough location pf a detected signal (In real life, 5G base station detection rang is 100m-300m). The primary key is the GPSID because every GPS information has a unique serial number ordered by the time (if the data is earliest collected, the serial number would be 1) in real life and the GPS must conclude the hemisphere information which represent the eastern, western, northern and the southern hemisphere by `E`, `W`, `N' and 'S'. In addition, every GPS information belongs to a base station (corresponds to a BSID). Thus, we set those as the attribute of this table and set the BSID as a foreign key.

Column Definition	Domain	Explanation
GPSID Int (11)	From 00000000000 to 99999999999	The serial number of the GPS data, it is corresponding to the time that the data was collected. For example, the first collected data's GPSID is 1. It is the primary key of table GPS.
E/W Char(10)	E and W	It represents the location of the GPS. The E means the Eastern hemisphere and W means the western hemisphere. It is not UNIQUE.
N/S Char (10)	N and S	It represents the location of the GPS. The N means the northern hemisphere and W means the southern hemisphere. It is not UNIQUE.
Longitude Varchar (20)	From 00000.0000 to 18000.0000	The longitude of the GPS. The format of the 'longitude' is 'dddmm.mmmm', 'ddd' represents the degree and the mm.mmmm represents the minutes. With the help of the E/W, it can tell you the precise longitude of the GPS. It can be the same in different cases.
Latitude Varchar (20)	From 0000.0000 to 9000.0000	The latitude of the GPS. The format of the 'latitude' is 'ddmm.mmmm', 'dd' represents the degree and the mm.mmmm represents the minutes.

		With the help of the N/S, it can tell you the precise latitude of the GPS. It can be the same in different cases.
BSID Varchar (255)	Any valid type of combination of letters and numbers is acceptable. For example, '4046' or 'BS123'	The BSID of the base station that the one collected the data of the GPS information, every GPS information belongs to a base station and a base station can have many GPS data thus the GPSID and the BSID are M:1 relation. The BSID is the serial number of the GPS following the time. ((if the base station is earliest built, the serial number would be 1)) And It is the foreign key of table GPS referencing the table BS.

Foreign keys and reasons:

Foreign key :`BSID`

The column `BSID` references bs.BSID, this is to make sure BSID are valid data reflecting the existing BSID of the base station. It corresponds to the GPS information belongs to the Bs station relationship in the ER diagram.

[add more blocks if needed]

Table name: virus type

Table design explanation

The virus type table is used to store the data of the virus information which will be displayed in the test report. The primary key is the virus type because every virus has a unique virus name (which is the virus type in this case) in real life. And the foot note describe the virus is needed as the example shows. Thus, we set those as the attribute of this table. And no more data needs to be referenced in this case.

Column	Domain	Explanation
Virus type Varchar (40)	Any valid virus type is acceptable like the 'COVID-19' , 'variola' or 'unknown'	The virus type is the unique name of a virus, and it can be 'unknown' when there is no data of this virus before. It is the primary key.
FootNote Text (65,535)	Any valid type of the foot note is acceptable.	You can even write a research report about the virus.

Foreign keys and reasons:

Foreign key : there is no foreign key.

Table name: detection

Table design explanation

The virus type table is used to store the data of the virus information which will be displayed in the test report. The primary key is the virus type because every virus has a unique virus name (which is the virus type in this case) in real life. And the foot note describes the virus is needed as the example shows. Thus, we set those as the attribute of this table. And no more data needs to be referenced in this case.

Column	Domain	Explanation
MobileID Varchar (40)	Any valid form of MobileID is acceptable such as 1-800-273-8255 or +86 15199998888, which must have character representing the place that the sim card belongs to.	It is the foreign key that references the table `person` And it is the phone number of the person who did this test.
Sample collect time Datetime (6)	From 1000-01-01 00:00:00 to 9999-12-31 23:59:59	The time that the Sample was collected. It must be before `Sample test time`.
Sample test time Datetime (6)	From 1000-01-01 00:00:00 to 9999-12-31 23:59:59	The time the sample was tested. It must be before `Sample report time`
Sample report time Datetime (6)	From 1000-01-01 00:00:00 to 9999-12-31 23:59:59	The time the sample was reported. It must be the latest time among all the time in this table.
DetectionID Varchar (20)	Any valid form of the combination of the letter and number is acceptable. For example ‘Detection 01	The serial number of the detection which must be UNIQUE. And it is the primary key of this table.
DocID Varchar (40)	Any valid form of the combination of the letter and number is acceptable. For example ‘D01’ or ‘LA hospital01 D01’	The string of letters and number that represents a specific doc which is UNIQUE and references the doctor.DocID.
Sample type Varchar (40)	Any valid virus type is acceptable like the ‘COVID-19’ , ‘variola’ or ’unknown’	The sample type of the virus which is the unique name of a virus, and it can be ‘unknown’ when there is no data of this virus before. It is the foreign key referencing `Virus type`.Virus type`

Foreign keys and reasons:

Foreign key : `MobileID`, `Sample type`, `DocID`

Reason: The column `MobileID` references person.MobileID, this is to make sure MobileID are valid data reflecting the existing MobileID of the persons. It corresponds to a person has a detection report relationship in the ER diagram.

The column `Sample type` references `virus type`. Virus type, this is to make sure Sample type are valid data reflecting the existing Sample type of viruses. It corresponds to the detection report concludes a virus type relationship in the ER diagram.

The column `DocID`references doctor.DocID, this is to make sure DocID are valid data reflecting the existing DocID of the doctors. It corresponds to the doctor do the detection report relationship in the ER diagram.

Table name: travelhistorytracking

Table design explanation:

The travelhistorytracking table is used to store the data of the travel history of a person which records the time that the person access to the range of the base station and the time he/she left which are connected with the sim card. So we need the BSID that referencing the bs.bsid and the SimcardID that referencing person.MobileID as foreign keys and the primary key is the TrackingID because it is UNIQUR for every tracking information as we set this as the serial number of a tracking information. Thus, we set those as attributes of this table.

Column	Domain	Explanation
SimcardID Varchar (40)	Any valid form of MobileID is acceptable such as 1-800-273-8255 or +86 15199998888, which must have character representing the place that the sim card belongs to.	It is a foreign key of this table referencing person.MobileID. And it is the phone number of who access to the range of the base station.
SimCardConnectionTime Datetime (6)	From 1000-01-01 00:00:00 to 9999-12-31 23:59:59	The time that the person access to the range of the base station
SimCardDisconnectTime Datetime (6)	From 1000-01-01 00:00:00 to 9999-12-31 23:59:59	The time that the person left the range of the base station.
TrackingID Varchar (20)	Any valid form of combination of letters and numbers is acceptable. For example 'T01'	The serial number of the tracking data and it is UNIQUE for every tracking information as well as it is the primary key of this table.
BSID Varchar (255)	Any valid type of combination of letters and numbers is	It is a foreign key of this table that referencing the bs.bsid.

	acceptable. For example, ‘4046’ or ‘BS123’	And it reflects which base station that collect this data.

Foreign keys and reasons:

Foreign key: `BSID`, `SImcardID`

Reason: The column `BSID` references bs.BSID, this is to make sure BSID are valid data reflecting the existing BSID of the base station. It corresponds to the travel history records belongs to a base station relationship in the ER diagram.

The column `SImcardID` references person.MobileID, this is to make sure SImcardID are valid data reflecting the existing MobileID of the citizens. It corresponds to travel history record records the person’s MobileID relationship in the ER diagram.

Table name: doctor

Table design explanation:

The doctor table is used to store the data of the doctor’s information in every hospital. The primary key is the DocID which is UNIQUE for every doctor and the foreign key is the HospitalID which reflects the Hospital that the doctor works in. in addition, every doctor has a name and it can be the same as other doctor’s. Thus we set those as attributes of this table.

Column	Domain	Explanation
DocID Varchar (40)	Any valid form of the combination of the letter and number is acceptable. For example ‘D01’ or ‘LA hospital01 D01’	It is the string that represents a specific doctor and it is unique for every doctor as well as it is the primary key of this table.
Doc name Varchar (40)	Any valid form of name is acceptable. For example ‘Mark’ or‘zhang san’	Every doctor has a name. It can be duplicate.
HospitalID Varchar (40)	Any valid form of combination of letters and numbers is acceptable. For example ‘LA hospital 01’	It is the foreign key that references the hospital.hospitalID and it reflects the hospital that the doctor works in.

Foreign keys and reasons:

Foreign key: `HospitalID`

Reason: The column `HospitalID` references hos[ital. `HospitalID`, this is to make sure `HospitalID` are valid data reflecting the existing `HospitalID` of the hospital. It corresponds to the doctor works in a hospital relationship in the ER diagram.

Table name: hospital

Table design explanation:

The hospital table is used to store the data of the hospital's information in every districts. The primary key is the HospitalID which is UNIQUE for every hospital and the foreign key is the districtID which reflects the district that the hospital locates. In addition, every hospital has a name and it can be the same as other hospital's. Thus we set those as attributes of this table.

Column	Domain	Explanation
Hospital ID Varchar (40)	Any valid form of combination of letters and numbers is acceptable. For example 'LA-H01' or 'H 01'	It is the string that represents a specific hospital. it is unique for every hospital as well as it is the primary key of this table.
Hospital name Varchar (40)	Any valid form of combination of letters, words and expressions is acceptable. For example 'LA first hospital '	The name of this hospital It can be duplicate.
DitrictID Varchar (40)	District id from 000 to 999	It is the foreign key referencing the district,ditrictID reflects district that the hospital locates

Foreign keys and reasons:

Foreign key: `DistrictID`

Reason: The column`DistrictID` references district.`DistrictID`, this is to make sure `DistrictID` are valid data reflecting the existing `DistrictID` of the district. It corresponds to the hospital locates in a district relationship in the ER diagram.

Table name: bs

Table design explanation:

The bs table is used to store the data of the base stations' information in every districts. The primary key is the BSID which is UNIQUE for every base station and the foreign key is the DistrcitID which reflects the district that the base station belongs to. Thus we set those as attributes of this table.

Column	Domain	Explanation

BSID Varchar (255)	Any valid type of combination of letters and numbers is acceptable. For example, '4046' or 'BS123'	BSID is unique for every base station as well as it is the primary key of this table. It is the string that represents a specific base station.
DistrictID Varchar (40)	District id from 000 to 999	It is the foreign key referencing the bs.bsid that reflects district that the base station locates

Foreign keys and reasons:

Foreign key: `DistrictID`

Reason: The column `DistrictID` references district.`DistrictID`, this is to make sure `DistrictID` are valid data reflecting the existing `DistrictID` of the district. It corresponds to the base station locates in a district relationship in the ER diagram.

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:

DocID

Doc_name

HospitalID

Hospital_name

Name

Sex

Age

MobileID

Sample type

Sample result

Sample collected time

Sample test time

Sample Report time

Footnote

DetectionID

FDs (Indicate partial or transitive dependencies):

Firstly, we find that the MobileID, DetectionID, DocID and the HospitalID are a unique key since every person has a mobile phone which corresponds to a unique MobileID and the same as that we set every detection report, doctor, Hospital has an ID uniquely corresponding to it. And for this viral test report, as defined, we set the DetectionID as the primary key of this table now. Then, we find that there are serval partial dependencies:

If you know Mobile ID and Detection ID, you can find out Sample type, Sample result, Sample collected time, Sample test time, Sample Report time, Footnote and with Detection ID only, you can find out the same result, too

If you know Hospital ID and Detection ID, you can find out Sample type, Sample result, Sample collected time, Sample test time, Sample Report time, Footnote and with Detection ID only, you can find out the same result, too

If you know Doctor ID and Detection ID, you can find out Sample type, Sample result, Sample collected time, Sample test time, Sample Report time, Footnote and with Detection ID only, you can find out the same result, too

Therefore, we should divide the Detection ID apart from other three unique keys and we know a detection report is supposed to have Sample type, Sample result, Sample collected time, Sample test time, Sample Report time, Footnote then we set those as attribute to the table `detection` and set the Detection ID as a primary key of this table. And the detection report belongs to a person thus we add the MobileID as a foreign key.

Namely, we find partial dependencies:

If you know Mobile ID and Detection ID, you can find out the person's name, age, sex, and Mobile ID and with Mobile ID only, you can find out the same result, too

If you know Mobile ID and Doctor ID, you can find out the person's name, age, sex, and Mobile ID and with Mobile ID only, you can find out the same result, too

If you know Mobile ID and Hospital ID, you can find out the person's name, age, sex, and Mobile ID and with Mobile ID only, you can find out the same result, too

Therefore, we should divide the Detection ID apart from other three unique keys and we know a detection report is supposed to have the name, sex, age and the Mobile ID of a person, then we set those as attribute to the table `person` and set the Mobile ID as a primary key of this table.

More:

If you know Doctor ID and Detection ID, you can find out the doctor's name, ID and the hospital he/she works in, and with doctor ID only, you can find out the same result, too

If you know Mobile ID and Doctor ID, you can find out the doctor's name, ID and the hospital he/she works in, and with doctor ID only, you can find out the same result, too

If you know Doctor ID and Hospital ID, you can find out the doctor's name, ID and the hospital he/she works in, and with doctor ID only, you can find out the same result, too

Therefore, we should divide the doctor ID apart from other three unique keys and we know a doctor is supposed to have the name, ID, then we set those as attribute to the table `doctor` and set the DocID as a primary key of this table as well as add the HospitalID as a foreign key .

And:

If you know Hospital ID and Detection ID, you can find out Hospital 's name, ID and the district it locates in (which is not the content in this case), and with Hospital ID only, you can find out the same result, too

If you know Hospital ID and Doctor ID, you can find out Hospital 's name, ID and the district it locates in (which is not the content in this case), and with Hospital ID only, you can find out the same result, too

If you know Mobile ID and Hospital ID, you can find out Hospital 's name, ID and the district it locates in (which is not the content in this case), and with Hospital ID only, you can find out the same result, too

Therefore, we should divide the Detection ID apart from other three unique keys and we know a hospital is supposed to have the name, ID, then we set those as attribute to the table hospital and set the Hospital ID as a primary key of this table.

Normalised tables and which normal form they are currently in:

Person:(3NF)

Name: varchar (40)	MobileID varchar (40)	Age: int (3)	Sex: varchar (20)

Detection: (2NF)

MobileID: Varchar (40)	Sample test time: datetime (6)	Sample report time: datetime(6)	Sample result: text	Sample collect time: datetime (6)	detectionID: varchar (40)	Sample type: Varchar (40)	DocID: Varchar (40)	Footnote: text

Doctor:(3NF)

DocID: Varchar (40)	Doc_name: Varchar (40)	HospitalID: Varchar (40)

Hospital: (3NF)

HospitalID: Varchar (40)	Hospital_Name: Varchar (40)	DistrictID: Varchar (40)

Stage 2

Attributes:

For the detection table, we have

MobileID

Sample type

Sample result

Sample collected time

Sample test time

Sample Report time

Footnote

DetectionID

FDs (Indicate partial or transitive dependencies):

And we find the transitive dependency:

If you infer the foot note through the virus type and you can infer the virus type once if you know the detection number which means the FootNote is functionally depends on Virus type and the Virus type is functionally depends on the detectionID. Thus, there is transitive dependency in the detection table.

Normalised tables and which normal form they are currently in:

Detection: (3NF)

MobileID: Varchar (40)	Sample test time: datetime (6)	Sample report time: datetime (6)	Sample result: text	Sample collect time: datetime (6)	detectionID: varchar (40)	DocID: Varchar (40)	Sample Type: Varchar (40)

Virsus type(3NF)

Virus Type : Varchar (40)	Type Description: Text

Stage 3 [If necessary]

Attributes:

FDs (Indicate partial or transitive dependencies):

Normalised tables and which normal form they are currently in:

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.

Following are the test data of this data base for those 18 cases.

It can be applied for all those cases.

And in the Specific case explanation I'll show you the specific data that I used in this data base.

INSERT INTO `bs` (`BSID`, `DistrictID`) VALUES

('4045', '213'),

('4046', '213'),

('4041', '404'),

('4047', '404');

INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

('23652546', '2021-10-19 19:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'D013', 'D03', 'Panic21'),

('233636', '2021-10-09 19:00:00.000000', '2021-10-09 19:30:00.000000', 'positive', '2021-10-09 16:00:00.000000', 'De01', 'D01', 'Panic21'),

('+86 7777777', '2021-10-03 02:00:00.000000', '2021-10-03 19:30:00.000000', 'negative', '2021-10-03 01:00:00.000000', 'De018', 'D01', 'Panic21'),

('+86 7777777', '2021-10-04 03:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-03 02:00:00.000000', 'De019', 'D01', 'Panic21'),

('87539454', '2021-10-04 19:00:00.000000', '2021-10-04 19:28:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De02', 'D01', 'Panic21'),

('86534343', '2021-10-03 19:00:00.000000', '2021-10-09 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De020', 'D02', 'Panic21'),

('86534343', '2021-10-03 19:05:00.000000', '2021-10-09 19:30:00.000000', 'negative', '2021-10-03 19:00:00.000000', 'De021', 'D02', 'Panic21'),

('45675865', '2021-10-04 19:00:00.000000', '2021-10-04 19:30:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De03', 'D03', 'Panic21'),

('231425534', '2021-10-05 19:00:00.000000', '2021-10-05 19:05:00.000000', 'positive', '2021-10-05 16:00:00.000000', 'De04', 'D04', 'Panic21'),

('34536344', '2021-10-05 19:00:00.000000', '2021-10-05 19:04:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De05', 'D04', 'Panic21'),

('86545865', '2021-10-05 19:00:00.000000', '2021-10-05 19:02:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De06', 'D06', 'Panic21'),

('28784343', '2021-10-05 19:00:00.000000', '2021-10-05 19:01:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De07', 'D08', 'Panic21'),

('233636', '2021-10-04 19:00:00.000000', '2021-10-04 19:05:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De08', 'D01', 'Panic21'),

('233636', '2021-10-05 19:07:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-05 19:06:00.000000', 'De09', 'D01', 'Panic21'),

('4567564', '2021-10-03 00:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-02 00:00:00.000000', 'De10', 'D01', 'Panic21'),

('4567564', '2021-10-04 00:00:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De11', 'D01', 'Panic21'),

('4567564', '2021-10-05 00:00:00.000000', '2021-10-06 19:30:00.000000', 'negative', '2021-10-04 16:00:00.000000', 'De12', 'D01', 'Panic21'),

('4567564', '2021-10-19 00:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'De14', 'D01', 'Panic21'),

('86545865', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De15', 'D01', 'Unknown'),

('28784343', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De16', 'D02', 'Unknown'),

('5324342', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De17', 'D03', 'Unknown');

INSERT INTO `district` (`DistrictID`, `RIskLevel`, `District_name`, `Region`) VALUES

('213', 'high', 'Centre Lukewarm Hillside', 'central'),

('303', 'low', 'Raspberry town', 'west'),

('404', 'mid', 'Glow Sand district', 'central'),

('415', 'low', 'Bunny Tail district', 'east'),

('718', 'high', 'Lenny town', 'south');

INSERT INTO `doctor` (`DocID`, `Doc Name`, `HospitalID`) VALUES

('D01', 'Singed', 'H01'),

('D02', 'Akali', 'H01'),

('D03', 'Sin', 'H02'),

('D04', 'Strange', 'H02'),

('D05', 'Tony stark', 'H03'),

('D06', 'peper', 'H03'),

('D07', 'Drake', 'H04'),

('D08', 'Laroi', 'H04'),

('D09', 'Mengduo', 'H05');

```
INSERT INTO `gps` (`GPSID`, `E/W`, `N/S`, `Longtitude`, `Latitude`, `BSID`) VALUES
(1, 'E', 'N', '118.429390', '34.107824', '4041'),
(2, 'E', 'S', '118.350562', '34.098976', '4047'),
(3, 'W', 'N', '82.277592', '42.589128', '4045'),
(4, 'E', 'N', '74.890385', '43.359051', '4046');

INSERT INTO `hospital` (`Hospital_ID`, `Hospital_Name`, `DistrictID`) VALUES
('H01', 'Central Lukewarm First Kingdom Hospital', '213'),
('H02', 'Central Lukewarm Second Kingdom Hospital', '213'),
('H03', 'Glow Sand district First Hospital', '404'),
('H04', 'Glow Sand district Second Hospital', '404'),
('H05', 'Raspberry town first Hospital', '303');

INSERT INTO `virsus type` (`Virus Type`, `Type Description`) VALUES
('Panic21', 'Do not panic if this happen'),
('Phasmophobia21', 'It is coming!'),
('Unknown', 'null');

INSERT INTO `travelhistorytracking` (`SimcardID`, `SimCardConnectionTime`, `SimCardDisconnectTime`,
`TrackingID`, `BSID`) VALUES
('233636', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T01', '4041'),
('233636', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T02', '4046'),
('8906453', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T03', '4041'),
('231425534', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T04', '4045'),
('64253465', '2021-10-10 19:30:00.000000', '2021-10-10 19:31:00.000000', 'T05', '4045'),
('64253465', '2021-10-10 19:30:00.000000', '2021-10-10 19:31:00.000000', 'T06', '4045'),
('4567564', '2021-10-17 19:30:00.000000', '2021-10-18 19:30:00.000000', 'T07', '4045');

INSERT INTO `person` (`Name`, `MobileID`, `Age`, `Sex`) VALUES
('Cnangzhang', '+86 7777777', 16, 'male'),
('Bren joy', '202020', 70, 'male'),
```

('Powder', '231425534', 15, 'female'),

('Mark', '233636', 42, 'male'),

('Suga', '23652546', 27, 'male'),

('Melee', '28784343', 26, 'male'),

('Kiana', '34536344', 15, 'female'),

('Arene', '4567564', 18, 'male'),

('Caitlyn', '45675865', 25, 'Female'),

('Blemishine', '5324342', 18, 'female'),

('Carnelian', '64253465', 33, 'female'),

('Whisperain', '86534343', 23, 'female'),

('Indigo', '86545865', 20, 'female'),

('Vi', '87539454', 24, 'female'),

('Heavyrain', '8906453', 15, 'female');

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:

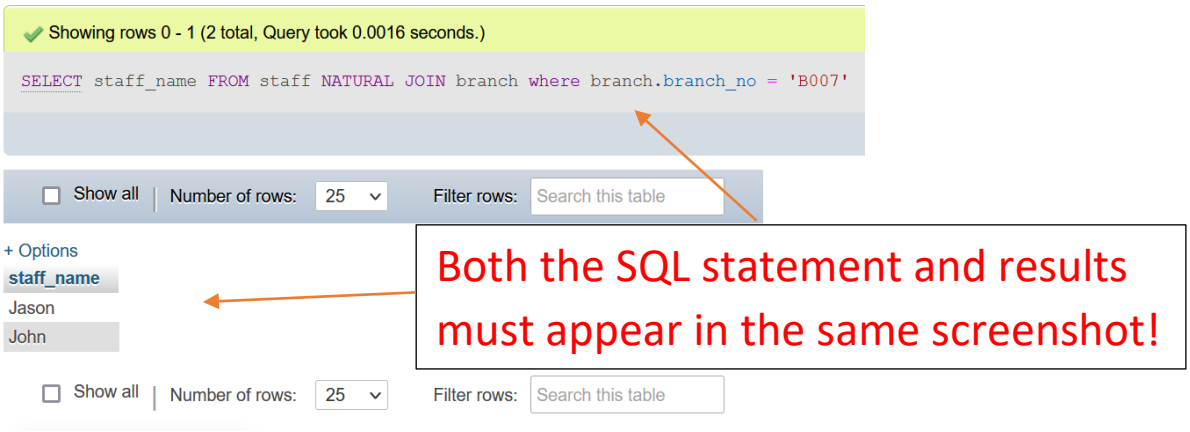
The following staff member information is added to the staff table (some attributes are hidden as they are not related to this task)

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 SimcardID
from `travelhistorytracking`,`bs`
where bs.DistrictID in
(select districtID
from travelhistorytracking,bs
where SimcardID = '233636')
AND   SimCardConnectionTime >= '2021-10-07 19:30:00'
AND   SimCardConnectionTime <='2021-10-09 19:30:00'
```

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

We set there are a lot districts among those 5 regions and we choose one of the districts and 2 hospitals in a district, 2 doctors in a hospital, a doctor did 2 detections. Those setting satisfy the M:1 relationship between district-hospital, hospital-doctor and doctor-detection as well as there are 2 base station in a district, more than 1 GPS information in the base station. Thus, we first settle the condition that between '2021-10-07 19:30:00' and '2021-10-09 19:30:00'. Then we settle place that mark had been to in the interval of the time mentioned above which is the context of line4, 5, 6. Then we combine the two conditions with and to constrain the domain of the bs which can directly satisfy the place that mark has been to in 48H as requested. Then, we put the statement that will select the SimcardID distinctly because there is more than one travel record for mark thus it has to be distinct.

In this case, we insert

('233636', '2021-10-09 19:00:00.000000', '2021-10-09 19:30:00.000000', 'positive', '2021-10-09 16:00:00.000000', 'De01', 'D01', 'Panic21'), to make mark tested positive as required and

('233636', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T01', '4041'),

('233636', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T02', '4046'), to make mark went to 2 district so that we can test if the code select more than one area that mark has been to.

And we set a person has been in the area that base station 4041 lies in and another person in the area that 4046 base station locates in. Both of the time is the same as the time mark went to.

('8906453', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T03', '4041'),

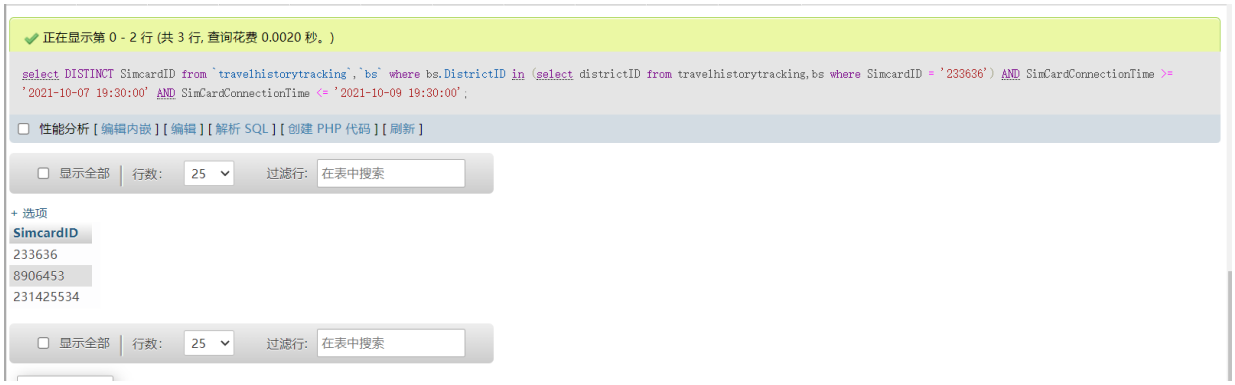
('231425534', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T04', '4045'),

(4045 is in the same district as 4046 according to my coding about how I create the database)

We have data whose SimcardConnectionTime is out of the domain from 2021-10-07 19:30 to 2021-10-09 19:30

and SimcardID didn't equals 233636. So the code is correct if we didn't select those data out pf the domain.

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):

https://blog.csdn.net/dengdun6257/article/details/102284035?utm_source=app&app_version=4.20.0

The GPS format:

`E/W` char (10) NOT NULL,

`N/S` char (10) NOT NULL,

`Longitude` varchar (20) NOT NULL,

`Latitude` varchar (20) NOT NULL,

E/W dddmm.mmmm the E/W represent the Eastern/Western hemisphere, and the ddd represents the degrees of the longitude and the mm.mmmm represents the minutes of the longitude.

N/S ddmm.mmmm the N/S represent the North/South hemisphere and the dd represents the degrees and the mm.mmmm represents the minutes of the latitude.

Your SQL statement(s) for travel record insertion:

In this case,
we set this person's name is "Carnelian" a "female" ages "33" and the phone numbe r is "64253465"

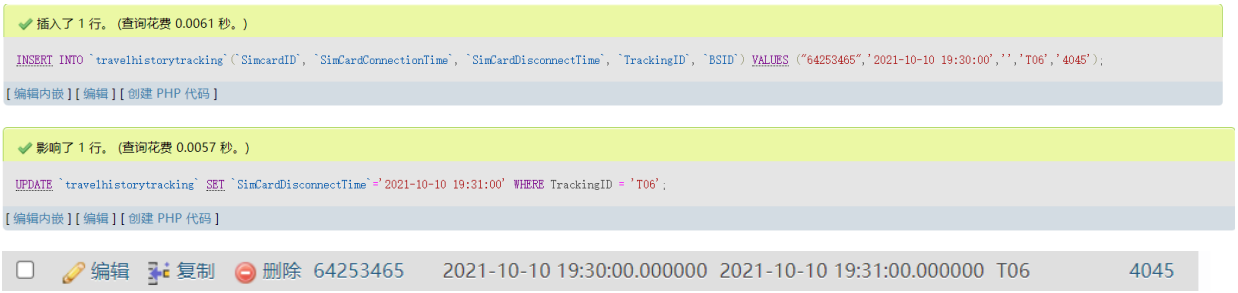
And she enters the range of the base station at 2021-10-10 19:30:00.

```
INSERT INTO `travelhistorytracking`(`SimcardID`, `SimCardConnectionTime`, `Si
mCardDisconnectTime`, `TrackingID`, `BSID`)
VALUES
("64253465",'2021-10-10 19:30:00','','T06','4045')
```

when the user moves out of the range

```
UPDATE `travelhistorytracking` SET
`SimCardDisconnectTime`='2021-10-10 19:31:00'
 WHERE TrackingID = 'T06';
```

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 Hospital_Name FROM

(SELECT `Hospital_Name`, AVG(TimeStampDiff(second,`Sample test time` , `Sample report time`)) AS
`Report generation time`

FROM (`hospital`INNER JOIN `doctor` ON hospital.Hospital_ID = doctor.HospitalID) INNER JOIN
`detection` ON doctor.DocID = Detection.DocID

GROUP BY `hospital`.`Hospital_Name`

ORDER BY `Report generation time` ASC

LIMIT 1) AS `newTable`
```

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

Test data: **we set up 4 hospitals**

INSERT INTO `hospital` (`Hospital_ID`, `Hospital_Name`, `DistrictID`) VALUES

('H01', 'Central Lukewarm First Kingdom Hospital', '213'),

('H02', 'Central Lukewarm Second Kingdom Hospital', '213'),

('H03', 'Glow Sand district First Hospital', '404'),

('H04', 'Glow Sand district Second Hospital', '404');

The we set many cases that make sure every hospital has enough data for the calculation

INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

('23652546', '2021-10-19 19:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'D013', 'D03', 'Panic21'),

('233636', '2021-10-09 19:00:00.000000', '2021-10-09 19:30:00.000000', 'positive', '2021-10-09 16:00:00.000000', 'De01', 'D01', 'Panic21'),

('+86 7777777', '2021-10-03 02:00:00.000000', '2021-10-03 19:30:00.000000', 'negative', '2021-10-03 01:00:00.000000', 'De018', 'D01', 'Panic21'),

('+86 7777777', '2021-10-04 03:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-03 02:00:00.000000', 'De019', 'D01', 'Panic21'),

('87539454', '2021-10-04 19:00:00.000000', '2021-10-04 19:28:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De02', 'D01', 'Panic21'),

('86534343', '2021-10-03 19:00:00.000000', '2021-10-09 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De020', 'D02', 'Panic21'),

('86534343', '2021-10-03 19:05:00.000000', '2021-10-09 19:30:00.000000', 'negative', '2021-10-03 19:00:00.000000', 'De021', 'D02', 'Panic21'),

('45675865', '2021-10-04 19:00:00.000000', '2021-10-04 19:30:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De03', 'D03', 'Panic21'),

('231425534', '2021-10-05 19:00:00.000000', '2021-10-05 19:05:00.000000', 'positive', '2021-10-05 16:00:00.000000', 'De04', 'D04', 'Panic21'),

('34536344', '2021-10-05 19:00:00.000000', '2021-10-05 19:04:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De05', 'D04', 'Panic21'),

('86545865', '2021-10-05 19:00:00.000000', '2021-10-05 19:02:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De06', 'D06', 'Panic21'),

('28784343', '2021-10-05 19:00:00.000000', '2021-10-05 19:01:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De07', 'D08', 'Panic21'),

('233636', '2021-10-04 19:00:00.000000', '2021-10-04 19:05:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De08', 'D01', 'Panic21'),

('233636', '2021-10-05 19:07:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-05 19:06:00.000000', 'De09', 'D01', 'Panic21'),

('4567564', '2021-10-03 00:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-02 00:00:00.000000', 'De10', 'D01', 'Panic21'),

('4567564', '2021-10-04 00:00:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De11', 'D01', 'Panic21'),

('4567564', '2021-10-05 00:00:00.000000', '2021-10-06 19:30:00.000000', 'negative', '2021-10-04 16:00:00.000000', 'De12', 'D01', 'Panic21'),

('4567564', '2021-10-19 00:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'De14', 'D01', 'Panic21'),

('86545865', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De15', 'D01', 'Unknown'),

('28784343', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De16', 'D02', 'Unknown'),

('5324342', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De17', 'D03', 'Unknown');

We first create a `newTable` to find out each hospital and its report generation time in ASC so that we can find the first row is the hospital which has the least time to do the test.

The 'Order by' in the where clause satisfy the `newTable` will display the `Report generation time` from in ascending order which means the smallest number is at the head.

The' SELECT `Hospital_Name`, AVG(TimeStampDiff(second,`Sample test time` , `Sample report time`)) AS `Report generation time`

FROM (`hospital`INNER JOIN `doctor` ON hospital.Hospital_ID = doctor.HospitalID) INNER JOIN `detection` ON doctor.DocID = Detection.DocID

GROUP BY `hospital`.`Hospital_Name`' statement select the `Hospital_Name` and the `Report generation time` grouped by the`Hospital_Name`

Then with the help of those, we get a new table that each hospital has a corresponding average report generation time displayed in ascending order.

And then **we select the first row of this table whose hospital name corresponds to the smallest time** by using 'Limit 1'

Then we combine them by using subquery.

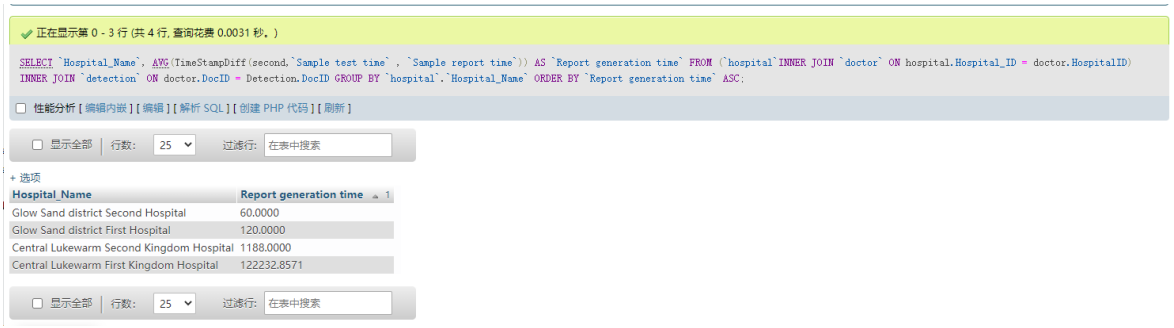
As calculated, we find the 4 hospitasl have four different data of the "Report generation time" and the Glow Sand district second Hospital has the least time

So, the result fit the data as what we calculate.

The result of the SELECT statement (screenshot):

We first create a `newTable` to find out each hospital and its report generation time in ASC so that we can find the first row is the hospital which has the least time to do the test.

This is the `newTable` looks like but **NOT THE FINAL RESULT**



The Final Result:

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 `MobileID` From detection where

`Sample test time`>='2021-10-03 00:00'

and `Sample Test Time` <='2021-10-05 00:00'

GROUP BY MobileID

HAVING

COUNT(MobileID) = 2

and timediff(max(`Sample Test Time`),min(`Sample Test Time`)) > '24:00:00';

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

We set a new person who have done 2 tests from 2021-10-03 00:00 to 2021-10-05 00:00 whose gap time is 25 hours

INSERT INTO `person`(`Name`, `Sex`, `Age`,`MobileID`)

VALUES

("Cnangzhang","male","16","+86 7777777")

INSERT INTO `detection`(`DetectionID`, `MobileID`,`Sample Result`, `Sample Collect Time`, `Sample Test Time`, `Sample Report Time`, `DocID`, `Sample Type`)

VALUES

("De018","+86 7777777", "negative", "2021-10-03 01:00:00", "2021-10-03 02:00:00", "2021-10-03 19:30:00", "D01", "Panic21")

("De019","+86 7777777", "negative", "2021-10-03 02:00:00", "2021-10-04 03:00:00", "2021-10-04 19:30:00", "D01", "Panic21")

Following set a person did 2 times of test but within gap time of 23 hours

INSERT INTO `person`(`Name`, `Sex`, `Age`,`MobileID`)

VALUES

("Whisperain","female","23","86534343")

INSERT INTO `detection`(`DetectionID`, `MobileID`,`Sample Result`, `Sample Collect Time`, `Sample Test Time`, `Sample Report Time`, `DocID`, `Sample Type`)

VALUES

("De020","86534343", "negative", "2021-10-03 16:00:00", "2021-10-03 19:00:00", "2021-10-09 19:30:00", "D02", "Panic21"),

("De021","86534343", "negative", "2021-10-03 19:00:00", "2021-10-03 19:05:00", "2021-10-09 19:30:00", "D02", "Panic21")

And then we set a person did three times of test whose gap time is 24 hours to testify if the code runs successfully

INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

('4567564', '2021-10-03 00:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-02 00:00:00.000000', 'De10', 'D01', 'Panic21'),

('4567564', '2021-10-04 00:00:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De11', 'D01', 'Panic21'),

('4567564', '2021-10-05 00:00:00.000000', '2021-10-06 19:30:00.000000', 'negative', '2021-10-04 16:00:00.000000', 'De12', 'D01', 'Panic21'),

As the question requested, the second and third person are not supposed to be selected and the

'HAVING COUNT(MobileID) = 2'

in where clause **pick out who did test more than 2 times**

and

'and timediff(max(`Sample Test Time`),min(`Sample Test Time`)) > '24:00:00';'

makes sure that the **gap time satisfy that it will be at least 24 hours.**

Then we use

'select `MobileID` From detection where

`Sample test time`>='2021-10-03 00:00'

and `Sample Test Time` <='2021-10-05 00:00'

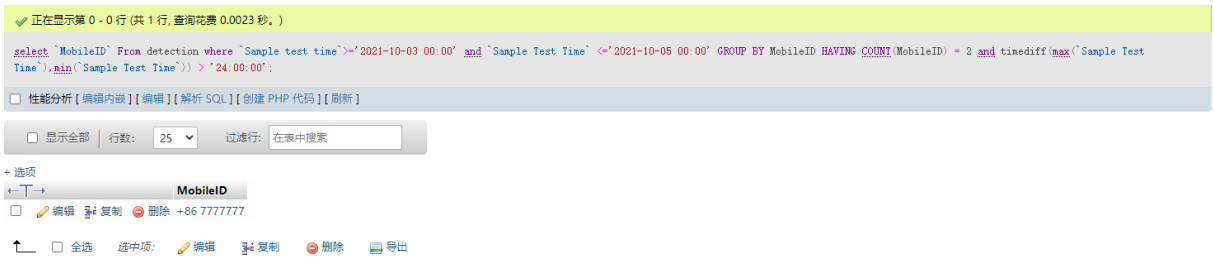
'to select all the mobileID of the person who did test between 2021-10-03 00:00 and 2021-10-05 00:00

and we need to list information by each MobileID thus we use 'group by'.

Then we use 'having' to add the condition that the test times must be 2 and have gap time of at least 24 hours.

Therefore, the result must be “+86 7777777” if the code it is correct.

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:

```
case 5
SELECT `District_name`,`RIskLevel`FROM `district`
ORDER BY CASE `RIskLevel`
WHEN 'high'THEN 1
WHEN 'mid' then 2
WHEN 'low' THEN 3
END
```

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`, person.MobileID
From `person`INNER JOIN`detection` INNER JOIN `district` INNER JOIN `hospital`
 INNER JOIN `doctor`
WHERE(district.DistrictID = hospital.DistrictID)
```

AND(hospital.Hospital_ID = doctor.HospitalID)
AND(doctor.DocID = detection.DocID)
AND(detection.MobileID = person.MobileID)
AND(`Sample result` = 'positive')
AND(`Sample report time` BETWEEN '2021-10-04 00:00:00' AND '2021-10-04 23:59:59')
AND district.District_name = 'Centre Lukewarm Hillside'

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

Test data: INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

('87539454', '2021-10-04 19:00:00.000000', '2021-10-04 19:28:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De02', 'D01', 'Panic21'),

('45675865', '2021-10-04 19:00:00.000000', '2021-10-04 19:30:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De03', 'D03', 'Panic21'),

We set two person Vi and Caitlyn did the test and the test report time is the "2021-10-04 19:28:00" and "2021-10-04 19:30:00" , which make both of them are tested positive in 2021-10-04 and in the district 'Centre Lukewarm Hillside'

Since D01 and D03 are the ID of the doctor that works in 'Centre Lukewarm Hillside First Hospital' and 'Centre Lukewarm Hillside Second Hospital' ,and both of then locates in centre lukewarm hillside.

The result of the SELECT statement (screenshot):

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 COUNT(DISTINCT D2.MobileID)- COUNT(DISTINCT D1.MobileID) AS `Increment` FROM `detection` D1, `detection` D2
WHERE (D1.`Sample Test Time` LIKE '2021-10-04%')
AND(D2.`Sample Test Time` LIKE '2021-10-05%')
AND (D1.`Sample Result` = 'positive')
AND(D2.`Sample Result` = 'positive')

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

Test data:

INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

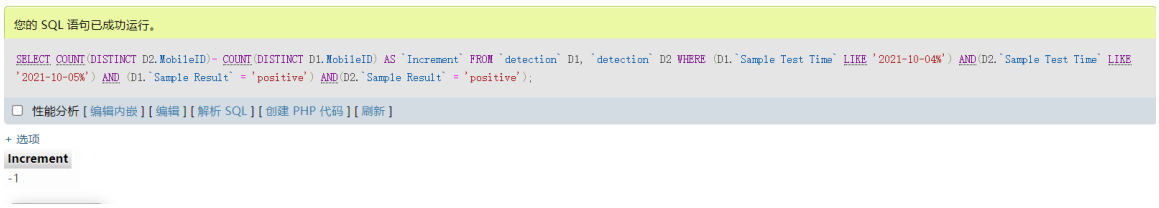
('87539454', '2021-10-04 19:00:00.000000', '2021-10-04 19:28:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De02', 'D01', 'Panic21'),

('45675865', '2021-10-04 19:00:00.000000', '2021-10-04 19:30:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De03', 'D03', 'Panic21'),

('231425534', '2021-10-05 19:00:00.000000', '2021-10-05 19:05:00.000000', 'positive', '2021-10-05 16:00:00.000000', 'De04', 'D04', 'Panic21'),

As you can see, we set 3 persons detected to be positive and two of the was tested in 10-04 and one of them tested in the 10-05, thus the data make sure the 1:M relation of than a day can have many positive cases and the result can test if the type of the data handles the case that the increment can be negative number.

The result of the SELECT statement (screenshots):



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 COUNT(DISTINCT T2.SimcardID)/ COUNT(DISTINCT T1.simcardID) AS `Spread rate` FROM `travelhistorytracking` T1, `travelhistorytracking` T2,`bs`B1,`bs` B2,`detection`,`district`

WHERE T1.SimcardID IN

(select DISTINCT SimcardID

from `travelhistorytracking`, `bs` B1

where B1.DistrictID in

(select districtID

from travelhistorytracking,bs

where SimcardID = '233636')

AND SimCardConnectionTime >= '2021-10-07 19:30:00'

AND SimCardConnectionTime <='2021-10-09 19:30:00')

AND T2.SimcardID

IN(

SELECT DISTINCT SimcardID FROM `detection` INNER JOIN `travelhistorytracking` INNER JOIN `bs` INNER JOIN `district`

ON detection.MobileID = travelhistorytracking.SimcardID

AND travelhistorytracking.BSID = bs.BSID

and bs.DistrictID = district.DistrictID

WHERE `Sample Result` = 'Positive' AND B2.districtID = B1.DistrictID AND `Simcardconnectiontime` BETWEEN '2021-10-09 19:30:00' AND '2021-10-23 19:30:00');

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

For T1.SImcardID, it is copied from the use case 1 as requested and for T2.Mobile ID, we take the `Sample result` = 'positive' and the time which is the 14 days later than Mark's Sample report time as the condition. Then we combine those together with an `AND` and those codes after 'T2.Simcard in` are the code to find the place that has new positive case in 14 days after detecting mark. In addition, the B1 and B2 make sure the T2.SimcardID is selected from the same area as T1.SImcardID did.

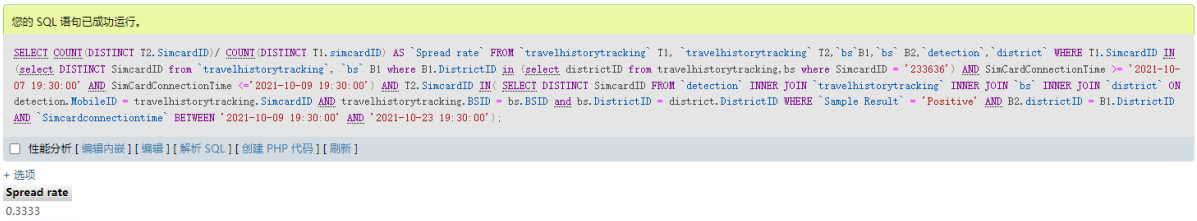
INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`)

VALUES

('4567564', '2021-10-19 00:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'De14', 'D01', 'Panic21'); to make there is positive case in the 14 days after mark was tested.

Thus, those mentioned above set the domain of the data. And the code will be check once if it selects the data out of the domain like what we talked about.

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:

Followed case 8, mark is the beginning of this plague, to trace any possible infected cases the government wants to find

who went to high risk level area in 14 days before mark was tested to be positive. Write a query to select all the citizens that satisfy the conditions.

Your SQL statement:

SELECT DISTINCT SimcardID FROM travelhistorytracking INNER JOIN bs INNER JOIN district ON travelhistorytracking.BSID = bs.BSID AND bs.DistrictID = district.DistrictID

WHERE District.RIskLevel = 'high'

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

INSERT INTO `travelhistorytracking` (`SimcardID`, `SimCardConnectionTime`, `SimCardDisconnectTime`, `TrackingID`, `BSID`) VALUES

('233636', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T01', '4041'),

('233636', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T02', '4046'),

('8906453', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T03', '4041'),

('231425534', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T04', '4045'),

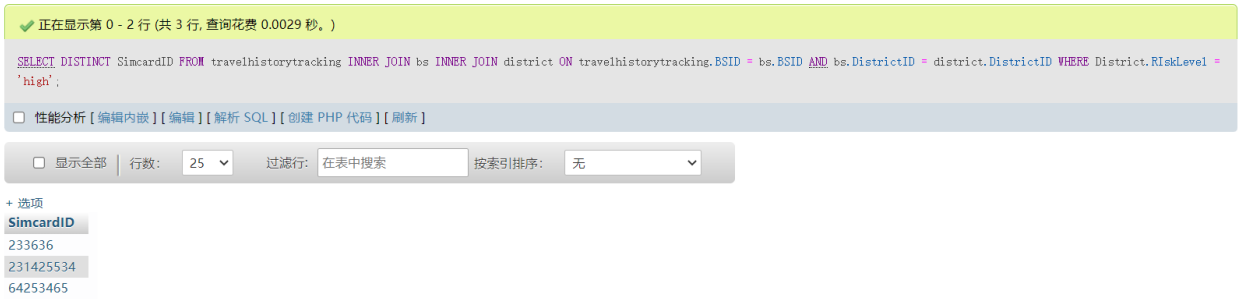
('64253465', '2021-10-10 19:30:00.000000', '2021-10-10 19:31:00.000000', 'T05', '4045');

Test data

As you can see, those are the data that 4041 and 4047 is the bases station of the mid risk level area called Glow Sand district and 4045 and 4046 base station belongs to a high risk-level area called Centre Lukewarm Hillside.

So, in this case the person's mobileID who went to the 4041 will not be recorded.

The result of the SELECT statement (screenshot):



Use case 2: The government wants to find the district where newly appears the positive case after mark carried virus to this country in 14 days. Please find a query to achieve that

Your SQL statement:

SELECT DISTINCT District.DistrictID FROM `detection` INNER JOIN `travelhistorytracking` INNER JOIN `bs` INNER JOIN `district`

ON detection.MobileID = travelhistorytracking.SimcardID

AND travelhistorytracking.BSID = bs.BSID

and bs.DistrictID = district.DistrictID

WHERE `Sample Result` = 'Positive' AND `Simcardconnectiontime` BETWEEN '2021-10-09 19:30:00' AND '2021-10-23 19:30:00';

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

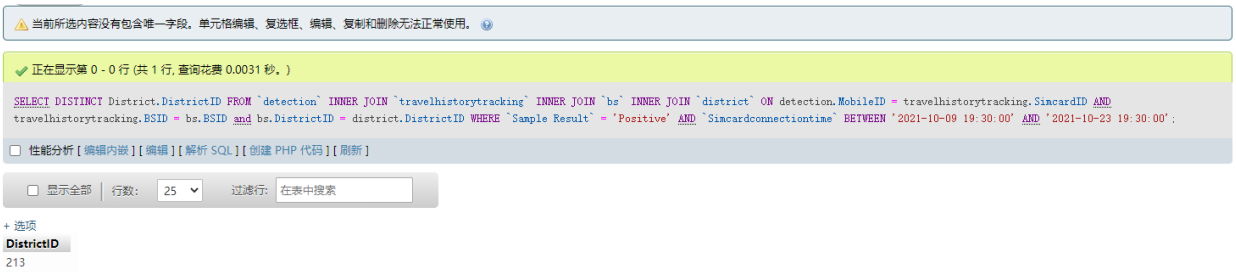
INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`)

VALUES

('4567564', '2021-10-19 00:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'De14', 'D01', 'Panic21');

We set this is the only positive case that has been detected in the following 14 days. He went to the Centre Lukewarm Hillside and the hospital that the D01 works in is located in this area. Thus, the ID 213 is the only district ID that can be output.

The result of the SELECT statement (screenshot):



Use case 3: The government will lock the district once new virus/viruses appears in a district and the positive case is more than 3. Please write a query that find all the district needs to be locked down

Your SQL statement:

SELECT DISTINCT district.DistrictID FROM district INNER JOIN hospital INNER JOIN doctor INNER JOIN detection INNER JOIN `virsus type`

ON district.DistrictID = hospital.DistrictID

AND hospital.Hospital_ID =doctor.HospitalID

AND detection.DocID = doctor.DocID

AND detection.`Sample Type` =`virsus type`.`Virus Type`

WHERE (`virus type`) = "unknown" AND (SELECT COUNT(detection.MobileID) >= "3" FROM `detection`);

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

We firstly insert a virus type unknown with no description and a new person who haven't done any test

INSERT INTO `virsus type`(`Virus Type`, `Type Description`)

VALUES

('Unknown','null')

INSERT INTO `person`(`Name`, `Sex`, `Age`,`MobileID`)

VALUES("Blemishine","female","18","5324342")

Then we set there are 3 positive cases of unknown viruses and all of then do the test in the hospitals of 'Centre lukewarm hillside' whose districtID is `213`

INSERT INTO `detection`(`DetectionID`, `MobileID`,`Sample Result`, `Sample Collect Time`, `Sample Test Time`, `Sample Report Time`, `DocID`, `Sample Type`)

VALUES

("De15","86545865", "positive", "2021-11-09 16:00:00", "2021-11-09 19:00:00", "2021-11-09 19:30:00", "D01", "Unknown"),

("De16"," 28784343", "positive", "2021-11-09 16:00:00", "2021-11-09 19:00:00", "2021-11-09 19:30:00", "D02", "Unknown"),

("De17", '5234342', "positive", "2021-11-09 16:00:00", "2021-11-09 19:00:00", "2021-11-09 19:30:00", "D03", "Unknown")

Thus, we us an `And` combine the conditions that virus type is unknown and sample result is positive together.

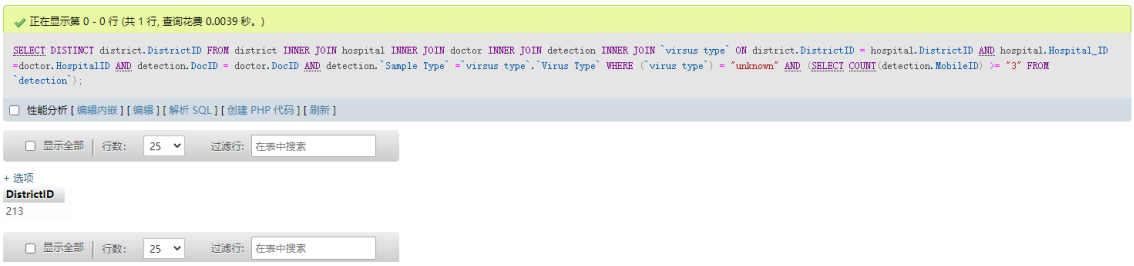
Then we use inner join to get all the table we need to form a new table that we select our answers from.

In the end, we put the combination of conditions in the where clause of the new table.

We only set there is 3 person get the "Unknown" virus and the all do the test in the district 213 since the doctor D01, D02 and D03 work in the hospital locates in the district.

Therefore, only the "Centre Lukewarm Hillside" district will be select if the code is right.

The result of the SELECT statement (screenshot):



Use case 4:

The doctor is more likely to be infected when a case test to be positive.

Now, the government wants to find doctors who have detected positive case in the past and detect them. Please write query to handle this case.

Your SQL statement:

SELECT DISTINCT doctor.DocID FROM doctor INNER JOIN detection

ON doctor.DocID = detection.DocID

WHERE detection.`Sample Result` = 'Positive'

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

INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

('23652546', '2021-10-19 19:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'D013', 'D03', 'Panic21'),

('233636', '2021-10-09 19:00:00.000000', '2021-10-09 19:30:00.000000', 'positive', '2021-10-09 16:00:00.000000', 'De01', 'D01', 'Panic21'),

('87539454', '2021-10-04 19:00:00.000000', '2021-10-04 19:28:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De02', 'D01', 'Panic21'),

('45675865', '2021-10-04 19:00:00.000000', '2021-10-04 19:30:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De03', 'D03', 'Panic21'),

('231425534', '2021-10-05 19:00:00.000000', '2021-10-05 19:05:00.000000', 'positive', '2021-10-05 16:00:00.000000', 'De04', 'D04', 'Panic21'),

('34536344', '2021-10-05 19:00:00.000000', '2021-10-05 19:04:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De05', 'D04', 'Panic21'),

('86545865', '2021-10-05 19:00:00.000000', '2021-10-05 19:02:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De06', 'D06', 'Panic21'),

('28784343', '2021-10-05 19:00:00.000000', '2021-10-05 19:01:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De07', 'D08', 'Panic21'),

('233636', '2021-10-04 19:00:00.000000', '2021-10-04 19:05:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De08', 'D01', 'Panic21'),

('233636', '2021-10-05 19:07:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-05 19:06:00.000000', 'De09', 'D01', 'Panic21'),

('4567564', '2021-10-03 00:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-02 00:00:00.000000', 'De10', 'D01', 'Panic21'),

('4567564', '2021-10-04 00:00:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De11', 'D01', 'Panic21'),

('4567564', '2021-10-05 00:00:00.000000', '2021-10-06 19:30:00.000000', 'negative', '2021-10-04 16:00:00.000000', 'De12', 'D01', 'Panic21'),

('4567564', '2021-10-19 00:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'De14', 'D01', 'Panic21'),

('86545865', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De15', 'D01', 'Unknown'),

('28784343', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De16', 'D02', 'Unknown'),

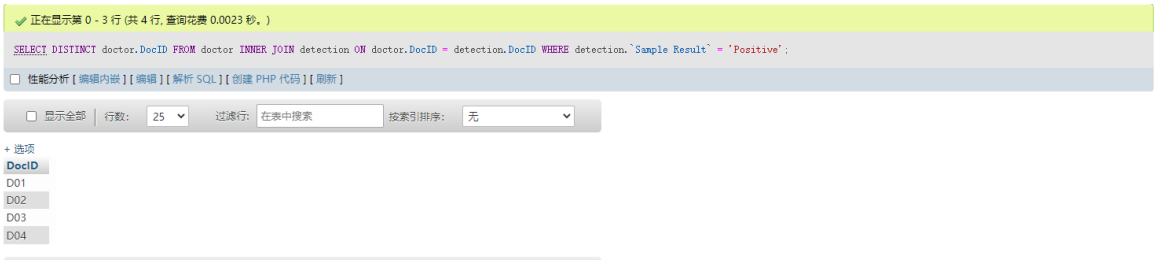
('5324342', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De17', 'D03', 'Unknown');

From those data above, we set doctors whose ID are D01, D02, D03, and D04 have tested positive case and doctor D06 and D08 didn't have detection of positive case.

Thus, we put the condition `Sample Result` = 'positive' in the where clause and we select distinct doctorID from doctor and detection table joined by inner join.

After seeing the data, we know there is positive cases detected by D01, D02,D03 and D04. Therefore, code would be right once it select those instead of some other doctorID.

The result of the SELECT statement (screenshot):



Use case 5:

Now the high risk-level area `Centre Lukewarm Hillside` has an outbreak of the disease. The government wants to select all the doctors who haven't done a detection in the low risk-level district to help the `Centre Lukewarm Hillside` district.

Please write a query to achieve that.

Your SQL statement:

SELECT DocID FROM

(SELECT doctor.`Doc Name`, hospital.Hospital_Name,doctor.DocID, district.RIskLevel FROM doctor INNER JOIN hospital INNER JOIN district ON

doctor.HospitalID = hospital.Hospital_ID AND hospital.DistrictID = district.DistrictID)

AS `new` WHERE RIskLevel = 'low'

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

Test data:

We first create a hospital in the low level area "Raspberry town" called "Raspberry town first hospital"

INSERT INTO `hospital`(`Hospital_ID`, `Hospital_Name`, `DistrictID`)

VALUES

("H05","Raspberry town first Hospital",'303')

Then we set one of the doctor in this hospital is "Mengduo" and his doctor ID is "D09" which makes sure there is a suitable object to select

INSERT INTO `doctor`(`DocID`, `Doc Name`, `HospitalID`)

VALUES ('D09','Mengduo','H05')

And the we create a table called 'new' that have all the doctors , corresponding hospital, and the district's risklevel

By using:

SELECT doctor.`Doc Name`, hospital.Hospital_Name,doctor.DocID, district.RIskLevel FROM doctor INNER JOIN hospital INNER JOIN district ON

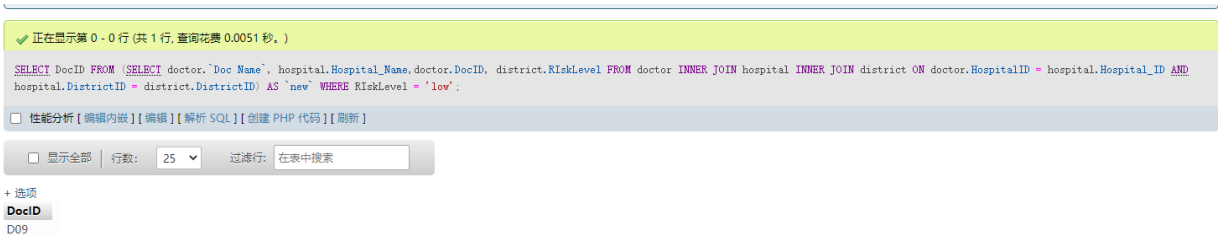
doctor.HospitalID = hospital.Hospital_ID AND hospital.DistrictID = district.DistrictID)

and we select the doctor ID from this table with constraints :" 'RIsklevel' = 'Low' "

The "Mengduo" is the only doctor that works in the hospital located in low risk-level are.

Therefore, the code would be right if it selects 'D09' (which is the doctor ID of Mengduo).

The result of the SELECT statement (screenshot):



Use case 6: From 2021-10-15 00:00 to 21-10-20 00:00, two of the base station of centre lukewarm hillside went broken down and causes some loss of the tracking information. Now, the government wants to find the Mobile ID of whom is not likely to go to there at the time the base station broke down.

Your SQL statement:

SELECT DISTINCT SimcardID FROM travelhistorytracking

WHERE SimCardConnectionTime BETWEEN '2021-10-15 00:00' AND '21-10-20 00:00'

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

We set the data that **only the last one** has a tracking records at that time.

INSERT INTO `travelhistorytracking` (`SimcardID`, `SimCardConnectionTime`, `SimCardDisconnectTime`, `TrackingID`, `BSID`) VALUES

('233636', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T01', '4041'),

('233636', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T02', '4046'),

('8906453', '2021-10-07 19:30:00.000000', '2021-10-08 19:30:00.000000', 'T03', '4041'),

('231425534', '2021-10-08 19:30:00.000000', '2021-10-09 19:30:00.000000', 'T04', '4045'),

('64253465', '2021-10-10 19:30:00.000000', '2021-10-10 19:31:00.000000', 'T05', '4045'),

('64253465', '2021-10-10 19:30:00.000000', '2021-10-10 19:31:00.000000', 'T06', '4045'),

('4567564', '2021-10-17 19:30:00.000000', '2021-10-18 19:30:00.000000', 'T07', '4045');

Then to solve this problem, we need to understand that the person didn't went to that place means he / she had tracking information at that time

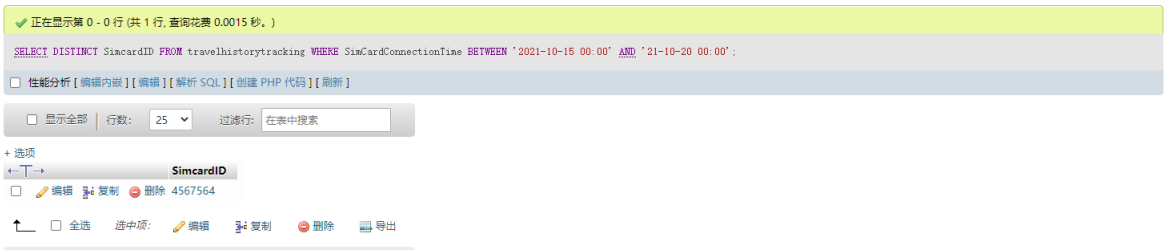
Thus, **we only need to select the Sim card ID which has a Sim card connection time (not null) at that time is okay.**

Therefore, we use the time interval as a condition for this select.

The "4567564" is the mobile ID of the person who is the only one that had a tracking history at that time

Therefore, the code would be right if it selects "4567564"

The result of the SELECT statement (screenshot):



Use case 7: The middle-aged man is way more vulnerable to fight against the disease. Please write a query the find out the number of all the patient whose age are above 40.

Your SQL statement:

SELECT COUNT(Name) FROM

(SELECT DISTINCT Name FROM

(SELECT MobileID, Name FROM `person` WHERE Age > 40) AS `new` INNER JOIN detection ON new.mobileID = detection.MobileID WHERE `Sample result` = 'positive')

AS `counter`

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

Test data: **we set only the MobileID = '202020' and '233636' 's person is the middle-aged person and mark is a patient but bren joy is not (till now)**

INSERT INTO `person` (`Name`, `MobileID`, `Age`, `Sex`) VALUES

('Cnangzhang', '+86 7777777', 16, 'male'),

('Bren joy', '202020', 70, 'male'),

('Powder', '231425534', 15, 'female'),

('Mark', '233636', 42, 'male'),

('Suga', '23652546', 27, 'male'),

('Melee', '28784343', 26, 'male'),

('Kiana', '34536344', 15, 'female'),

('Arene', '4567564', 18, 'male'),

('Caitlyn', '45675865', 25, 'Female'),

('Blemishine', '5324342', 18, 'female'),

('Carnelian', '64253465', 33, 'female'),

('Whisperain', '86534343', 23, 'female'),

('Indigo', '86545865', 20, 'female'),

('Vi', '87539454', 24, 'female'),

('Heavyrain', '8906453', 15, 'female');

INSERT INTO `detection` (`MobileID`, `Sample Test Time`, `Sample Report Time`, `Sample Result`, `Sample Collect Time`, `DetectionID`, `DocID`, `Sample Type`) VALUES

('23652546', '2021-10-19 19:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'D013', 'D03', 'Panic21'),

('233636', '2021-10-09 19:00:00.000000', '2021-10-09 19:30:00.000000', 'positive', '2021-10-09 16:00:00.000000', 'De01', 'D01', 'Panic21'),

('+86 7777777', '2021-10-03 02:00:00.000000', '2021-10-03 19:30:00.000000', 'negative', '2021-10-03 01:00:00.000000', 'De018', 'D01', 'Panic21'),

('+86 7777777', '2021-10-04 03:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-03 02:00:00.000000', 'De019', 'D01', 'Panic21'),

('87539454', '2021-10-04 19:00:00.000000', '2021-10-04 19:28:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De02', 'D01', 'Panic21'),

('86534343', '2021-10-03 19:00:00.000000', '2021-10-09 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De020', 'D02', 'Panic21'),

('86534343', '2021-10-03 19:05:00.000000', '2021-10-09 19:30:00.000000', 'negative', '2021-10-03 19:00:00.000000', 'De021', 'D02', 'Panic21'),

('45675865', '2021-10-04 19:00:00.000000', '2021-10-04 19:30:00.000000', 'positive', '2021-10-04 16:00:00.000000', 'De03', 'D03', 'Panic21'),

('231425534', '2021-10-05 19:00:00.000000', '2021-10-05 19:05:00.000000', 'positive', '2021-10-05 16:00:00.000000', 'De04', 'D04', 'Panic21'),

('34536344', '2021-10-05 19:00:00.000000', '2021-10-05 19:04:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De05', 'D04', 'Panic21'),

('86545865', '2021-10-05 19:00:00.000000', '2021-10-05 19:02:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De06', 'D06', 'Panic21'),

('28784343', '2021-10-05 19:00:00.000000', '2021-10-05 19:01:00.000000', 'negative', '2021-10-05 16:00:00.000000', 'De07', 'D08', 'Panic21'),

('233636', '2021-10-04 19:00:00.000000', '2021-10-04 19:05:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De08', 'D01', 'Panic21'),

('233636', '2021-10-05 19:07:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-05 19:06:00.000000', 'De09', 'D01', 'Panic21'),

('4567564', '2021-10-03 00:00:00.000000', '2021-10-04 19:30:00.000000', 'negative', '2021-10-02 00:00:00.000000', 'De10', 'D01', 'Panic21'),

('4567564', '2021-10-04 00:00:00.000000', '2021-10-05 19:30:00.000000', 'negative', '2021-10-03 16:00:00.000000', 'De11', 'D01', 'Panic21'),

('4567564', '2021-10-05 00:00:00.000000', '2021-10-06 19:30:00.000000', 'negative', '2021-10-04 16:00:00.000000', 'De12', 'D01', 'Panic21'),

('4567564', '2021-10-19 00:00:00.000000', '2021-10-19 19:30:00.000000', 'positive', '2021-10-19 16:00:00.000000', 'De14', 'D01', 'Panic21'),

('86545865', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De15', 'D01', 'Unknown'),

('28784343', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De16', 'D02', 'Unknown'),

('5324342', '2021-11-09 19:00:00.000000', '2021-11-09 19:30:00.000000', 'positive', '2021-11-09 16:00:00.000000', 'De17', 'D03', 'Unknown');

Why: we set there are 2 people aged above 40 and one of them are patient, thus if and only if the counter = 1 make sure the code is right

So, if the inner statement ((SELECT MobileID, Name FROM `person` WHERE Age > 40) AS `new`) is wrong then it will select the sample whose number is way more than 2

And if the statement ((SELECT DISTINCT Name FROM

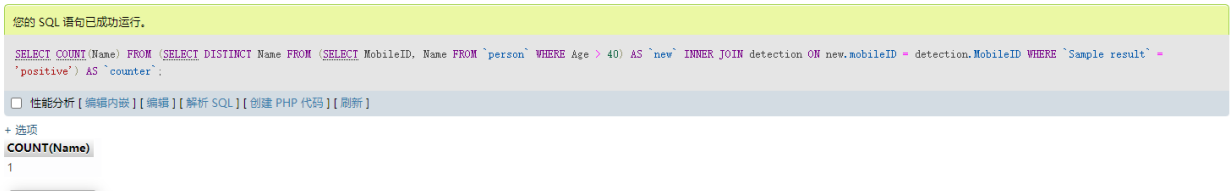
(SELECT MobileID, Name FROM `person` WHERE Age > 40) AS `new` INNER JOIN detection ON new.mobileID = detection.MobileID WHERE `Sample result` = 'positive')

) is wrong then the counter will collect mark more than 1 times for which the counter will be 2 in the end.

Mark is the only PATIENT whose age is above 40

Therefore, the code would be right if it selects the count equals 1.

The result of the SELECT statement (screenshot):



Use case 8:

The government wants to find if the virus is more harmful for the eld (>40) or the young (20<). Please write a query to find the answer.

Your SQL statement:

SELECT (SELECT COUNT(Name) FROM

(SELECT DISTINCT Name FROM

(SELECT MobileID, Name FROM `person` WHERE Age > 40) AS `new` INNER JOIN detection ON new.mobileID = detection.MobileID WHERE `Sample result` = 'positive')

AS `counter1`

) -(SELECT COUNT(Name) FROM

(SELECT DISTINCT Name FROM

(SELECT MobileID, Name FROM `person` WHERE Age < 30) AS `new` INNER JOIN detection ON new.mobileID = detection.MobileID WHERE `Sample result` = 'positive')

AS `counter2`

) AS `result`

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

Test data:

INSERT INTO `person` (`Name`, `MobileID`, `Age`, `Sex`) VALUES

('Cnangzhang', '+86 7777777', 16, 'male'),

('Bren joy', '202020', 70, 'male'),

('Powder', '231425534', 15, 'female'),

('Mark', '233636', 42, 'male'),

('Suga', '23652546', 27, 'male'),

('Melee', '28784343', 26, 'male'),

('Kiana', '34536344', 15, 'female'),

('Arene', '4567564', 18, 'male'),

('Caitlyn', '45675865', 25, 'Female'),

('Blemishine', '5324342', 18, 'female'),

('Carnelian', '64253465', 33, 'female'),

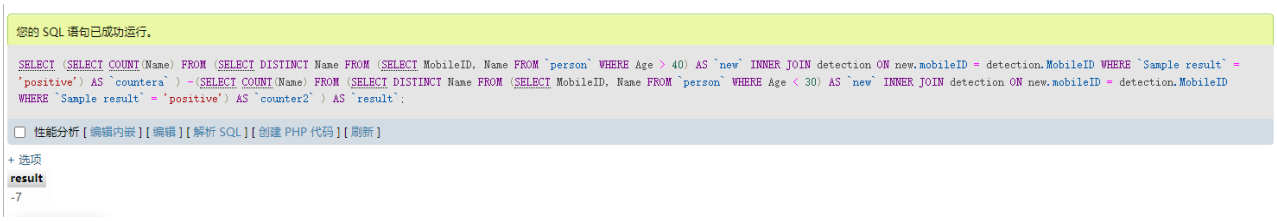
('Whisperain', '86534343', 23, 'female'),

('Indigo', '86545865', 20, 'female'),

('Vi', '87539454', 24, 'female'),

('Heavyrain', '8906453', 15, 'female');

The result of the SELECT statement (screenshot):



We set there are 2 eld one and 9 younger one according to the definition. As you can see, the upper subquery calculates the elder just like case 7 and the subquery below do the math for the younger.

Thus, if any part of the 2 calculation is wrong the sub will be wrong.

Therefore, the code is correct if it select -7.

Use case 9:

Recently, the Bunny Tail district has many new positive cases appearing in 2 days. Please write a query to update the risk level of the district.

Your SQL statement:

UPDATE `district` SET `RIskLevel`='high'WHERE `DistrictID` = '415'

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

Test data:

We set there are 4 other districts except “Raspberry town”

INSERT INTO `district` (`DistrictID`, `RIskLevel`, `District_name`, `Region`) VALUES

('213', 'high', 'Centre Lukewarm Hillside', 'central'),

('303', 'low', 'Raspberry town', 'west'),

('404', 'mid', 'Glow Sand district', 'central'),

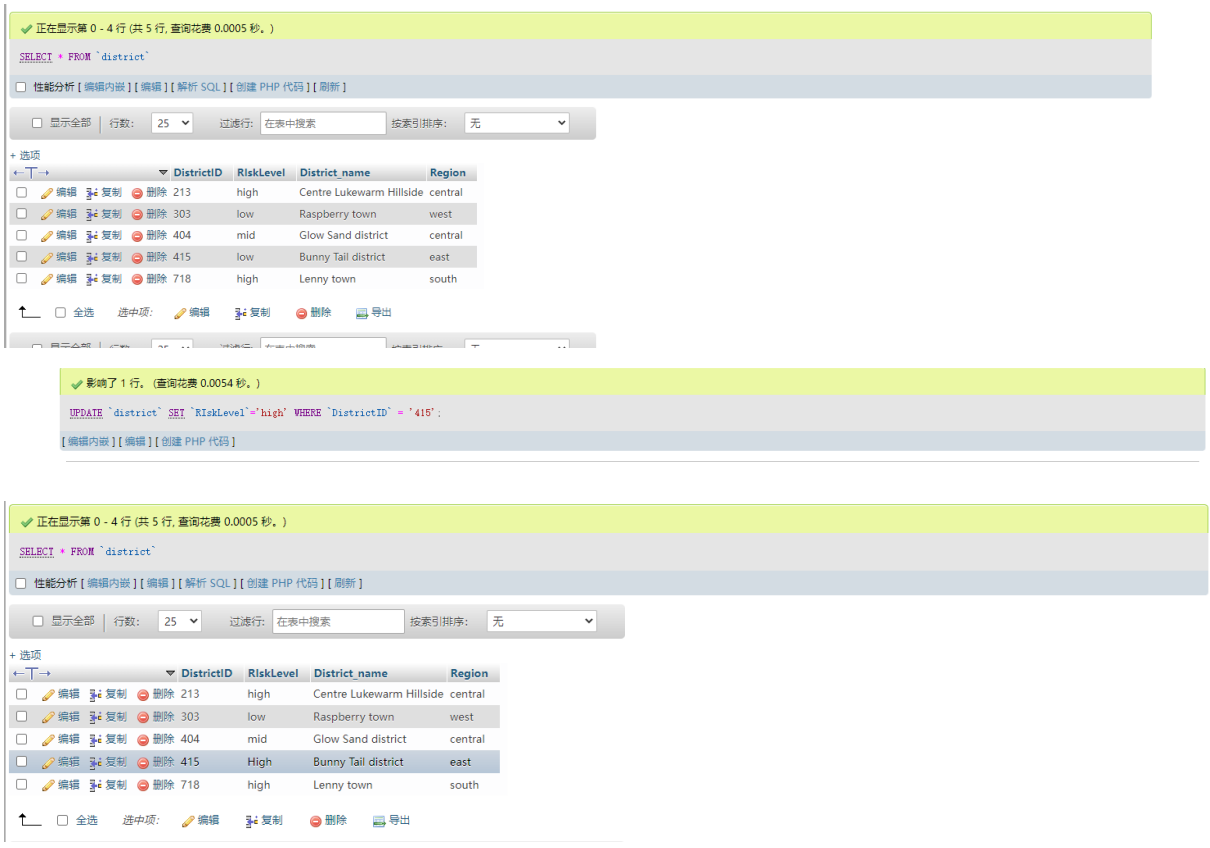
('415', 'low', 'Bunny Tail district', 'east'),

('718', 'high', 'Lenny town', 'south');

Why: **we only need to change the risk level of the district whose district ID is '415' so we can check the query once the other 4 rows changes after we run the code.**

Thus, the code is right if it turns the risk-level of the raspberry town into high.

The result of the SELECT statement (screenshot):



Use case 10: Now, there is a mistake of the detection of a person whose mobileID is 5324342. Please find a query to find the person's information immediately.

.

Your SQL statement:

SELECT * FROM `person` WHERE MobileID = '5324342' ;

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

INSERT INTO `person` (`Name`, `MobileID`, `Age`, `Sex`) VALUES

('Cnangzhang', '+86 7777777', 16, 'male'),

('Bren joy', '202020', 70, 'male'),

('Powder', '231425534', 15, 'female'),

('Mark', '233636', 42, 'male'),

('Suga', '23652546', 27, 'male'),

('Melee', '28784343', 26, 'male'),

('Kiana', '34536344', 15, 'female'),

('Arene', '4567564', 18, 'male'),

('Caitlyn', '45675865', 25, 'Female'),

('Blemishine', '5324342', 18, 'female'),

('Carnelian', '64253465', 33, 'female'),

('Whisperain', '86534343', 23, 'female'),

('Indigo', '86545865', 20, 'female'),

('Vi', '87539454', 24, 'female'),

('Heavyrain', '8906453', 15, 'female');

We set the person is the blemishine whose phone number is 5324342, thus we use a select * from person table so that we can get he/ she ‘s information from the data base.

The result of the SELECT statement (screenshot):