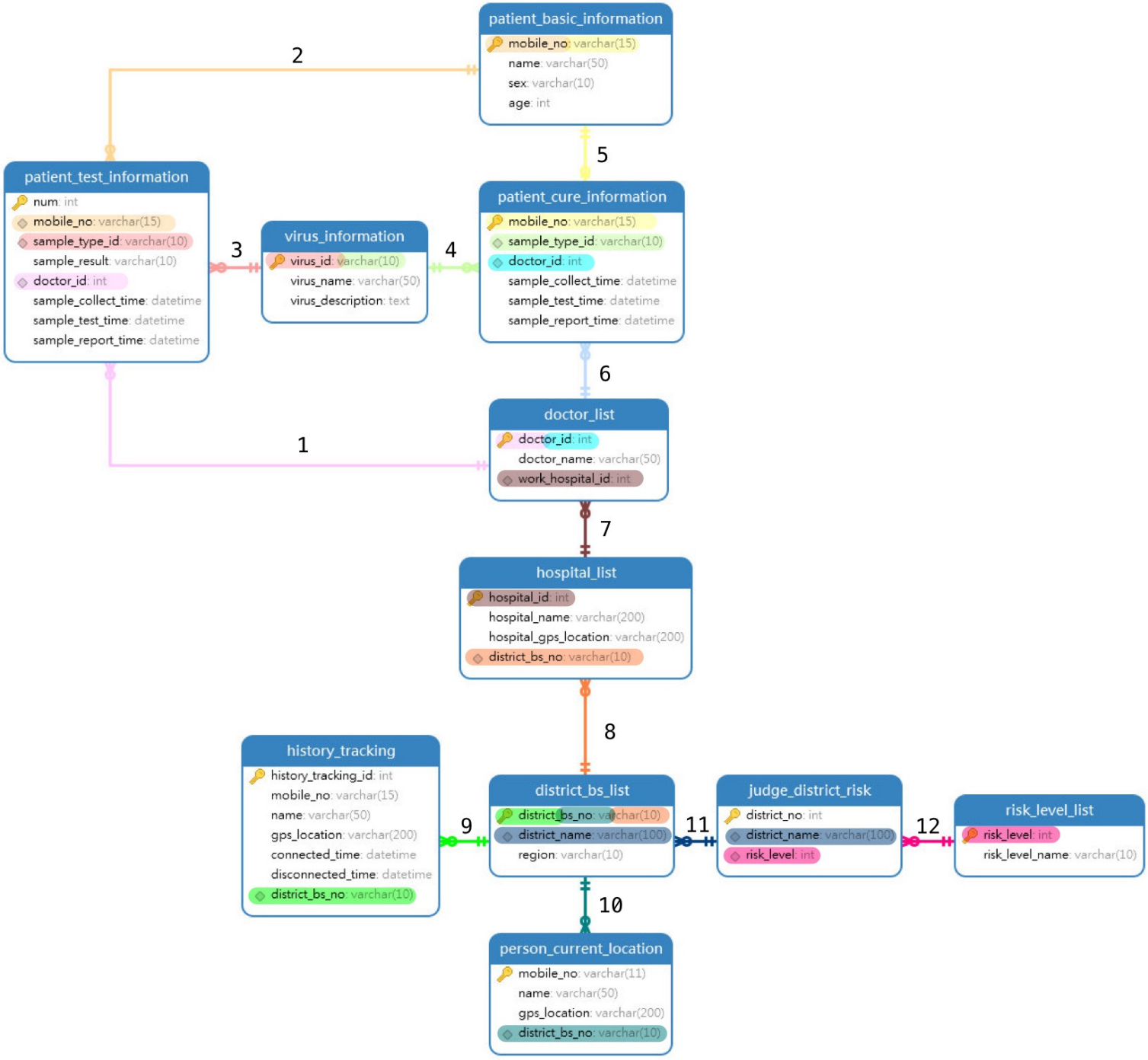
ER Diagram


patient_basic_information
mobile_no: varchar(15)
name: varchar(50)
sex: varchar(10)
age: int
2
5
patient_test_information
num: int
mobile_no: varchar(15)
sample_type_id: varchar(10)
sample_result: varchar(10)
doctor_id: int
sample_collect_time: datetime
sample_test_time: datetime
sample_report_time: datetime
3
virus_information
virus_id: varchar(10)
virus_name: varchar(50)
virus_description: text
4
patient_cure_information
mobile_no: varchar(15)
sample_type_id: varchar(10)
doctor_id: int
sample_collect_time: datetime
sample_test_time: datetime
sample_report_time: datetime
6
1
doctor_list
doctor_id: int
doctor_name: varchar(50)
work_hospital_id: int
7
hospital_list
hospital_id: int
hospital_name: varchar(200)
hospital_gps_location: varchar(200)
district_bs_no: varchar(10)
8
history_tracking
history_tracking_id: int
mobile_no: varchar(15)
name: varchar(50)
gps_location: varchar(200)
connected_time: datetime
disconnected_time: datetime
district_bs_no: varchar(10)
9
district_bs_list
district_bs_no: varchar(10)
district_name: varchar(100)
region: varchar(10)
11
judge_district_risk
district_no: int
district_name: varchar(100)
risk_level: int
12
risk_level_list
risk_level: int
risk_level_name: varchar(10)
10
person_current_location
mobile_no: varchar(11)
name: varchar(50)
gps_location: varchar(200)
district_bs_no: varchar(10)

Database Design Details

Tables

------ Detailed explanations of tables ------

Table 1

Table name: patient_basic_information

Table design explanation:

The Patient_basic_information is used to store the basic information related to people who have done the viral tests (not to record information of all people in the Lukewarm Kingdom). The primary key is mobile_no as it is unique for every patient. *[The aim of creating this table is to satisfy the 3NF design.]*

Attributes:

Column Definition	Domain	Explanation
mobile_no varchar(15)	The proper number for a telephone number is 11 digits. Mark's phone number (233636) is a special case.	This column is used to record the phone numbers of people who have done the viral tests (not all people's phone numbers in the Lukewarm Kingdom). The specified number of telephone numbers is 11 digits (the telephone number 233636 is given as the subject and not taken into account), which is randomly generated by the system without any special rules. However, considering that the phone number might be insufficient, the length of 'varchar' was extended to 15. The reason for using 'varchar' instead of 'int' is that the default length of 'int' is 11 digits. Using 'varchar' can prevent the occurrence of numbers that need to be larger than 11 digits for some reason. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by personal phone numbers not being recorded in the table.
name varchar(50)	All valid names will be accepted. The length of each name is limited to 50 characters. For example, the name 'Mark' is legal.	This column is used to record the names of people who have done the viral tests. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
sex varchar(10)	The gender categories are limited to three choices: male, female and bisexual. The length of each gender is limited to 10 characters.	This column is used to record the genders of people who have done the viral tests. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory. In order to simplify querying, the sample data added in this report does not consider bisexual people.

age int(5)	The age range is 0 to 200. And this number cannot exceed 5 digits.	This column is used to record the ages of people who have done the viral tests. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.

Foreign keys and reasons:

No foreign keys are set in this table.

Table 2

Table name: patient_test_information

Table design explanation:

The 'patient_test_information' is used to store the test information related to people who have done the viral tests (not to record information of all people in the Lukewarm Kingdom). The primary key is 'num' as it is unique for every test recording. *[The aim of creating this table is to satisfy most of the information in the given sample report (Figure 1 in the task sheet).]*

Attributes:

Column Definition	Domain	Explanation
num int(11)	The proper number for 'num' is 11 digits.	This column is used to record the quantity of viral tests report. The reason for choosing 'int(11)' is that this column is just a series of digits and the number '11' is the default value of 'int'. This design is very similar to the serial number in banks because 'num' is unique for every viral test. And 'num' itself does not have any special meaning. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by 'num' not being recorded in this table. Especially, 'num' is also set to AUTO_INCREMENT when the database administrator adds new recordings into this table.
mobile_no varchar(15)	The proper number for a telephone number is 11 digits. Mark's phone number (233636) is a special case.	This column is used to record the phone numbers of people who have done the viral tests (not all people's phone numbers in the Lukewarm Kingdom). The specified number of telephone numbers is 11 digits (the telephone number 233636 is given as the subject and not taken into account), which is randomly generated by the system without any special rules. However, considering that the phone number might be insufficient, the length of 'varchar' was extended to 15. The reason for using 'varchar' instead of 'int' is that the default length of 'int' is 11 digits. Using 'varchar' can prevent the occurrence of numbers that need to be larger than 11 digits for some reason. This column serves as the foreign key to reference 'mobile_no' in the 'patient_basic_information' table and is set to NOT

		NULL to prevent information loss caused by personal phone numbers not being recorded in the table. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (patient_basic_information), check whether the record has a foreign key. If so, the foreign key records in the child table (patient_test_information) are also deleted. It is convenient for the administrator to maintain the database in the future.
sample_type_id varchar(10)	The sample_type_id is made up of 'virus' and 'number'. For example, 'virus1' is valid. The length of each sample_type_id is limited to 10 characters.	This column is used to record the type of virus people choose to test for. The aim of choosing varchar(10) instead of int(11) is that this attribute needs to be distinguished from 'num' (in this table) to avoid unnecessary confusion. And the domain of the 'number' part (five characters left except the word 'virus') is enough to record different kinds of viruses. This is a foreign key that references 'virus_id' in the 'virus_information' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (virus_information), check whether the record has a foreign key. If so, the foreign key records in the child table (patient_test_information) are also deleted. It is convenient for the administrator to maintain the database in the future.
sample_result varchar(10)	The sample_result categories are limited to two choices: positive, negative. The length of each sample_result is limited to 10 characters.	This column is used to record the final virus test results. 'varchar(10)' is chosen as the data type. The words 'positive' and 'negative' are 8 characters long, but considering the special cases, 10 characters are chosen to ensure that no errors occur. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
doctor_id int(11)	Doctors' id numbers are made up of a string of 10 digits. For example, one doctor's name is 'Halbert'. His id number is '1100600101'.	This column is used to record each doctor's id number. The reason for choosing 'int(11)' is that this column is just a series of digits and the number '11' is the default value of 'int'. This string of numbers is not randomly generated by computers, but a certain naming rule. Take '1100600101' as an example, '11' (east) represents the region serial number, '006' (Godlyland) represents district serial number, '001' (hospital1) represents hospital serial number, '01' represents personal id. This is a foreign key that references 'doctor_id' in the 'doctor_list' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting

		a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (doctor_list), check whether the record has a foreign key. If so, the foreign key records in the child table (patient_test_information) are also deleted. It is convenient for the administrator to maintain the database in the future.
sample_collect_time datetime	There is no fixed length limit for 'sample_collect_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'.	This column is used to record each patient's sample collection time. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
sample_test_time datetime	There is no fixed length limit for 'sample_test_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each patient's sample test time. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
sample_report_time datetime	There is no fixed length limit for 'sample_report_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each patient's sample report time. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.

Foreign keys and reasons:

The column 'mobile_no' references 'mobile_no' in the 'patient_basic_information' table, this is to make sure that the administrator can connect the 'patient_basic_information' table through the phone number to find each patient's specific name, gender and age. It corresponds to the number '2' in the ER diagram.

The column 'sample_type_id' references 'virus_id' in the 'virus_information' table, this is to make sure that the administrator can connect the 'virus_information' table through the 'virus_id' to find each virus's specific name and description. It corresponds to the number '3' in the ER diagram.

The column 'doctor_id' references 'doctor_id' in the 'doctor_list' table, this is to make sure that the administrator can connect the 'doctor_list' table through the 'doctor_id' to find each doctor's specific id, name, and workplace. It corresponds to the number '1' in the ER diagram.

Table 3

Table name: patient_cure_information

Table design explanation:

The 'patient_cure_information' is used to store the test information about patients who tested 'positive' in the 'patient_test_information' showed 'negative' for the first time after a period of treatment (test

results during treatment are not included). The primary key is ‘num’ as it is unique for every test recording. *[The aim of creating this table is to help finish extended cases.]*

Attributes:

Column Definition	Domain	Explanation
mobile_no varchar(15)	The proper number for a telephone number is 11 digits. Mark's phone number (233636) is a special case.	This column is used to record the phone numbers of people who have done the viral tests (not all people’s phone numbers in the Lukewarm Kingdom). The specified number of telephone numbers is 11 digits (the telephone number 233636 is given as the subject and not taken into account), which is randomly generated by the system without any special rules. However, considering that the phone number might be insufficient, the length of ‘varchar’ was extended to 15. The reason for using ‘varchar’ instead of ‘int’ is that the default length of ‘int’ is 11 digits. Using ‘varchar’ can prevent the occurrence of numbers that need to be larger than 11 digits for some reason. This column serves as the foreign key to reference ‘mobile_no’ in the ‘patient_basic_information’ table and is set to NOT NULL to prevent information loss caused by personal phone numbers not being recorded in the table. Besides, when setting a foreign key, ‘on delete’ and ‘on update’ should be set CASCADE. That is, when deleting a record from the parent table (patient_basic_information), check whether the record has a foreign key. If so, the foreign key records in the child table (patient_test_information) are also deleted. It is convenient for the administrator to maintain the database in the future.
sample_type_id varchar(10)	The sample_type_id is made up of ‘virus’ and ‘number’. For example, ‘virus1’ is valid. The length of each sample_type_id is limited to 10 characters.	This column is used to record the type of virus people choose to test for. The aim of choosing varchar(10) instead of int(11) is that this attribute needs to be distinguished from ‘num’ (in table 2) to avoid unnecessary confusion. And the domain of the ‘number’ part (five characters left except the word ‘virus’) is enough to record different kinds of viruses. This is a foreign key that references ‘virus_id’ in the ‘virus_information’ table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, ‘on delete’ and ‘on update’ should be set CASCADE. That is, when deleting a record from the parent table (virus_information), check whether the record has a foreign key. If so, the foreign key records in the child table (patient_test_information) are also

		deleted. It is convenient for the administrator to maintain the database in the future.
doctor_id int(11)	Doctors' id numbers are made up of a string of 10 digits. For example, one doctor's name is 'Halbert'. His id number is '1100600101'.	This column is used to record each doctor's id number. The reason for choosing 'int(11)' is that this column is just a series of digits and the number '11' is the default value of 'int'. This string of numbers is not randomly generated by computers, but a certain naming rule. Take '1100600101' as an example, '11' (east) represents the region serial number, '006' (Godlyland) represents district serial number, '001' (hospital1) represents hospital serial number, '01' represents personal id. This is a foreign key that references 'doctor_id' in the 'doctor_list' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (doctor_list), check whether the record has a foreign key. If so, the foreign key records in the child table (patient_test_information) are also deleted. It is convenient for the administrator to maintain the database in the future.
sample_collect_time datetime	There is no fixed length limit for 'sample_collect_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each patient's sample collection time. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
sample_test_time datetime	There is no fixed length limit for 'sample_test_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each patient's sample test time. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
sample_report_time datetime	There is no fixed length limit for 'sample_report_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each patient's sample report time. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.

Foreign keys and reasons:

The column ‘mobile_no’ references ‘mobile_no’ in the ‘patient_basic_information’ table, this is to make sure that the administrator can connect the ‘patient_basic_information’ table through the phone number to find each patient’s specific name, gender and age. It corresponds to the number ‘5’ in the ER diagram.

The column ‘sample_type_id’ references ‘virus_id’ in the ‘virus_information’ table, this is to make sure that the administrator can connect the ‘virus_information’ table through the ‘virus_id’ to find each virus’s specific name and description. It corresponds to the number ‘4’ relationship in the ER diagram.

The column ‘doctor_id’ references ‘doctor_id’ in the ‘doctor_list’ table, this is to make sure that the administrator can connect the ‘doctor_list’ table through the ‘doctor_id’ to find each doctor’s specific id, name, and workplace. It corresponds to the ‘6’ in the ER diagram.

Table 4

Table name: doctor_list

Table design explanation:

The ‘doctor_list’ is used to store the test information about doctors in different hospitals. The primary key is ‘doctor_id’ as it is unique for every doctor. *[The aim of creating this table is to satisfy the 3NF design.]*

Attributes:

Column Definition	Domain	Explanation
doctor_id int(11)	Doctors' id numbers are made up of a string of 10 digits. For example, one doctor’s name is ‘Halbert’. His id number is ‘1100600101’.	This column is used to record each doctor’s id number. The reason for choosing ‘int(11)’ is that this column is just a series of digits and the number ‘11’ is the default value of ‘int’. This string of numbers is not randomly generated by computers, but a certain naming rule. Take ‘1100600101’ as an example, ‘11’ (east) represents the region number, ‘006’ (Godlyland) represents district number, ‘001’ (hospital1) represents hospital number, ‘01’ represents personal id. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by ‘doctor_id’ not being recorded in this table.
name varchar(50)	All valid names will be accepted. The length of each name is limited to 50 characters. For example, the name ‘Elliott’ is legal.	This column is used to record the names of doctors in different hospitals. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
work_hospital_id int(11)	The proper serial number for ‘work_hospital_id’ is no more than 11 digits.	This column is used to record the hospital’s number where each doctor works. The reason for choosing ‘int(11)’ is that this column is just a series of digits and the number ‘11’ is the default value of ‘int’. And ‘work_hospital_id’ itself does not have any special meaning. This is a foreign key that references ‘hospital_id’ in the ‘hospital_list’ table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, ‘on delete’ and ‘on update’ should be set CASCADE. That is, when deleting a record from the parent table (hospital_list), check whether the record has a

		foreign key. If so, the foreign key records in the child table (doctor_list) are also deleted. It is convenient for the administrator to maintain the database in the future.

Foreign keys and reasons:

The column 'work_hospital_id' references 'hospital_id' in the 'hospital_list' table, this is to make sure that the administrator can connect the 'hospital_list' table through the hospital's serial number to find each hospital's specific name, GPS location and the base station (BS) where each hospital is located. It corresponds to the number '7' in the ER diagram.

Table 5

Table name: hospital_list

Table design explanation:

The 'hospital_list' is used to store the test information about hospitals. The primary key is 'hospital_id' as it is unique for every hospital. *[The aim of creating this table is to satisfy the 3NF design.]*

Attributes:

Column Definition	Domain	Explanation
hospital_id int(11)	The proper serial number for 'hospital_id' is no more than 11 digits.	This column is used to record the hospital's serial number. The reason for choosing 'int(11)' is that this column is just a series of digits and the number '11' is the default value of 'int'. And 'hospital_id' itself does not have any special meaning. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by 'hospital_id' not being recorded in this table.
hospital_name varchar(200)	The length of each hospital name is limited to 200 characters. For example, one hospital's name is 'hospital1'.	This column is used to record the names of different hospitals. The reason for choosing 'varchar(200)' is that this column is made up of the word 'hospital' and one number (1, 2, 3…). And 'varchar(200)' has enough space to store this number if needed. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
hospital_gps_location varchar(200)	The length of each GPS location is limited to 200 characters.	This column is used to record hospitals' GPS locations. The reason for choosing 'varchar(200)' is that this column is very long. And its format is strictly defined: $GPGGA, <UTC>, <latitude>, <northern latitude/south latitude>, <longitude>, <east longitude/west longitude>, <positioning quality indicators>, <number of satellites used>, <horizontal precision>, <the height of the antenna from sea level>, <the unit of height>, <the height of the earth ellipsoid relative to sea level>, <the unit of height>, <differential GPS data duration>, <differential reference base station number>, <hh> (the GPS format will be explained later in the use case 2). This attribute is greatly useful to help complete the calculation about distance. This

		column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
district_bs_no varchar(10)	The length of each base station's (BS) serial number is limited to 10 characters.	This column is used to record serial numbers of base stations (BS) where the hospital is located. The reason for choosing 'varchar(10)' is that this column is only made up of one capital letter (C/E/W/S/N) and one number (1, 2, 3…). And the capital letter only has 5 choices, that is 'C', 'E', 'W', 'S' and 'N'. This attribute is greatly useful to help complete the calculation about distance. This is a foreign key that references 'district_bs_no' in the 'district_bs_list' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (district_bs_list), check whether the record has a foreign key. If so, the foreign key records in the child table (hospital_list) are also deleted. It is convenient for the administrator to maintain the database in the future.

Foreign keys and reasons:

The column 'district_bs_no' references 'district_bs_no' in the 'district_bs_list' table, this is to make sure that the administrator can connect the 'district_bs_list' table through the base station's (BS) serial number to find each base station's (BS) specific name and region (center/east/west/south/north). It corresponds to the number '8' in the ER diagram.

Table 6

Table name: district_bs_list

Table design explanation:

The 'district_bs_list' is used to store the test information about base stations (BS). The primary key is 'district_bs_no' as it is unique for every base station (BS). *[The aim of creating this table is to satisfy the 3NF design.]*

Attributes:

Column Definition	Domain	Explanation
district_bs_no varchar(10)	The length of each base station's (BS) serial number is limited to 10 characters.	This column is used to record serial numbers of base stations (BS) where the hospital is located. The reason for choosing 'varchar(10)' is that this column is only made up of one capital letter (C/E/W/S/N) and one number (1, 2, 3…). And the capital letter only has 5 choices, that is 'C', 'E', 'W', 'S' and 'N'. This attribute is greatly useful to help complete the calculation about distance. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by 'district_bs_no' not being recorded in this table.
district_name varchar(100)	The length of each hospital name is limited to 100 characters.	This column is used to record the names of different districts. And 'varchar(100)' has enough space to store names. This is a foreign key that

		references 'district_name' in the 'judge_district_risk' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (judge_district_risk), check whether the record has a foreign key. If so, the foreign key records in the child table (district_bs_list) are also deleted. It is convenient for the administrator to maintain the database in the future.
region varchar(10)	The region categories are limited to five choices: center, east, west, south and north. The length of each region is limited to 10 characters.	This column is used to record regions where base stations (BS) are located. The reason for choosing 'varchar(10)' is to ensure that each region string has enough room to store. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.

Foreign keys and reasons:

The column 'district_name' references 'district_name' in the 'judge_district_risk' table, this is to make sure that the administrator can connect the 'judge_district_risk' table through the district's name to find each district's specific name and risk level. It corresponds to the number '11' in the ER diagram.

Table 7

Table name: judge_district_risk

Table design explanation:

The 'judge_district_risk' is used to store the test information about district risk level. The primary key is 'district_no' as it is unique for every district. *[The aim of creating this table is to satisfy the 3NF design.]*

Attributes:

Column Definition	Domain	Explanation
district_no int(10)	The length of each 'district_no' is limited to 10 digits.	This column is used to record the serial numbers of different districts. The reason for choosing 'int(10)' is to leave enough space to store each district's serial number. And 'district_no' itself does not have any special meaning. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by 'district_no' not being recorded in this table. Especially, 'district_no' is also set to AUTO_INCREMENT when the database administrator adds new recordings into this table.
district_name varchar(100)	The length of each hospital name is limited to 100 characters.	This column is used to record the names of different districts. And 'varchar(100)' has enough space to store names. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
risk_level int(5)	The 'risk_level' categories are limited to three choices: 1, 2 and	This column is used to record each district's risk level. The reason for choosing 'int(5)' is that this column is just a series of digits. This string of

	3. The length of each number is limited to 5 digits.	numbers is not randomly generated by computers, but a certain naming rule. '1' represents the low-risk level, '2' represents the middle-risk level, '3' represents the high-risk level. This is a foreign key that references 'risk_level' in the 'risk_level_list' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (risk_level_list), check whether the record has a foreign key. If so, the foreign key records in the child table (judge_district_risk) are also deleted. It is convenient for the administrator to maintain the database in the future.

Foreign keys and reasons:

The column 'risk_level' references 'risk_level' in the 'risk_level_list' table, this is to make sure that the administrator can connect the 'risk_level_list' table through the district's name to help finish ordering in the following use case. It corresponds to the number '12' in the ER diagram.

Table 8

Table name: risk_level_list

Table design explanation:

The 'risk_level_list' is used to store the test information about district risk level. The primary key is 'risk_level' as it is unique for every level. *[The aim of creating this table is to satisfy the 3NF design and to complete related use cases.]*

Attributes:

Column Definition	Domain	Explanation
risk_level int(5)	The 'risk_level' categories are limited to three choices: 1, 2 and 3. The length of each number is limited to 5 digits.	This column is used to record each district's risk level. The reason for choosing 'int(5)' is that this column is just a series of digits. This string of numbers is not randomly generated by computers, but a certain naming rule. '1' represents the low-risk level, '2' represents the middle-risk level, '3' represents the high-risk level. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by 'district_no' not being recorded in this table.
risk_level_name varchar(100)	The length of each 'risk_level_name' is limited to 100 characters.	This column is used to record the name of each risk level (low/mid/high). And 'varchar(100)' has enough space to store names. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.

Foreign keys and reasons:

No foreign keys are set in this table.

Table 9

Table name: history_tracking

Table design explanation:

The 'history_tracking' is used to store the information about people's history tracking (all people in the Lukewarm Kingdom). The primary key is 'history_tracking_id' as it is unique for every history tracking. *[The aim of creating this table is to realize real-time tracking of everyone's location information. When a serious epidemic prevails in some place, the database administrator can use this table to find the travel records of all positive patients. Additionally, this table can record history tracking about one person repeatedly.]*

Attributes:

Column Definition	Domain	Explanation
history_tracking_id int(11)	The length of each 'history_tracking_id' is limited to 11 numbers.	This column is used to record the quantity of history tracking. The reason for choosing 'int(11)' is that this column is just a series of digits and the number '11' is the default value of 'int'. And 'history_tracking_id' itself does not have any special meaning. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by 'history_tracking_id' not being recorded in this table. Especially, 'history_tracking_id' is also set to AUTO_INCREMENT when the database administrator adds new recordings into this table.
mobile_no varchar(15)	The proper number for a telephone number is 11 digits. Mark's phone number (233636) is a special case.	This column is used to record the phone numbers of all people in the Lukewarm Kingdom. The specified number of telephone numbers is 11 digits (the telephone number 233636 is given as the subject and not taken into account), which is randomly generated by the system without any special rules. However, considering that the phone number might be insufficient, the length of 'varchar' was extended to 15. The reason for using 'varchar' instead of 'int' is that the default length of 'int' is 11 digits. Using 'varchar' can prevent the occurrence of numbers that need to be larger than 11 digits for some reason. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
name varchar(50)	All valid names will be accepted. The length of each name is limited to 50 characters. For example, the name 'Mark' is legal.	This column is used to record the names of all people in the Lukewarm Kingdom. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
gps_location varchar(200)	The length of each GPS location is limited to 200 characters.	This column is used to people's GPS locations. The reason for choosing 'varchar(200)' is that this column is very long. And its format is strictly defined: $GPGGA, <UTC>, <latitude>, <northern

		latitude/south latitude>, <longitude>, <east longitude/west longitude>, <positioning quality indicators>, <number of satellites used>, <horizontal precision>, <the height of the antenna from sea level>, <the unit of height>, <the height of the earth ellipsoid relative to sea level>, <the unit of height>, <differential GPS data duration>, <differential reference base station number>, <hh> (the GPS format will be explained later in the use case 2). This attribute is greatly useful to help complete the calculation about distance. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
connected_time datetime	There is no fixed length limit for 'connected_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each person's 'connected time'. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
disconnected_time datetime	There is no fixed length limit for 'disconnected_time'. The default datetime format is 'YYYY-MM-DD HH:MM:SS'. If no data is entered in this field, set the default value to '1999-09-09 00:00:00'.	This column is used to record each person's 'disconnected time'. The reason for choosing 'datetime' is that this column will be used to obtain accurate time and meet the calculation requirements for the date in the future. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
district_bs_no varchar(10)	The length of each base station's (BS) serial number is limited to 10 characters.	This column is used to record serial numbers of base stations (BS) where the person is located. The reason for choosing 'varchar(10)' is that this column is only made up of one capital letter (C/E/W/S/N) and one number (1, 2, 3…). And the capital letter only has 5 choices, that is 'C', 'E', 'W', 'S' and 'N'. This attribute is greatly useful to help complete the calculation about distance. This is a foreign key that references 'district_bs_no' in the 'district_bs_list' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (district_bs_list), check whether the record has a foreign key. If so, the foreign key records in the child table (history_tracking) are also deleted. It is convenient for the administrator to maintain the database in the future.

Foreign keys and reasons:

The column ‘district_bs_no’ references ‘district_bs_no’ in the ‘district_bs_list’ table, this is to make sure that the administrator can connect the ‘district_bs_list’ table through the base station’s (BS) serial number to find each base station’s (BS) specific name and region (center/east/west/south/north). It corresponds to the number ‘9’ in the ER diagram.

Table 10

Table name: person_current_location

Table design explanation:

The ‘person_current_location’ is used to store information about people’s current location (all people in the Lukewarm Kingdom). The primary key is ‘mobile_no’ as it is unique for every person. *[The aim of creating this table is to record current tracking of everyone's location information.]*

Attributes:

Column Definition	Domain	Explanation
mobile_no varchar(15)	The proper number for a telephone number is 11 digits. Mark's phone number (233636) is a special case.	This column is used to record the phone numbers of all people in the Lukewarm Kingdom. The specified number of telephone numbers is 11 digits (the telephone number 233636 is given as the subject and not taken into account), which is randomly generated by the system without any special rules. However, considering that the phone number might be insufficient, the length of ‘varchar’ was extended to 15. The reason for using ‘varchar’ instead of ‘int’ is that the default length of ‘int’ is 11 digits. Using ‘varchar’ can prevent the occurrence of numbers that need to be larger than 11 digits for some reason. This column serves as the primary key and is set to NOT NULL to prevent information loss caused by ‘mobile_no’ not being recorded in this table. Additionally, it is mandatory.
name varchar(50)	All valid names will be accepted. The length of each name is limited to 50 characters. For example, the name ‘Mark’ is legal.	This column is used to record the names of all people in the Lukewarm Kingdom. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
gps_location varchar(200)	The length of each GPS location is limited to 200 characters.	This column is used to people’s GPS locations. The reason for choosing ‘varchar(200)’ is that this column is very long. And its format is strictly defined: $GPGGA, <UTC>, <latitude>, <northern latitude/south latitude>, <longitude>, <east longitude/west longitude>, <positioning quality indicators>, <number of satellites used>, <horizontal precision>, <the height of the antenna from sea level>, <the unit of height>, <the height of the earth ellipsoid relative to sea level>, <the unit of height>, <differential GPS data duration>,

		<differential reference base station number>, <hh> (the GPS format will be explained later in the use case 2). This attribute is greatly useful to help complete the calculation about distance. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
district_bs_no varchar(10)	The length of each base station's (BS) serial number is limited to 10 characters.	This column is used to record serial numbers of base stations (BS) where the person is located. The reason for choosing 'varchar(10)' is that this column is only made up of one capital letter (C/E/W/S/N) and one number (1, 2, 3…). And the capital letter only has 5 choices, that is 'C', 'E', 'W', 'S' and 'N'. This attribute is greatly useful to help complete the calculation about distance. This is a foreign key that references 'district_bs_no' in the 'district_bs_list' table and is set to NOT NULL. Additionally, it is mandatory. Besides, when setting a foreign key, 'on delete' and 'on update' should be set CASCADE. That is, when deleting a record from the parent table (district_bs_list), check whether the record has a foreign key. If so, the foreign key records in the child table (person_current_location) are also deleted. It is convenient for the administrator to maintain the database in the future.

Foreign keys and reasons:

The column 'district_bs_no' references 'district_bs_no' in the 'district_bs_list' table, this is to make sure that the administrator can connect the 'district_bs_list' table through the base station's (BS) serial number to find each base station's (BS) specific name and region (center/east/west/south/north). It corresponds to the number '10' in the ER diagram.

Table 11

Table name: virus_information

Table design explanation:

The 'virus_information' is used to store information about each kind of virus. The primary key is 'virus_id' as it is unique for every virus. *[The aim of creating this table is to satisfy the 3NF design.]*

Attributes:

Column Definition	Domain	Explanation
virus_id varchar(10)	The sample_type_id is made up of 'virus' and 'number'. For example, 'virus1' is valid. The length of each sample_type_id is limited to 10 characters.	This column is used to record the serial numbers of different viruses. The aim of choosing varchar(10) instead of int(11) is that this attribute needs to be distinguished from 'num' (in table 2) to avoid unnecessary confusion. And the domain of the 'number' part (five characters left except the word 'virus') is enough to record different kinds of viruses. This column serves as the primary key and is set to NOT NULL to prevent information loss

		caused by 'virus_id' not being recorded in this table.
virus_name varchar(50)	The length of each 'risk_level_name' is limited to 100 characters.	This column is used to record the names of different viruses. And 'varchar(50)' has enough space to store names. The name is made up of the word 'virus' and one number. For example, 'Coughid-21' is valid. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.
virus_description text	This 'virus description' does not have character limitations.	This column is used to record descriptions of different viruses. And 'text' has enough space to store names. This column is a nonprime attribute and is set to NOT NULL. Additionally, it is mandatory.

Foreign keys and reasons:

No foreign keys are set in this table.

utes: mo

obile_no, doctor_name).

For **(mobile_no, doctor_name) -> hospital**, there is a partial function dependency of the non-primary attribute **hospital** on the key **(mobile_no, doctor_name)**.

Normalised tables and which normal form they are currently in:

After modified, currently in 2NF

1. Table 1:

mobile_no	name	sex	age

2. Table 2

mobile_no	sample_type	sample_result	doctor_name	sample_collection_time	sample_test_time	sample_report_time	hospital	virus_description

Stage 2

Attributes:

Primary attributes: mobile_no, doctor_name.

No Primary attributes: name, sex, age, sample_type, sample_result, virus_description, hospital, sample_collection_time, sample_test_time, sample_report_time.

FDs (Indicate partial or transitive dependencies):

Transitive dependence:

For Table 2, the primary key is **mobile_no**, the primary attribute is **mobile_no**, and the non-primary attributes are **doctor_name** and **hospital**. Because **doctor_name** is dependent on **mobile_no** and also **hospital** is dependent on **doctor_name**, there is a transfer function dependency of the non-primary

attribute **hospital** for the key **mobile_no**, so the design of Table 2, does not meet the requirements of 3NF.

For Table 2, the primary key is **mobile_no**, the primary attribute is **mobile_no**, and the non-primary attributes are **sample_type** and **virus_description**. Because **sample_type** is dependent on **mobile_no** and also **virus_description** is dependent on **sample_type**, there is a transfer function dependency of the non-primary attribute **virus_description** for the key **mobile_no**, so the design of Table 2, does not meet the requirements of 3NF.

Normalised tables and which normal form they are currently in:

After modified, currently in 3NF

1. Table 1:

mobile_no	name	sex	age

2. Table 2

mobile_no	sample_type_id	sample_result	doctor_id	sample_collection_time	sample_test_time	sample_report_time

3. Table 3

sample_type_id	sample_type_name	virus_description

4. Table 4

doctor_id	doctor_name	hospital_name

Use Cases

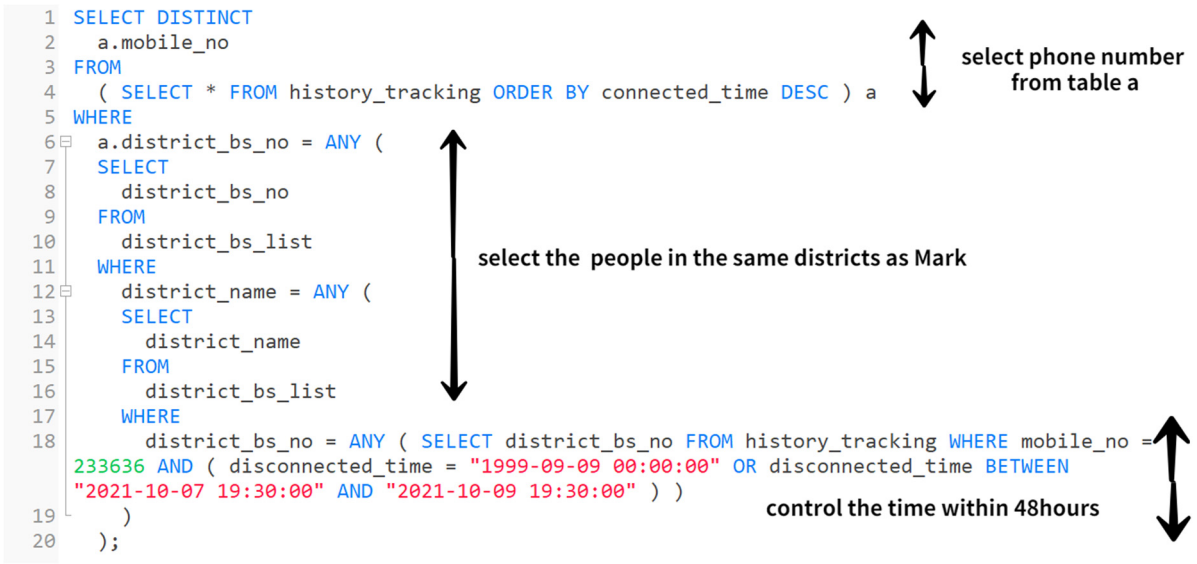
Important Use Cases

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

- SQL statement:

```
select
    distinct a.mobile_no
from
    (select * from history_tracking order by connected_time desc) a
where
    a.district_bs_no = any(
    select
        district_bs_no from district_bs_list
    where
        district_name = any(
        select
```

```
            district_name
        from
            district_bs_list
        where
            district_bs_no = any(
            select
                district_bs_no
            from
                history_tracking
            where
                mobile_no = 233636
            and (disconnected_time = "1999-09-09 00:00:00" or
disconnected_time between "2021-10-07 19:30:00" and "2021-10-09 19:30:00")
            )
        )
    );
```



➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	name	connected_time	disconnected_time	district_bs_no
13252155887	Wuxiangzhibing	2021-10-07 21:34:26	1999-09-09 00:00:00	C1
233636	Mark	2021-10-09 19:00:00	1999-09-09 00:00:00	C1
72530750739	Leishilaimu	2021-10-04 15:12:46	1999-09-09 00:00:00	C6
47029460746	Xifenglang	2021-10-03 02:02:02	1999-09-09 00:00:00	C5
233636	Mark	2021-10-01 12:34:23	2021-10-09 19:00:00	C5
1763059086	Leiniao	2021-10-03 00:49:14	1999-09-09 00:00:00	C5
41265152774	Qiuqiuwang	2021-09-16 21:59:08	2021-10-08 00:00:00	C6
80766943391	Qiuqiuren	2021-10-11 01:01:02	1999-09-09 00:00:00	E4

2. Test data in Table 2:

district_bs_no	district_name	region
C1	Godlyland	center
C2	Godlyland	center
C3	Godlyland	center
C5	Centre Lukewarm Hillside	center
C6	Centre Lukewarm Hillside	center

3. Explanations:

✓ *Introduction to the test tables:*

Table 1 is the main test table and Table 2 is the district reference table (to show information about the districts where each base station is located).

✓ *Introduction to the test data:*

Target data are '13252155887', '233636', '72530750739', '47029460746', '1763059086' and '41265152774'. Invalid data are '80766943391'.

✓ *Reasons for choosing these data:*

Firstly, there are two records for Mark in Table 1: one showing Mark at base station 'C5' and the other showing Mark at base station 'C1'. 'C5' is within the boundaries of 'Centre Lukewarm Hillside' and 'C1' is within the boundaries of 'Godlyland'. The simulation here is that Mark arrives in 'Godlyland' from 'Centre Lukewarm Hillside' at 19:30 on 09-Oct-2021. So there are two districts affected by Mark, none of the other areas qualifies. Then, because the time limit is within the last 48 hours, people that are not within the time limit also need to be excluded. To conclude, there should be 6 target data and 1 invalid data.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT DISTINCT
  a.mobile_no
FROM
  ( SELECT * FROM history_tracking ORDER BY connected_time DESC ) a
WHERE
  a.district_bs_no = ANY (
  SELECT
    district_bs_no
  FROM
    district_bs_list
  WHERE
    district_name = ANY (
    SELECT
      district_name
    FROM
      district_bs_list
    WHERE
      district_bs_no = ANY ( SELECT district_bs_no FROM history_tracking WHERE mobile_no = 233636 AND (
disconnected_time = "1999-09-09 00:00:00" OR disconnected_time BETWEEN "2021-10-07 19:30:00" AND "2021-10-09
19:30:00" ) )
    )
  );
```

Message | Summary | Result 1 | Profile | Status

mobile_no
13252155887
233636
72530750739
47029460746
1763059086
41265152774

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 the source:

1. GPS location explanations:
   All GPS locations are made up of 16 fields. The framework for the GPS position expression is: $GPGGA, <UTC>, <latitude>, <northern latitude/south latitude>, <longitude>, <east longitude/west longitude>, <positioning quality indicators>, <number of satellites used>, <horizontal precision>, <the height of the antenna from sea level>, <the unit of height>, <the height of the earth ellipsoid relative to sea level>, <the unit of height>, <differential GPS data duration>, <differential reference base station number>, <hh>. In this report, only longitude and latitude are used. Considering the need to ensure the authenticity of the data, a real-life standardised format is used.

- SQL statement(s) for travel record insertion:

1. Assumption:
   The person's name is Paul and his mobile phone number is '14352311111', then assume that Paul has never been to any district in the Kingdom before (outside of the Kingdom). He is now going to a district in the north of the Kingdom called 'Cookland' at the base station called 'N1' (the time at this point is 2021-11-1 18:19:00), and one hour later, to a district in the north of the Kingdom called 'Kiwiland' at the base station called 'N4' (Assuming 'N1' and 'N4' are next to each other and there is no other base station between them).

2. SQL statements:
   - Stage 1: Paul enters 'Cookland'.
   - ✓ *Step 1:*
     Fill in the 'person_current_location' table with Paul's 'mobile number', 'name', 'GPS location', and 'district_bs_no'.
   - ✓ *Statement 1:*

```
INSERT INTO `cpt103cw3`.`person_current_location` (`mobile_no`,
`name`, `gps_location`, `district_bs_no`) VALUES (14352311111,
'Paul','$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,
,,0000*76', 'N1');
```

   - ✓ *Step 2:*
     Record Paul's trip history in the 'history_tracking' table, which includes: 'mobile number', 'name', 'GPS location', 'connected_time', 'disconnected_time' and 'district_bs_no'. In the 'connected_time' column, the current time is entered as '2021-11-1 18:19:00', and in the 'disconnected_time column', the system default time is temporarily entered as '1999-09-09 00:00:00' (The reason for saying 'temporarily' is that Paul has not yet left 'Cookland', but the system default time is entered because Paul will leave 'Cookland' in an hour and the 'disconnected_time' field cannot be null). And the district serial number 'N1' is filled in the 'district_bs_no field' column ('Cookland' has already been entered in the database, so this step is omitted here).
   - ✓ *Statement 2:*

```
INSERT INTO `cpt103cw3`.`history_tracking` (`mobile_no`, `name`,
`gps_location`, `connected_time`, `disconnected_time`,
`district_bs_no`) VALUES (14352311111, 'Paul',
'$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,,,0000*
76', '2021-11-01 18:19:00', '1999-09-09 00:00:00', 'N1');
```

   - Stage 2: Paul leaves 'Cookland' for 'Kiwiland' in an hour.
   - ✓ *Step 3:*
     Update 'N1' to 'N4' in 'district_bs_no' in the 'person_current_location' table.
   - ✓ *Statement 3:*

```
UPDATE `cpt103cw3`.`person_current_location` SET `name` = 'Paul',
`gps_location` =
'$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,,,0000*
76', `district_bs_no` = 'N4' WHERE `mobile_no` = 14352311111;
```

✓ *Step 4:*

Update the default time ('1999-09-09 00:00:00') in 'disconnected_time' to '2021-11-1 19:19:00' in the history_tracking table.

✓ *Statement 4:*

```
UPDATE `cpt103cw3`.`history_tracking` SET `mobile_no` =
'14352311111', `name` = 'Paul', `gps_location` =
'$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,,,0000*
76', `connected_time` = '2021-11-01 18:19:00', `disconnected_time` =
'2021-11-01 19:19:00', `district_bs_no` = 'N1' WHERE
`history_tracking_id` = 19;
```

✓ *Step 5:*

Add a new record to the 'history_tracking' table to indicate that Paul is currently at the 'N4' base station in 'Kiwiland'.

✓ *Statement 5:*

```
INSERT INTO `cpt103cw3`.`history_tracking` (`mobile_no`, `name`,
`gps_location`, `connected_time`, `disconnected_time`,
`district_bs_no`) VALUES (14352311111, 'Paul',
'$GPGGA,082006.000,3852.9234,N,11527.4160,E,1,08,1.0,20.6,M,,,,0000*
76', '2021-11-01 19:19:00', '1999-09-09 00:00:00', 'N4');
```

➢ <u>The result of the SELECT statements (screenshot):</u>

◆ Stage 1: Paul enters 'Cookland'.

✓ *Step 1:*

```
1 SELECT
2   *
3 FROM
4   person_current_location
5 WHERE
6   NAME = 'Paul';
```

Message Summary Result 1 Profile Status

mobile_no	name	gps_location	district_bs_no
14352311111	Paul	$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,,,0000*76	N1

✓ *Step 2:*

```
1 SELECT
2   *
3 FROM
4   history_tracking
5 WHERE
6   NAME = 'Paul';
```

Message Summary Result 1 Profile Status

history_tracking_id	mobile_no	name	gps_location	connected_time	disconnected_time	district_bs_no
18	14352311111	Paul	$GPGGA,082006.000,385;	2021-11-01 18:19:00	1999-09-09 00:00:00	N1

◆ Stage 2: Paul leaves 'Cookland' for 'Kiwiland' in an hour.

✓ *Step 3:*

```
1  SELECT
2    *
3  FROM
4    person_current_location
5  WHERE
6    NAME = 'Paul';
```

Message Summary Result 1 Profile Status

mobile_no	name	gps_location	district_bs_no
14352311111	Paul	$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,,,0000*76	N4

✓ *Step 4:*

```
1  SELECT
2    *
3  FROM
4    history_tracking
5  WHERE
6    NAME = 'Paul'
7    AND district_bs_no = 'N1';
```

Message Summary Result 1 Profile Status

history_tracking_id	mobile_no	name	gps_location	connected_time	disconnected_time	district_bs_no
18	14352311111	Paul	$GPGGA,082006.000,3852.9234,N,11527.4258,E,1,08,1.0,20.6,M,,,,0000*76	2021-11-01 18:19:00	1999-09-09 00:00:00	N1

✓ *Step 5:*

```
1  SELECT
2    *
3  FROM
4    history_tracking
5  WHERE
6    NAME = 'Paul'
7    AND district_bs_no = 'N4';
```

Message Summary Result 1 Profile Status

history_tracking_id	mobile_no	name	gps_location	connected_time	disconnected_time	district_bs_no
19	14352311111	Paul	$GPGGA,082006.000,385:	2021-11-01 19:19:00	1999-09-09 00:00:00	N4

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.

➢ <u>SQL statement:</u>

```
select
    a.avgtime, b.work_hospital_id
from
    (select
        doctor_id, avg(timestampdiff(MINUTE, sample_test_time,
sample_report_time)) as avgtime, substr(doctor_id, 7, 2) as hospital_no
    from
        patient_test_information
    group by
        substr(doctor_id, 7, 2)
    order by
        avgtime asc
    limit 1) a
left join
      doctor_list b
on
    a.doctor_id = b.doctor_id;
```

```sql
SELECT
  a.avgtime,
  b.work_hospital_id
FROM
  (
  SELECT
    doctor_id,
    avg(
    timestampdiff( MINUTE, sample_test_time, sample_report_time )) AS avgtime,
    substr( doctor_id, 7, 2 ) AS hospital_no
  FROM
    patient_test_information
  GROUP BY
    substr( doctor_id, 7, 2 )
  ORDER BY
    avgtime ASC
    LIMIT 1
  ) a
  LEFT JOIN doctor_list b ON a.doctor_id = b.doctor_id;
```

select the target hospital and the least average report generation time

calculate average report generation time and sort by hospital number

join two tables

➢ Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	doctor_id	sample_test_time	sample_report_time
13626192177	1100600101	2021-10-26 17:00:00	2021-10-26 20:00:00
70915870657	1100600101	2021-11-01 17:12:18	2021-11-01 20:02:34
12555812470	1100600101	2021-11-01 17:34:03	2021-11-01 20:23:43
93871558168	1100700205	2021-11-01 16:04:45	2021-11-01 17:34:04
20695282222	1100700205	2021-11-01 16:23:56	2021-11-01 17:44:45
72530750739	1100100608	2021-10-04 18:38:59	2021-10-04 19:00:15
15305045857	1100100608	2021-10-03 18:44:45	2021-10-03 19:00:53

2. Test data in Table 2:

doctor_id	work_hospital_id
1100600101	1
1100700205	2
1100100608	6

3. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the main test table and Table 2 is the district reference table (to show where each doctor works).

✓ *Introduction to the test data:*
Target data is 'work_hospital_id' named '6'. Invalid data are 'work_hospital_id' named '1' and '2'.

✓ *Reasons for choosing these data:*
Firstly, in terms of column design, 'sample_test_time' and 'sample_report_time' are the two important columns. Secondly, it was predetermined that hospital '6' was the most efficient hospital with a report generation time of less than an hour, while the other two hospitals took more than an

hour. To make it easier to check that the SQL statements are correct, only 7 data items are arranged here.

➢ The result of the SELECT statement (screenshot):

```
SELECT
  a.avgtime,
  b.work_hospital_id
FROM
  (
  SELECT
    doctor_id,
    avg(
    timestampdiff( MINUTE, sample_test_time, sample_report_time )) AS avgtime,
    substr( doctor_id, 7, 2 ) AS hospital_no
  FROM
    patient_test_information
  GROUP BY
    substr( doctor_id, 7, 2 )
  ORDER BY
    avgtime ASC
    LIMIT 1
  ) a
  LEFT JOIN doctor_list b ON a.doctor_id = b.doctor_id;
```

Message | Summary | Result 1 | Profile | Status

avgtime	work_hospital_id
18.5000	6

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).

➢ SQL statement:

```
select
    mobile_no, count(*) as number
from
    patient_test_information
where
    sample_collect_time between “2021-10-03 00:00:00” and “2021-10-05
00:00:00”
group by
    mobile_no
having
    number = 2;
```

```
SELECT
  mobile_no,
  count(*) AS number
FROM
  patient_test_information
WHERE
  sample_collect_time BETWEEN "2021-10-03 00:00:00"
  AND "2021-10-05 00:00:00"
GROUP BY
  mobile_no
HAVING
  number = 2;
```

select the information to be displayed: the phone number of the person who did the viral tests twice within the specified time period.

control time within '2021-10-03 00:00' to '2021-10-03 00:00'

limit the number of virus tests to 2

➢ Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_collect_time
93871558168	2021-10-03 00:00:00
93871558168	2021-10-04 00:00:00
93871558168	2021-10-05 00:00:00
80766943391	2021-10-03 01:00:00
80766943391	2021-10-04 01:23:00
47029460746	2021-10-04 17:29:52
72530750739	2021-10-04 17:38:29
16406019403	2021-10-04 17:42:10
15305045857	2021-10-03 17:44:36
81918316107	2021-10-05 18:17:47

2. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the main test table (to show the different conditions for each person).

✓ *Introduction to the test data:*
Target data is the phone number called '80766943391'. Invalid data are phone numbers called '93871558168', '47029460746', '72530750739', '16406019403', '15305045857' and '81918316107'.

✓ *Reasons for choosing these data:*
Firstly, for the 7 different phone numbers in Table 1, there are three groups of cases. The first is the case where the phone number is '93871558168': this person did 3 virus tests within the limited period. The second is the case where the phone number is '80766943391': the person did 2 virus tests in the limited period. The third is the case where the phone numbers are '47029460746', '72530750739', '16406019403' and '15305045857': each person did 1 virus test within the limited period. The last case is for the phone number '81918316107': this person did 1 virus test outside the limited period. These four cases are listed to check that whether the SQL statement is correct (only the second case '80766943391' is compliant).

➢ <u>The result of the SELECT statement (screenshot):</u>

```
1  SELECT
2    mobile_no,
3    count(*) AS number
4  FROM
5    patient_test_information
6  WHERE
7    sample_collect_time BETWEEN "2021-10-03 00:00:00"
8    AND "2021-10-05 00:00:00"
9  GROUP BY
10   mobile_no
11 HAVING
12   number = 2;
```

信息 结果 1 剖析 状态

mobile_no	number
80766943391	2

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

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

➢ SQL statement:

```
select
    a.district_name, b.risk_level_name
from
    (select * from judge_district_risk limit 5) a
left join
    risk_level_list b
on
    a.risk_level = b.risk_level
order by
   a.risk_level desc;
```

```
1 SELECT
2   a.district_name,
3   b.risk_level_name
4 FROM
5   ( SELECT * FROM judge_district_risk LIMIT 5 ) a
6   LEFT JOIN risk_level_list b ON a.risk_level = b.risk_level
7 ORDER BY
8   a.risk_level DESC;
```

select area name and risk level

select the first five entries in the 'judge_district_risk' table

rank the districts in order of highest to lowest risk

➢ Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

district_name	risk_level
Centre Lukewarm Hillside	3
Glow Sand district	2
Lenny town	3
Raspberry town	1
Bunny Tail district	1

2. Test data in Table 2:

risk_level	risk_level_name
1	low
2	mid
3	high

3. Explanations:

✓ *Introduction to the test tables:*

Table 1 is the main test table and Table 2 is the district reference table (to show information about 'risk_level' in digital form).

✓ *Introduction to the test data:*
The target data is all the records in Table 1. The task of this use case is to sort the data in order of risk from highest to lowest.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
1 SELECT
2   a.district_name,
3   b.risk_level_name
4 FROM
5   ( SELECT * FROM judge_district_risk LIMIT 5 ) a
6   LEFT JOIN risk_level_list b ON a.risk_level = b.risk_level
7 ORDER BY
8   a.risk_level DESC;
```

Message Summary Result 1 Profile Status

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

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

➢ <u>SQL statement:</u>

```
select
    a.mobile_no, b.name
from
    (
    select
        t1.*, t2.district_bs_no
    from
        patient_test_information t1
    left join
        hospital_list t2
    on
        substr(t1.doctor_id, 7, 2) = t2.hospital_id
    ) a
left join
    patient_basic_information b
on
    a.mobile_no = b.mobile_no
where
    a.sample_result = "positive" and a.sample_report_time like "2021-10-04
__:__:__"
and
    a.district_bs_no = any(
    select
        district_bs_no
    from
        district_bs_list
    where
        district_name = "Centre Lukewarm Hillside"
    );
```

```sql
SELECT
  a.mobile_no,
  b.NAME
FROM
  (
  SELECT
    t1.*,
    t2.district_bs_no
  FROM
    patient_test_information t1
    LEFT JOIN hospital_list t2 ON substr( t1.doctor_id, 7, 2 ) = t2.hospital_id
  ) a
  LEFT JOIN patient_basic_information b ON a.mobile_no = b.mobile_no
WHERE
  a.sample_result = "positive"
  AND a.sample_report_time LIKE "2021-10-04 __:__:__"
  AND a.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE
district_name = "Centre Lukewarm Hillside" );
```

select the phone number and name of the eligible person

make table joins and put limits on time and the test result

➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_result	sample_report_time	district_bs_no
80766943391	positive	2021-10-04 23:59:11	E4
47029460746	positive	2021-10-04 19:30:16	C5
72530750739	positive	2021-10-04 19:00:15	C6
16406019403	negative	2021-10-04 19:42:28	C5
15305045857	negative	2021-10-03 19:00:53	C6

2. Test data in Table 2:

district_bs_no	district_name
E4	Perryland
C5	Centre Lukewarm Hillside
C6	Centre Lukewarm Hillside

3. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the main test table (to show the different cases of testing virus) and Table 2 is the district reference table (to show information about the districts where each base station is located).

✓ *Introduction to the test data:*
Target data are the phone number called ‘47029460746’ and ‘72530750739’. Invalid data are phone numbers called ‘80766943391’, ‘16406019403’ and ‘15305045857’.

✓ *Reasons for choosing these data:*
Firstly, these 5 records in Table 1 can be divided into four groups. The first group is the person whose telephone number is ‘80766943391’. He tested positive within the specified time frame, but his district is out of range. The second group is those with telephone numbers ‘47029460746’ and ‘72530750739’. They are the correct data. The third group is the person whose telephone number is

'16406019403'. He matches the time and district but the test result is negative. The last group is the person whose telephone number is '15305045857'. His time and test result did not qualify.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT
  a.mobile_no,
  b.NAME
FROM
  (
  SELECT
    t1.*,
    t2.district_bs_no
  FROM
    patient_test_information t1
    LEFT JOIN hospital_list t2 ON substr( t1.doctor_id, 7, 2 ) = t2.hospital_id
  ) a
  LEFT JOIN patient_basic_information b ON a.mobile_no = b.mobile_no
WHERE
  a.sample_result = "positive"
  AND a.sample_report_time LIKE "2021-10-04 __:__:__"
  AND a.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE
district_name = "Centre Lukewarm Hillside" );
```

信息 结果 1 剖析 状态

mobile_no	NAME
47029460746	Xifenglan
72530750739	Leishilain

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.

➢ <u>SQL statement:</u>

```
select
    (a.count - b.count) as increasement
from
    ((select count(mobile_no) as count
    from
        (
        select
            t1.*, t2.district_bs_no
        from
            patient_test_information t1
        left join
            hospital_list t2
        on
            substr(t1.doctor_id, 7, 2) = t2.hospital_id
        ) m
    where
        m.sample_result = "positive" and m.sample_report_time like "2021-
10-05 __:__:__"
    and
        m.district_bs_no = any(
        select
            district_bs_no
        from
            district_bs_list
        where
            district_name = "Centre Lukewarm Hillside")) as a,
    (select count(mobile_no) as count
```

```
    from
        (
        select
            t1.*, t2.district_bs_no
        from
            patient_test_information t1
        left join
            hospital_list t2
        on
            substr(t1.doctor_id, 7, 2) = t2.hospital_id
        ) n
    where
        n.sample_result = "positive" and n.sample_report_time like "2021-
10-04 __:__:__"
    and
        n.district_bs_no = any(
        select
            district_bs_no
        from
            district_bs_list
        where
            district_name = "Centre Lukewarm Hillside")) as b);
```

```
SELECT
  ( a.count - b.count ) AS increasement
FROM
  ((
    SELECT
      count( mobile_no ) AS count
    FROM
      (
      SELECT
        t1.*,
        t2.district_bs_no
      FROM
        patient_test_information t1
        LEFT JOIN hospital_list t2 ON substr( t1.doctor_id, 7, 2 ) = t2.hospital_id
      ) m
    WHERE
      m.sample_result = "positive"
      AND m.sample_report_time LIKE "2021-10-05 __:__:__"
      AND m.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE district_name = "Centre Lukewarm Hillside" )) AS a,
  (
  SELECT
    count( mobile_no ) AS count
  FROM
    (
    SELECT
      t1.*,
      t2.district_bs_no
    FROM
      patient_test_information t1
      LEFT JOIN hospital_list t2 ON substr( t1.doctor_id, 7, 2 ) = t2.hospital_id
    ) n
  WHERE
    n.sample_result = "positive"
    AND n.sample_report_time LIKE "2021-10-04 __:__:__"
  AND n.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE district_name = "Centre Lukewarm Hillside" )) AS b
);
```

calculate new increments

new positive cases in 'Centre Lukewarm Hillside' district on 05/10/2021

new positive cases in 'Centre Lukewarm Hillside' district on 04/10/2021

➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_result	sample_report_time	district_bs_no
47029460746	positive	2021-10-04 19:30:16	C5
72530750739	positive	2021-10-04 19:00:15	C6
16406019403	negative	2021-10-04 19:42:28	C5
81918316107	negative	2021-10-05 18:20:09	C5

2. Test data in Table 2:

district_bs_no	district_name

C5	Centre Lukewarm Hillside
C6	Centre Lukewarm Hillside

3. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the main test table (to show the different cases of testing virus) and Table 2 is the district reference table (to show information about the districts where each base station is located).

✓ *Introduction to the test data:*
All data in Table 1 are valid.

✓ *Reasons for choosing these data:*
First, there are 3 records about 4 October 2021 in Table 1. In these records, only the records with mobile phone numbers named '72530750739' and '47029460746' are needed. The person with the mobile phone number named '16406019403' has a negative test result, so it does not qualify. Finally, there are 2 positive cases on 4 October 2021. However, there is only one record about 5 October 2021. And its test result is negative, which does not match the requirement. So, on 5 October 2021, there are 0 positive cases. Based on the definition of 'increasement' in this use case, the result for 'increasement' is -2.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT
  ( a.count - b.count ) AS increasement
FROM
  ((
    SELECT
      count( mobile_no ) AS count
    FROM
      (
      SELECT
        t1.*,
        t2.district_bs_no
      FROM
        patient_test_information t1
        LEFT JOIN hospital_list t2 ON substr( t1.doctor_id, 7, 2 ) = t2.hospital_id
      ) m
    WHERE
      m.sample_result = "positive"
      AND m.sample_report_time LIKE "2021-10-05 __:__:__"
      AND m.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE district_name = "Centre Lukewarm Hillside" )) AS a,
    (
    SELECT
      count( mobile_no ) AS count
    FROM
      (
      SELECT
        t1.*,
        t2.district_bs_no
      FROM
        patient_test_information t1
        LEFT JOIN hospital_list t2 ON substr( t1.doctor_id, 7, 2 ) = t2.hospital_id
      ) n
    WHERE
      n.sample_result = "positive"
      AND n.sample_report_time LIKE "2021-10-04 __:__:__"
    AND n.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE district_name = "Centre Lukewarm Hillside" )) AS b
  );
```

信息 结果 1 剖析 状态

increasement
-2

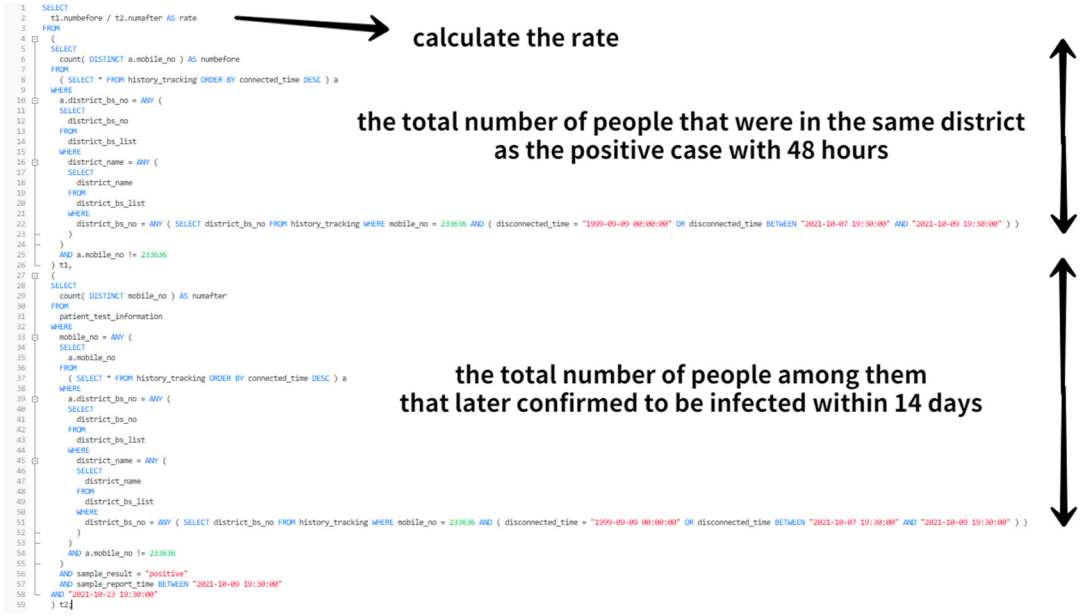
Use case 8: Assume that the spread rate of a virus is calculated by dividing the total number of people that were in the same district as the positive case with 48 hours (calculated in ***use case 1***) by the total number of people among them that later confirmed to be infected in 14 days. Again, assume that a person called Mark was tested to be positive at 19:30 on 09-Oct-2021 and he is the only person in the country that has coughed-19. Please write a query that calculates the spread rate of the virus.

➢ <u>SQL statement:</u>

```
select
    t1.numbefore / t2.numafter as rate
```

```
from
    (
    select
        count(distinct a.mobile_no) as numbefore
    from
        (select * from history_tracking order by connected_time desc) a
    where
        a.district_bs_no = any(
        select
            district_bs_no from district_bs_list
        where
            district_name = any(
            select
                district_name
            from
                district_bs_list
            where
                district_bs_no = any(
                select
                    district_bs_no
                from
                    history_tracking
                where
                    mobile_no = 233636 and (disconnected_time = "1999-09-09
00:00:00" or disconnected_time between "2021-10-07 19:30:00" and "2021-10-
09 19:30:00")
                )
            )
        )
    and
        a.mobile_no != 233636
    ) t1,
    (
    select
        count(distinct mobile_no) as numafter
    from
        patient_test_information
    where
        mobile_no = any(
        select
            a.mobile_no
        from
            (select * from history_tracking order by connected_time desc) a
        where
            a.district_bs_no = any(
            select
                district_bs_no from district_bs_list
            where
                district_name = any(
                select
                    district_name
                from
                    district_bs_list
                where
                    district_bs_no = any(
                    select
                        district_bs_no
                    from
                        history_tracking
                    where
```

```
                        mobile_no = 233636 and (disconnected_time = "1999-
09-09 00:00:00" or disconnected_time between "2021-10-07 19:30:00" and
"2021-10-09 19:30:00")
                    )
                )
            )
        and
            a.mobile_no != 233636
        )
    and
        sample_result = "positive"
    and
        sample_report_time between "2021-10-09 19:30:00" and "2021-10-23
19:30:00"
    ) t2;
```



➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	name	connected_time	disconnected_time
13252155887	Wuxiangzhibing	2021-10-07 21:34:26	1999-09-09 00:00:00
233636	Mark	2021-10-09 19:00:00	1999-09-09 00:00:00
72530750739	Leishilaimu	2021-10-04 15:12:46	1999-09-09 00:00:00
47029460746	Xifenglang	2021-10-03 02:02:02	1999-09-09 00:00:00
1763059086	Leiniao	2021-10-03 00:49:14	1999-09-09 00:00:00
41265152774	Qiuqiuwang	2021-09-16 21:59:08	2021-10-08 00:00:00

2. Test data in Table 2:

mobile_no	name	sample_result	sample_report_time
13252155887	Wuxiangzhibing	positive	2021-10-10 18:29:39

72530750739	Leishilaimu	positive	2021-10-04 19:00:15
47029460746	Xifenglang	positive	2021-10-04 19:30:16
1763059086	Leiniao	negative	2021-10-11 16:21:42
41265152774	Qiuqiuwang	positive	2021-10-24 16:23:14

3. Explanations:

✓ *Introduction to the test tables:*

Table 1 is the history tracking table (the 5 records which do not include 'Mark' are the affected people in one district) and Table 2 is the viral test table which records the people infected later within 14 days.

✓ *Introduction to the test data:*

All data in Table 1 are valid.

✓ *Reasons for choosing these data:*

Table 2 is the main table that tests whether this SQL statement is right. Firstly, the data in Table 2 can be divided into 3 groups. The first group is the data with the name called 'Wuxiangzhibing'. It represents the person who was tested positive within 24 days. And he is the only one that was infected within the limited period. The second group is the data with the names called 'Leishilaimu', 'Xifenglang' and 'Qiuqiuwang'. They represent persons who were tested positive but were not within the limited period. The third group is the data with the name called 'Leiniao'. It represents the person who did the virus test within the limited period but was tested negative. Therefore, there is only one infected person ('Wuxiangzhibing') who meets the requirement.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT
  t1.numbefore / t2.numafter AS rate
FROM
  (
  SELECT
    count( DISTINCT a.mobile_no ) AS numbefore
  FROM
    ( SELECT * FROM history_tracking ORDER BY connected_time DESC ) a
  WHERE
    a.district_bs_no = ANY (
    SELECT
      district_bs_no
    FROM
      district_bs_list
    WHERE
      district_name = ANY (
      SELECT
        district_name
      FROM
        district_bs_list
      WHERE
        district_bs_no = ANY ( SELECT district_bs_no FROM history_tracking WHERE mobile_no = 233636 AND ( disconnected_time = "1999-09-09 00:00:00" OR disconnected_time BETWEEN "2021-10-07 19:30:00" AND "2021-10-09 19:30:00" ) )
      )
    )
    AND a.mobile_no != 233636
  ) t1,
  (
  SELECT
    count( DISTINCT mobile_no ) AS numafter
  FROM
    patient_test_information
  WHERE
    mobile_no = ANY (
    SELECT
      a.mobile_no
    FROM
      ( SELECT * FROM history_tracking ORDER BY connected_time DESC ) a
    WHERE
      a.district_bs_no = ANY (
      SELECT
        district_bs_no
      FROM
        district_bs_list
      WHERE
        district_name = ANY (
        SELECT
          district_name
        FROM
          district_bs_list
        WHERE
          district_bs_no = ANY ( SELECT district_bs_no FROM history_tracking WHERE mobile_no = 233636 AND ( disconnected_time = "1999-09-09 00:00:00" OR disconnected_time BETWEEN "2021-10-07 19:30:00" AND "2021-10-09 19:30:00" ) )
        )
      )
      AND a.mobile_no != 233636
    )
```

信息 结果 1 剖析 状态

rate
5.0000

Extended Use Cases

Use case 1: Suppose that two hospitals have their own GPS locations. The two hospitals are called "hospital1" and "hospital13" respectively. First, please determine the longitude and return true if they are equal, false if they are not. Then, please determine the latitude and return true if they are equal, or false if they are not. Use SQL statements to implement this.

- SQL statement:

```
select
   (case
when
   substr(a.hospital_gps_location, 31, 10) =
   substr(b.hospital_gps_location, 31, 10)
then
   'true'
else
   'false'
end) as whether_longitude_equal,
   (case
when
   substr(a.hospital_gps_location, 19, 9) =
   substr(b.hospital_gps_location, 19, 9)
then
   'true'
else
   'false'
end) as whether_latitude_equal
from
   (select * from hospital_list where hospital_name = 'hospital1') a,
   (select * from hospital_list where hospital_name = 'hospital13') b;
```

```
SELECT
  (
  CASE

      WHEN substr( a.hospital_gps_location, 31, 10 ) = substr( b.hospital_gps_location, 31, 10
) THEN
      'true' ELSE 'false'
    END
    ) AS whether_longitude_equal,
    (
    CASE

        WHEN substr( a.hospital_gps_location, 19, 9 ) = substr( b.hospital_gps_location, 19, 9
) THEN
        'true' ELSE 'false'
      END
      ) AS whether_latitude_equal
    FROM
    ( SELECT * FROM hospital_list WHERE hospital_name = 'hospital1' ) a,
  ( SELECT * FROM hospital_list WHERE hospital_name = 'hospital13' ) b;
```

determine whether two longitudes are equal

determine whether two latitudes are equal

- Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

hospital_name	hospital_gps_location
hospital1	$GPGGA,082006.000,3852.0100,N,11527.0100,E,1,08,1.0,20.6,M,,,,0000*35
hospital13	$GPGGA,082006.000,3852.0100,N,11527.1300,E,1,08,1.0,20.6,M,,,,0000*35

2. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the hospital GPS location table (the highlighted words are mainly used).

✓ *Introduction to the test data:*
All data in Table 1 are valid.

✓ *Reasons for choosing these data:*
In the process of querying with SQL statements, a completely new conditional judgement statement was used that was not taught by teachers in class. The conditional judgement statement was used to get results in different situations.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT
  (
  CASE

      WHEN substr( a.hospital_gps_location, 31, 10 ) = substr( b.hospital_gps_location, 31, 10
) THEN
      'true' ELSE 'false'
    END
    ) AS whether_longitude_equal,
    (
    CASE

      WHEN substr( a.hospital_gps_location, 19, 9 ) = substr( b.hospital_gps_location, 19, 9
) THEN
      'true' ELSE 'false'
    END
    ) AS whether_latitude_equal
  FROM
  ( SELECT * FROM hospital_list WHERE hospital_name = 'hospital1' ) a,
 ( SELECT * FROM hospital_list WHERE hospital_name = 'hospital13' ) b;
```

信息 结果 1 剖析 状态

whether_longitude_equal	whether_latitude_equal
false	true

Use case 2: The definition of the risk level: a district with one or more positive cases is considered high risk if they stay for more than 24 hours. A district with positive cases for 1 week is considered the middle risk (a positive case passes through but does not stay for the full 24 hours within 7 days). Districts with no positive cases for 1 week are considered low risk. Please select all the high and middle-risk districts based on the definition of the risk level.

➢ <u>SQL statement:</u>

1. For the high risk:

```
select
    distinct t1.district_name
from
    district_bs_list t1
where
    district_bs_no = any(
    select
        distinct b.district_bs_no
    from
        (select * from patient_test_information where sample_result =
"positive") a
    left join
```

```
        history_tracking b
    on
        a.mobile_no = b.mobile_no
    where
        (b.disconnected_time = "1999-09-09 00:00:00")
    or
        timestampdiff(hour, b.connected_time, b.disconnected_time) >= 24
    );
```

```
SELECT DISTINCT
  t1.district_name
FROM
  district_bs_list t1
WHERE
  district_bs_no = ANY (
  SELECT DISTINCT
    b.district_bs_no
  FROM
    ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) a
    LEFT JOIN history_tracking b ON a.mobile_no = b.mobile_no
  WHERE
    ( b.disconnected_time = "1999-09-09 00:00:00" )
  OR timestampdiff( HOUR, b.connected_time, b.disconnected_time ) >= 24
  );
```

select the target district name

The limit is based on the 'positive' result and a stay of more than 24 hours at one district

2. For the middle risk:

```
select
    t2.district_name
from
    district_bs_list t2
where
    district_bs_no = any(
    select
        distinct c.district_bs_no
    from
        (select * from patient_test_information where sample_result =
"positive") d
    left join
        history_tracking c
    on
        d.mobile_no = c.mobile_no
    where
        timestampdiff(hour, c.connected_time, c.disconnected_time) < 24
    and
        c.disconnected_time != "1999-09-09 00:00:00"
    );
```

```
SELECT
  t2.district_name
FROM
  district_bs_list t2
WHERE
  district_bs_no = ANY (
  SELECT DISTINCT
    c.district_bs_no
  FROM
    ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) d
    LEFT JOIN history_tracking c ON d.mobile_no = c.mobile_no
  WHERE
    timestampdiff( HOUR, c.connected_time, c.disconnected_time ) < 24
  AND c.disconnected_time != "1999-09-09 00:00:00"
  );
```

select the target district name

The limit is based on the 'positive' result and a stay of less than 24 hours at one district

- Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_result
80766943391	positive
72530750739	positive
47029460746	positive
233636	positive
13252155887	positive
41265152774	positive

2. Test data in Table 2:

mobile_no	connected_time	disconnected_time	district_bs_no
80766943391	2021-10-11 01:01:02	1999-09-09 00:00:00	E4
80766943391	2021-10-10 15:02:38	2021-10-11 01:01:02	N4
72530750739	2021-10-04 15:12:46	1999-09-09 00:00:00	C6
72530750739	2021-09-01 15:12:46	2021-10-04 15:12:46	W5
47029460746	2021-10-03 02:02:02	1999-09-09 00:00:00	C5
47029460746	2021-09-30 02:02:02	2021-10-03 02:02:02	W1
233636	2021-10-09 19:00:00	1999-09-09 00:00:00	C1
233636	2021-09-01 02:23:11	2021-10-01 12:34:23	C4
233636	2021-10-01 12:34:23	2021-10-09 19:00:00	C5
13252155887	2021-10-07 21:34:26	1999-09-09 00:00:00	C1
41265152774	2021-09-16 21:59:08	2021-10-08 00:00:00	C6
41265152774	2021-10-08 00:00:00	2021-10-24 15:19:56	C8
41265152774	2021-10-24 15:19:56	1999-09-09 00:00:00	C7

3. Test data in Table 3:

district_bs_no	district_name
C1	Godlyland
C4	Farrellland
C5	Centre Lukewarm Hillside
C6	Centre Lukewarm Hillside
C7	Farrellland
C8	Farrellland
W1	Leslieland
W5	Earlland
E4	Perryland
N4	Kiwiland

4. Explanations:

- ✓ *Introduction to the test tables:*
  Table 1 is the viral tests table (to show mobile phone numbers of positive patients). Table 2 is the history tracking table about each patient (If '1999-09-09 00:00:00' appears in the column 'disconnected_time', it means that the patient has not left this base station yet. '1999-09-09 00:00:00' is the default value). And Table 3 is the base station table (to record names of districts where each base station is located).

✓ *Introduction to the test data:*
All data in Table 1, 2 and 3 are valid.

✓ *Reasons for choosing these data:*
First, all data presented in Table 1 are representative of all positive patients. All patients in Table 2 except for the positive patient with the phone number '13252155887' had visited more than one district. Among all the patients who had been to more than one district, only the patient with the phone number '80766943391' had been in the base station 'N4' (district called 'Kiwiland') for less than 24 hours. So, except for 'Kiwiland' which is a medium risk district, all the other districts ('Godlyland', 'Farrellland Centre Lukewarm Hillside', 'Earlland' and 'Perryland') are high-risk districts.

➢ <u>The result of the SELECT statement (screenshot):</u>

1. For the high risk:

```
SELECT DISTINCT
  t1.district_name
FROM
  district_bs_list t1
WHERE
  district_bs_no = ANY (
  SELECT DISTINCT
    b.district_bs_no
  FROM
    ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) a
    LEFT JOIN history_tracking b ON a.mobile_no = b.mobile_no
  WHERE
    ( b.disconnected_time = "1999-09-09 00:00:00" )
  OR timestampdiff( HOUR, b.connected_time, b.disconnected_time ) >= 24
  );
```

信息 结果 1 剖析 状态

district_name
Centre Lukewarm Hi
Earlland
Farrellland
Godlyland
Leslieland
Perryland

2. For the middle risk:

```sql
SELECT
  t2.district_name
FROM
  district_bs_list t2
WHERE
  district_bs_no = ANY (
  SELECT DISTINCT
    c.district_bs_no
  FROM
    ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) d
    LEFT JOIN history_tracking c ON d.mobile_no = c.mobile_no
  WHERE
    timestampdiff( HOUR, c.connected_time, c.disconnected_time ) < 24
  AND c.disconnected_time != "1999-09-09 00:00:00"
  );
```

信息 结果 1 剖析 状态

district_name
Kiwiland

Use case 3: According to use case 1, please write SQL statements to update the risk level of each district.

➢ SQL statement:

1. Update for the high risk:

```sql
update
	judge_district_risk
set
	risk_level = 3
where
	district_name = any(
	select
		distinct t1.district_name
	from
		district_bs_list t1
	where
		district_bs_no = any(
		select
			distinct b.district_bs_no
		from
			(select * from patient_test_information where
sample_result = "positive") a
		left join
			history_tracking b
		on
			a.mobile_no = b.mobile_no
		where
			(b.disconnected_time = "1999-09-09 00:00:00")
		or
			timestampdiff(hour, b.connected_time,
b.disconnected_time) >= 24
		)
);
```

```sql
UPDATE judge_district_risk
SET risk_level = 3
WHERE
  district_name = ANY (
  SELECT DISTINCT
    t1.district_name
  FROM
    district_bs_list t1
  WHERE
    district_bs_no = ANY (
    SELECT DISTINCT
      b.district_bs_no
    FROM
      ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) a
      LEFT JOIN history_tracking b ON a.mobile_no = b.mobile_no
    WHERE
      ( b.disconnected_time = "1999-09-09 00:00:00" )
      OR timestampdiff( HOUR, b.connected_time, b.disconnected_time ) >= 24
    )
  );
```

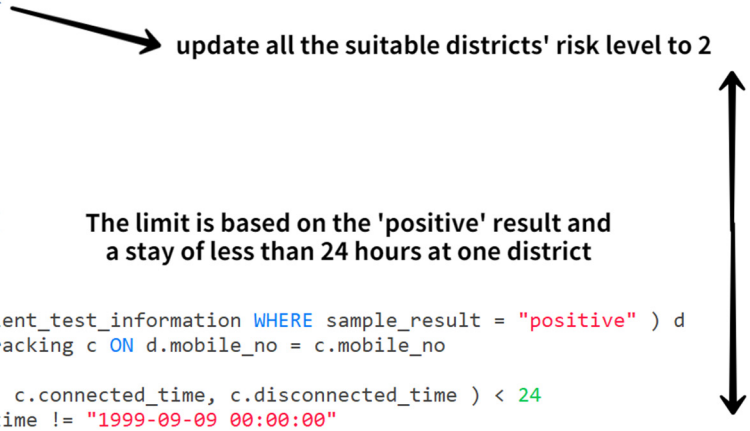
update all the suitable districts' risk level to 3

The limit is based on the 'positive' result and a stay of more than 24 hours at one district

2. Update for the middle risk:

```sql
update
	judge_district_risk
set
	risk_level = 2
where
	district_name = any(
	select
		t2.district_name
	from
		district_bs_list t2
	where
		district_bs_no = any(
		select
			distinct c.district_bs_no
		from
(select * from patient_test_information where sample_result = "positive") d
		left join
			history_tracking c
		on
			d.mobile_no = c.mobile_no
		where
timestampdiff(hour, c.connected_time, c.disconnected_time) < 24
		and
			c.disconnected_time != "1999-09-09 00:00:00"
		)
	);
```

```sql
UPDATE judge_district_risk
SET risk_level = 2
WHERE
  district_name = ANY (
  SELECT
    t2.district_name
  FROM
    district_bs_list t2
  WHERE
    district_bs_no = ANY (
    SELECT DISTINCT
      c.district_bs_no
    FROM
      ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) d
      LEFT JOIN history_tracking c ON d.mobile_no = c.mobile_no
    WHERE
      timestampdiff( HOUR, c.connected_time, c.disconnected_time ) < 24
      AND c.disconnected_time != "1999-09-09 00:00:00"
    )
  );
```



➢ Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

district_name	risk_level
Godlyland	1
Farrellland	1
Centre Lukewarm Hillside	1
Leslieland	1
Earlland	1
Perryland	1
Kiwiland	1

2. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the original risk level table.

✓ *Introduction to the test data:*
All data in Table 1 are valid.

✓ *Reasons for choosing these data:*
The risk level for all districts in Table 1 is set to 1 by default, as this can make it obvious that all changes in the 'risk_level' column. Thus, it justifies the SQL statement.

➢ The result of the SELECT statement (screenshot):

1. Update for the high risk:

```sql
SELECT
  district_name,
  risk_level
FROM
  judge_district_risk
WHERE
  risk_level = 3;
```

信息 结果 1 剖析 状态

district_name	risk_level
Centre Lukewarm Hi	3
Godlyland	3
Farrellland	3
Perryland	3
Leslieland	3
Earlland	3

2. Update for the middle risk:

```sql
SELECT
  district_name,
  risk_level
FROM
  judge_district_risk
WHERE
  risk_level = 2;
```

信息 结果 1 剖析 状态

district_name	risk_level
Kiwiland	2

Use case 4: Having known the locations of 12 hospitals, and now we assume that the location of a person is '$GPGGA,082006.000,3852.0110,N,11527.0110,E,1,08,1.0,20.6,M,,,,0000*35'. Please list the name of the nearest hospital and the difference of the GPS location between the person and the hospital (only accuracy and latitude are considered in the calculation).

➢ SQL statement:

```sql
select
hospital_name, (substr(hospital_gps_location, 19, 9) - 3852.0110) as
northdiff,
      (substr(hospital_gps_location, 31, 10) - 11527.0110) as eastdiff
from
      hospital_list
order by
      northdiff asc, eastdiff asc
limit 1;
```

```sql
SELECT
  hospital_name,
  ( substr( hospital_gps_location, 19, 9 ) - 3852.0110 ) AS northdiff,
  ( substr( hospital_gps_location, 31, 10 ) - 11527.0110 ) AS eastdiff
FROM
  hospital_list
ORDER BY
  northdiff ASC,
  eastdiff ASC
  LIMIT 1;
```

use each hospital's GPS location to minus the person's GPS location

sort in ascending order and find the hospital with the smallest difference

➢ Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

hospital_name	hospital_gps_location
hospital1	$GPGGA,082006.000,3852.0100,N,11527.0100,E,1,08,1.0,20.6,M,,,,0000*35
hospital2	$GPGGA,082006.000,3852.0200,N,11527.0200,E,1,08,1.0,20.6,M,,,,0000*35
hospital3	$GPGGA,082006.000,3852.0300,N,11527.0300,E,1,08,1.0,20.6,M,,,,0000*35
hospital4	$GPGGA,082006.000,3852.0400,N,11527.0400,E,1,08,1.0,20.6,M,,,,0000*35
hospital5	$GPGGA,082006.000,3852.0500,N,11527.0500,E,1,08,1.0,20.6,M,,,,0000*35
hospital6	$GPGGA,082006.000,3852.0600,N,11527.0600,E,1,08,1.0,20.6,M,,,,0000*35
hospital7	$GPGGA,082006.000,3852.0700,N,11527.0700,E,1,08,1.0,20.6,M,,,,0000*35
hospital8	$GPGGA,082006.000,3852.0800,N,11527.0800,E,1,08,1.0,20.6,M,,,,0000*35
hospital9	$GPGGA,082006.000,3852.0900,N,11527.0900,E,1,08,1.0,20.6,M,,,,0000*35
hospital10	$GPGGA,082006.000,3852.1000,N,11527.1000,E,1,08,1.0,20.6,M,,,,0000*35
hospital11	$GPGGA,082006.000,3852.1100,N,11527.1100,E,1,08,1.0,20.6,M,,,,0000*35
hospital12	$GPGGA,082006.000,3852.1200,N,11527.1200,E,1,08,1.0,20.6,M,,,,0000*35

2. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the hospital's GPS location table (the highlighted words are mainly used).

✓ *Introduction to the test data:*
Target data in Table 1 is 'hospital1'. And invalid data are 'hospital2' to 'hospital12'.

✓ *Reasons for choosing these data:*
To better prove the correctness of the SQL statement, all 12 hospitals that can currently do virus tests and their corresponding GPS locations are listed in Table 1. And it is assumed that 'hospital1' is the nearest hospital to this person.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
1  SELECT
2    hospital_name,
3    ( substr( hospital_gps_location, 19, 9 ) - 3852.0110 ) AS northdiff,
4    ( substr( hospital_gps_location, 31, 10 ) - 11527.0110 ) AS eastdiff
5  FROM
6    hospital_list
7  ORDER BY
8    northdiff ASC,
9    eastdiff ASC
10   LIMIT 1;
```

信息 结果 1 剖析 状态

hospital_name	northdiff	eastdiff
hospital1	0.9999999997489795	0.0000000002037268

Use case 5: Assume that the definition of 'cure time' is the time interval from first detected with the virus to first detected as negative for a patient. Please list the phone number, name and time interval of the patient with the shortest cure time.

➢ <u>SQL statement:</u>

```
select
    a.mobile_no, (timestampdiff(HOUR, b.sample_report_time,
a.sample_report_time)) as mintimediff
from
    patient_cure_information a
left join
    (select * from patient_test_information where sample_result =
"positive") b
```

```
on
    a.mobile_no = b.mobile_no
order by
    mintimediff asc
limit 1;
```

```
SELECT
  a.mobile_no,
  (
  timestampdiff( HOUR, b.sample_report_time, a.sample_report_time )) AS mintimediff
FROM
  patient_cure_information a
  LEFT JOIN ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) b ON a
.mobile_no = b.mobile_no
ORDER BY
  mintimediff ASC
  LIMIT 1;
```

subtract the two times to select the patient with the shortest cure time

➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

3. Test data in Table 1:

mobile_no	sample_report_time (before)	sample_report_time (after)
13252155887	2021-10-10 18:29:39	2021-10-17 18:29:39
233636	2021-10-09 19:30:00	2021-10-26 19:30:00
41265152774	2021-10-24 16:23:14	2021-11-01 11:36:53
47029460746	2021-10-04 19:30:16	2021-10-07 19:30:16
72530750739	2021-10-04 19:00:15	2021-10-11 19:39:15
80766943391	2021-10-04 23:59:11	2021-10-11 23:59:11

4. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the patient cure timetable.

✓ *Introduction to the test data:*
Target data in Table 1 is '47029460746'. And invalid data are '13252155887', '233636', '41265152774', '72530750739', and '80766943391'.

✓ *Reasons for choosing these data:*
In Table 1, 'sample_report_time (before)' represents the time when the patient was first tested positive and 'sample_report_time (after)' represents the time when the patient was cured (first negative after hospitalisation). By comparing 'sample_report_time (before)' and 'sample_report_time (after)', it can be seen that only the patient with the mobile phone number called '47029460746 ' has the shortest healing time of three days (72 hours).

➢ <u>The result of the SELECT statement (screenshot):</u>

```sql
SELECT
  a.mobile_no,
  (
  timestampdiff( HOUR, b.sample_report_time, a.sample_report_time )) AS mintimediff
FROM
  patient_cure_information a
  LEFT JOIN ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) b ON a
.mobile_no = b.mobile_no
ORDER BY
  mintimediff ASC
  LIMIT 1;
```

信息 结果 1 剖析 状态

mobile_no	mintimediff
47029460746	72

Use case 6: According to the needs of epidemic prevention and control, it is now stipulated that: As of '2021-10-11 00:00:00', when a district is at high risk, no one is allowed to enter or leave every base station. When a district is at middle risk, no one will be allowed to enter or leave the district but will be allowed to circulate between base stations. Please list the mobile phone numbers and names of all "high-risk district" violators.

➢ <u>SQL statement:</u>

```sql
select
      mobile_no, name
from
      history_tracking
where
      disconnected_time > "2021-10-11 00:00:00"
and
      district_bs_no = any(
      select
            distinct b.district_bs_no
      from
(select * from patient_test_information where sample_result = "positive") a
left join
            history_tracking b
      on
            a.mobile_no = b.mobile_no
      where
            (b.disconnected_time = "1999-09-09 00:00:00")
      or
timestampdiff(hour, b.connected_time, b.disconnected_time) >= 24
);
```

```sql
SELECT
  mobile_no,
NAME
FROM
  history_tracking
WHERE
  disconnected_time > "2021-10-11 00:00:00"
  AND district_bs_no = ANY (
  SELECT DISTINCT
    b.district_bs_no
  FROM
    ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) a
    LEFT JOIN history_tracking b ON a.mobile_no = b.mobile_no
  WHERE
    ( b.disconnected_time = "1999-09-09 00:00:00" )
  OR timestampdiff( HOUR, b.connected_time, b.disconnected_time ) >= 24
  );
```

select the people who travelled after October 11

select of positive patients in high risk districts

➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	name	connected_time	disconnected_time	district_bs_no
41265152774	Qiuqiuwang	2021-10-08 00:00:00	2021-10-24 15:19:56	C8
41265152774	Qiuqiuwang	2021-10-24 15:19:56	1999-09-09 00:00:00	C7
80766943391	Qiuqiuren	2021-10-10 15:02:38	2021-10-11 01:01:02	N4
80766943391	Qiuqiuren	2021-10-11 01:01:02	1999-09-09 00:00:00	E4
47029460746	Xifenglang	2021-10-03 02:02:02	1999-09-09 00:00:00	C5

2. Test data in Table 2:

district_bs_no	district_name
C5	Centre Lukewarm Hillside
C7	Farrellland
C8	Farrellland
E4	Perryland
N4	Kiwiland

3. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the history tracking table (to record the people who travelled before or after October 11). And Table 2 is the base station table (to record all base stations in high and middle-risk districts, 'Kiwiland' is a middle-risk district and all other districts are high risk).

✓ *Introduction to the test data:*
Target data in Table 1 is 'Qiuqiuwang'. And invalid data are 'Qiuqiuren' and 'Xifenglang'.

✓ *Reasons for choosing these data:*
The data are arranged for two purposes. The first purpose is to detect whether the SQL statement in the subquery about high-risk districts is correct. The second purpose is to exclude all people who do not match the requirement. First, the positive patient named 'Qiuqiuwang' travelled between different base stations in the high-risk districts after October 11 (from base station 'C8' to 'C7'). So, he was the desired one. Then, the person named 'Qiuqiuren' is the violator in the middle-risk district, because his original district is 'Kiwiland' which belongs to the middle-risk district. So, he is the person to be excluded. The last one is a person named 'Xifenglang', who never left his original base station in the high-risk district. He is also a person that needs to be excluded. To conclude, only 'Qiuqiuwang' is the person who meets the requirement.

➢ <u>The result of the SELECT statement (screenshot):</u>

```sql
SELECT
  mobile_no,
NAME
FROM
  history_tracking
WHERE
  disconnected_time > "2021-10-11 00:00:00"
  AND district_bs_no = ANY (
  SELECT DISTINCT
    b.district_bs_no
  FROM
    ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) a
    LEFT JOIN history_tracking b ON a.mobile_no = b.mobile_no
  WHERE
    ( b.disconnected_time = "1999-09-09 00:00:00" )
  OR timestampdiff( HOUR, b.connected_time, b.disconnected_time ) >= 24
  );
```

信息 结果 1 剖析 状态

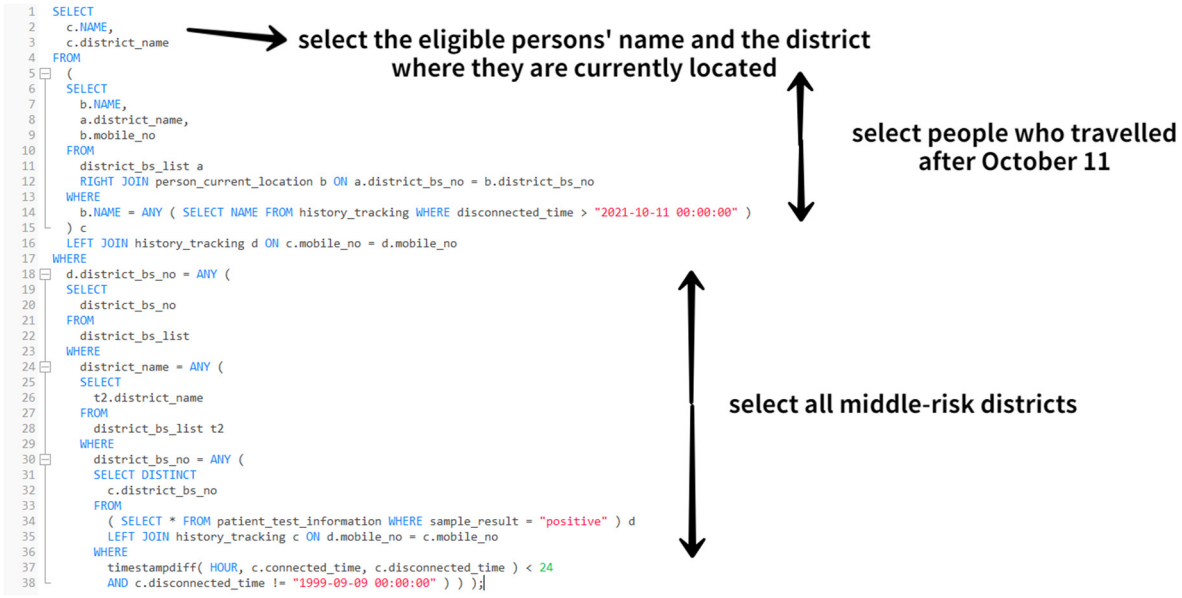
mobile_no	NAME
41265152774	Qiuqiuwa

Use case 7: According to the rule related to the epidemic prevention and control in 'Use case 5', please list the names of all the positive patients from middle-risk districts and their current location (leaving the original base station after October 11).

- SQL statement:

```sql
select
	c.name, c.district_name
from
	(
	select
		b.name, a.district_name, b.mobile_no
	from
		district_bs_list a
	right join
		person_current_location b
	on
		a.district_bs_no = b.district_bs_no
	where
		b.name = any(
		select
			name
		from
			history_tracking
		where
			disconnected_time > "2021-10-11 00:00:00"
		)
	) c
left join
	history_tracking d
on
	c.mobile_no = d.mobile_no
where
	d.district_bs_no = any(
		select
			district_bs_no
		from
			district_bs_list
		where
			district_name = any(
			select
				t2.district_name
```

```
                    from
                          district_bs_list t2
                    where
                          district_bs_no = any(
                          select
                                distinct c.district_bs_no
                          from
(select * from patient_test_information where sample_result = "positive") d
                          left join
                                history_tracking c
                          on
                                d.mobile_no = c.mobile_no
                          where
timestampdiff(hour, c.connected_time, c.disconnected_time) < 24
                          and
                                c.disconnected_time != "1999-09-09 00:00:00"
                          )
              )
       );
```



➢ <u>Test data and why it can prove that the SELECT statement works:</u>

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	name	connected_time	disconnected_time	district_bs_no
80766943391	Qiuqiuren	2021-10-10 15:02:38	2021-10-11 01:01:02	N4
80766943391	Qiuqiuren	2021-10-11 01:01:02	1999-09-09 00:00:00	E4
14352311111	Paul	2021-11-01 18:19:00	2021-11-01 19:19:00	N1
14352311111	Paul	2021-11-01 19:19:00	1999-09-09 00:00:00	N4
233636	Mark	2021-09-01 02:23:11	2021-10-01 12:34:23	C4

233636	Mark	2021-10-09 19:00:00	1999-09-09 00:00:00	C1

2. Test data in Table 2:

district_bs_no	district_name
C1	Godlyland
C4	Farrellland
E4	Perryland
N1	Cookland
N4	Kiwiland

3. Explanations:

✓ *Introduction to the test tables:*

Table 1 is the history tracking table (to record the people who travelled after October 11). And Table 2 is the base station table (to record all base stations in high and middle-risk districts. 'Kiwiland' and 'Cookland' are middle-risk districts. 'Farrellland' is high risk).

✓ *Introduction to the test data:*

Target data in Table 1 are 'Qiuqiuren' and 'Paul'. And invalid data is 'Mark'.

✓ *Reasons for choosing these data:*

The data are arranged for two purposes. The first purpose is to detect whether the SQL statement in the subquery about middle-risk districts is correct. The second purpose is to exclude all people who do not match the requirement. The data in Table 1 are divided into two groups. The first group is the people who meet the requirements. They are 'Qiuqiuren' and 'Paul'. Then, the second group is the excluded person called 'Mark'. Although he left the base station after October 11, his original location was a high-risk district. So, he does not meet the query conditions. To conclude, only 'Qiuqiuren' and 'Paul' will be shown in the result.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT
  c.NAME, c.district_name
FROM
  (SELECT
    b.NAME,
    a.district_name,
    b.mobile_no
  FROM
    district_bs_list a
    RIGHT JOIN person_current_location b ON a.district_bs_no = b.district_bs_no
  WHERE
  b.NAME = ANY ( SELECT NAME FROM history_tracking WHERE disconnected_time > "2021-10-11 00:00:00" )) c
  LEFT JOIN history_tracking d ON c.mobile_no = d.mobile_no
WHERE
  d.district_bs_no = ANY (
  SELECT
    district_bs_no
  FROM
    district_bs_list
  WHERE
    district_name = ANY (
    SELECT
      t2.district_name
    FROM
      district_bs_list t2
    WHERE
      district_bs_no = ANY (
      SELECT DISTINCT
        c.district_bs_no
      FROM
        ( SELECT * FROM patient_test_information WHERE sample_result = "positive" ) d
        LEFT JOIN history_tracking c ON d.mobile_no = c.mobile_no
      WHERE
        timestampdiff( HOUR, c.connected_time, c.disconnected_time ) < 24
      AND c.disconnected_time != "1999-09-09 00:00:00" )));
```

信息 结果 1 剖析 状态

NAME	district_name
Qiuqiuren	Perryland
Paul	Kiwiland

Use case 8: Please list mobile phone numbers of all negative persons who did their viral tests at the same hospital as the positive patients.

➢ SQL statement:

```
select
mobile_no, hospital_no
from
(select *, substr(doctor_id, 7, 2) as hospital_no from
patient_test_information) a
where
a.sample_result = "negative"
and
a.hospital_no = any(select substr(doctor_id, 7, 2) as hospital_no from
patient_cure_information group by hospital_no)
and
mobile_no != any(select mobile_no from patient_cure_information);
```

```
1  SELECT
2    mobile_no,
3    hospital_no
4  FROM
5    ( SELECT *, substr( doctor_id, 7, 2 ) AS hospital_no FROM patient_test_information )
   a
6  WHERE
7    a.sample_result = "negative"
8    AND a.hospital_no = ANY ( SELECT substr( doctor_id, 7, 2 ) AS hospital_no FROM
   patient_cure_information GROUP BY hospital_no )
9    AND a.mobile_no != ANY ( SELECT mobile_no FROM patient_cure_information );
```

select negative patients' mobile phone numbers and the corresponding hospitals

select the hospital where the positive case was detected

➢ Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_result	hospital_id
80766943391	positive	04
47029460746	positive	05
72530750739	positive	06
233636	positive	01
13252155887	positive	01
41265152774	positive	05

2. Test data in Table 2:

mobile_no	sample_result	hospital_id
13626192177	negative	01
70915870657	negative	01
12555812470	negative	01
80766943391	negative	04
16406019403	negative	05
15305045857	negative	06
81918316107	negative	05
1763059086	negative	05

79059729316	negative	01
90747767631	negative	02
93209071135	negative	03

3. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the positive patient table (the reference table). And Table 2 is the negative patient table (the main table).

✓ *Introduction to the test data:*
In Table 2, the target data are the first 9 records. And invalid data are '90747767631' and '93209071135'.

✓ *Reasons for choosing these data:*
The data in Table 1 are required to be completed in the first step of writing the SQL statement, so they can detect whether the subquery is correct. The data in Table 2 that needs to be excluded are the records with mobile phone numbers called '90747767631' and '93209071135' (the last two ones). Because the hospital named '02' and '03' is not listed in Table 1.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
1 SELECT
2   mobile_no,
3   hospital_no
4 FROM
5   ( SELECT *, substr( doctor_id, 7, 2 ) AS hospital_no FROM patient_test_information )
  a
6 WHERE
7   a.sample_result = "negative"
8   AND a.hospital_no = ANY ( SELECT substr( doctor_id, 7, 2 ) AS hospital_no FROM
  patient_cure_information GROUP BY hospital_no )
9   AND mobile_no != ANY ( SELECT mobile_no FROM patient_cure_information );
```

Message | Summary | Result 1 | Profile | Status

mobile_no	hospital_no
13626192177	01
70915870657	01
12555812470	01
80766943391	04
16406019403	05
15305045857	06
81918316107	05
1763059086	05
79059729316	01

Use case 9: Please use the information in 'Use case 7' (the use case given by teachers, not this extended case) and list history tracking of patients who were tested positive on '2021-10-04' in the past 14 days (which districts positive patients have visited).

➢ <u>SQL statement:</u>

```
select
      a.mobile_no, b.district_name
from
      (
      select
            mobile_no, district_bs_no
      from
            history_tracking
```

```
	where
		mobile_no = any(
		select
			mobile_no
		from
			(select
				m.*, n.district_bs_no
			from
				(select
					*, substr(doctor_id, 7, 2) as hospital_no
				from
					patient_test_information
				) m
			left join
				hospital_list n
			on
				m.hospital_no = n.hospital_id
			) t1
		where
t1.sample_result = "positive" and sample_report_time like "2021-10-04
__:__:__"
		and
			t1.district_bs_no = any(
			select
				district_bs_no
			from
				district_bs_list
			where
				district_name = "Centre Lukewarm Hillside"
			)
		)
	and
		disconnected_time != "1999-09-09 00:00:00"
	) a
left join
	district_bs_list b
on
	a. district_bs_no = b.district_bs_no;
```

```
SELECT
  a.mobile_no,
  b.district_name
FROM
  (SELECT
    mobile_no,
    district_bs_no
  FROM
    history_tracking
  WHERE
    mobile_no = ANY (SELECT
      mobile_no
    FROM
      (SELECT
        m.*,
        n.district_bs_no
      FROM
        ( SELECT *, substr( doctor_id, 7, 2 ) AS hospital_no FROM patient_test_information ) m
        LEFT JOIN hospital_list n ON m.hospital_no = n.hospital_id
      ) t1
    WHERE
      t1.sample_result = "positive"
      AND sample_report_time LIKE "2021-10-04 __:__:__"
      AND t1.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE district_name = "Centre Lukewarm Hillside" )
  )
  AND disconnected_time != "1999-09-09 00:00:00"
) a
LEFT JOIN district_bs_list b ON a.district_bs_no = b.district_bs_no;
```

select the history tracking of the positive patients

select patients who were tested positive on '2021-10-04' in 'Centre Lukewarm Hillside'

- Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_result	sample_report_time
47029460746	positive	2021-10-04 19:30:16
72530750739	positive	2021-10-04 19:00:15
41265152774	positive	2021-10-24 16:23:14

2. Test data in Table 2:

mobile_no	district_bs_no
47029460746	W1
72530750739	W5
41265152774	C6

3. Explanations:

✓ *Introduction to the test tables:*
Table 1 is the positive patient table (the reference table). And Table 2 is the history tracking table (the main table).

✓ *Introduction to the test data:*
In Table 2, the target data are the first 2 records. And invalid data is '41265152774'.

✓ *Reasons for choosing these data:*
The reason for arranging these data was mainly to test whether the SQL statement could select the history trips of the patient who was tested positive on October 4. So, as long as the patients are negative or not tested positive on October 4, they will be excluded from consideration.

➢ <u>The result of the SELECT statement (screenshot):</u>

```
SELECT
  a.mobile_no,
  b.district_name
FROM
  (SELECT
    mobile_no,
    district_bs_no
  FROM
    history_tracking
  WHERE
    mobile_no = ANY (SELECT
      mobile_no
    FROM
      (SELECT
        m.*,
        n.district_bs_no
      FROM
        ( SELECT *, substr( doctor_id, 7, 2 ) AS hospital_no FROM patient_test_information ) m
        LEFT JOIN hospital_list n ON m.hospital_no = n.hospital_id
      ) t1
    WHERE
      t1.sample_result = "positive"
      AND sample_report_time LIKE "2021-10-04 __:__:__"
      AND t1.district_bs_no = ANY ( SELECT district_bs_no FROM district_bs_list WHERE district_name = "Centre Lukewarm Hillside" )
    )
    AND disconnected_time != "1999-09-09 00:00:00"
  ) a
  LEFT JOIN district_bs_list b ON a.district_bs_no = b.district_bs_no;
```

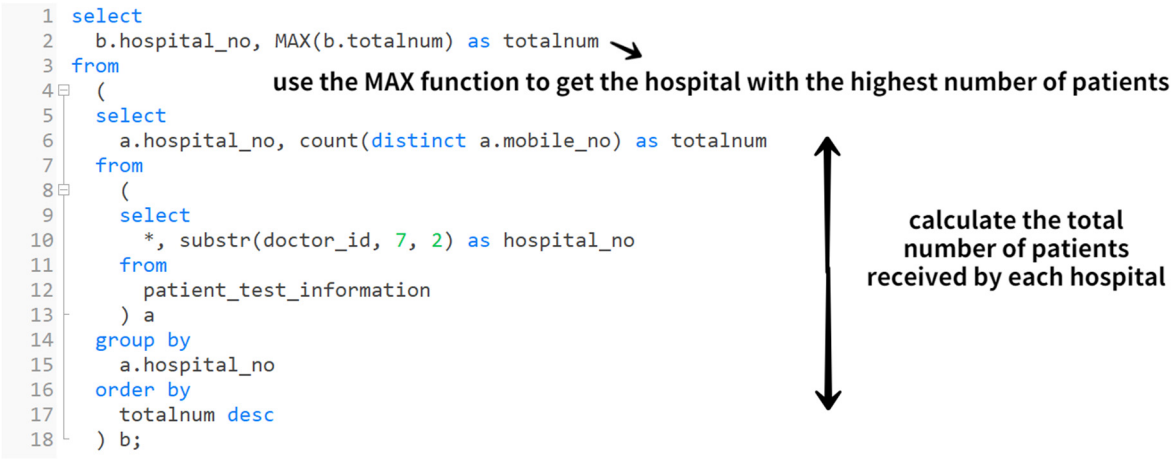
信息 结果 1 剖析 状态

mobile_no	district_name
47029460746	Leslieland
72530750739	Earlland

Use case 10: Please select the hospital that receives the most patients.

- SQL statement:

```
select
      b.hospital_no, MAX(b.totalnum) as totalnum
from
      (
      select
            a.hospital_no, count(distinct a.mobile_no) as totalnum
      from
            (
            select
                  *, substr(doctor_id, 7, 2) as hospital_no
            from
                  patient_test_information
            ) a
      group by
            a.hospital_no
      order by
            totalnum desc
      ) b;
```



- Test data and why it can prove that the SELECT statement works:

The useful information is added to these tables (some attributes are hidden as they are not related to this task)

1. Test data in Table 1:

mobile_no	sample_result	doctor_id
13626192177	negative	1100600101
70915870657	negative	1100600101
12555812470	negative	1100600101
233636	positive	1100600101
13252155887	positive	1100600101
79059729316	negative	1100600101
80766943391	negative	1200900406
80766943391	positive	1200900406

2. Explanations:

✓ *Introduction to the test tables:*

Table 1 is the viral test table.

- ✓ *Introduction to the test data:*
  In Table 1, the target data is the 'doctor_id' called '1100600101'. And invalid data is '1200900406' (the serial number of each hospital can be determined by each 'doctor_id').

- ✓ *Reasons for choosing these data:*
  These data can be counted by the COUNT function and selected by the MAX function in the SQL statement. So, the more records stored in the table, the more superior the use of the aggregation function can be.

- ➢ <u>The result of the SELECT statement (screenshot):</u>

```sql
select
  b.hospital_no, MAX(b.totalnum) as totalnum
from
  (
  select
    a.hospital_no, count(distinct a.mobile_no) as totalnum
  from
    (
    select
      *, substr(doctor_id, 7, 2) as hospital_no
    from
      patient_test_information
    ) a
  group by
    a.hospital_no
  order by
    totalnum desc
  ) b;
```

信息 结果 1 剖析 状态

hospital_no	totalnum
01	6