

CS5200 Fall 2020: Practicum 1

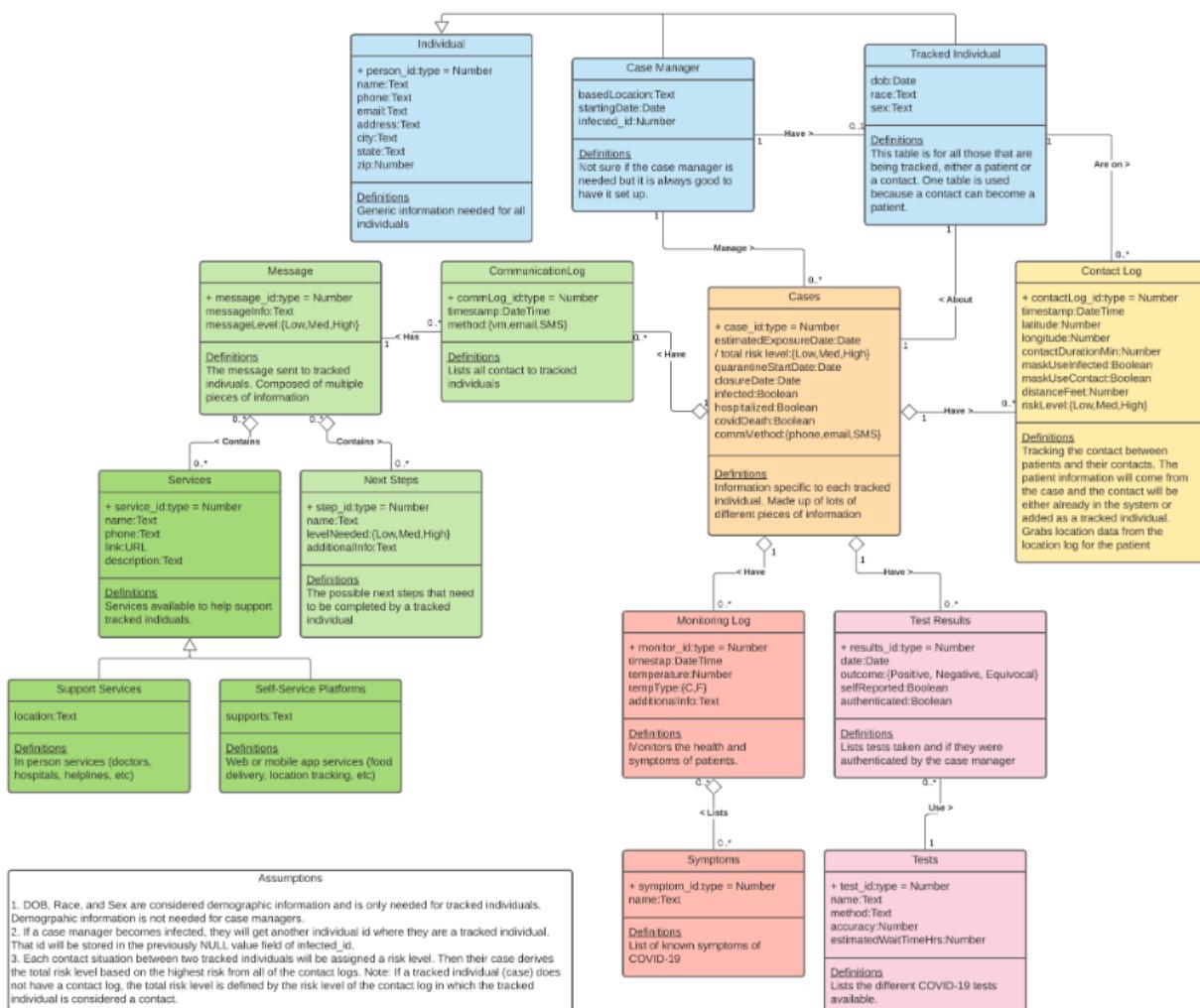
Chandra Davis, Evan Douglass

The steps we completed for Practicum 1 are detailed below. Please note that there are links to each image that requires it in each section. We decided to focus our attention on the case management aspect of the contract tracing problem.

Conceptual Model: UML

View the conceptual model in Lucid Chart here:

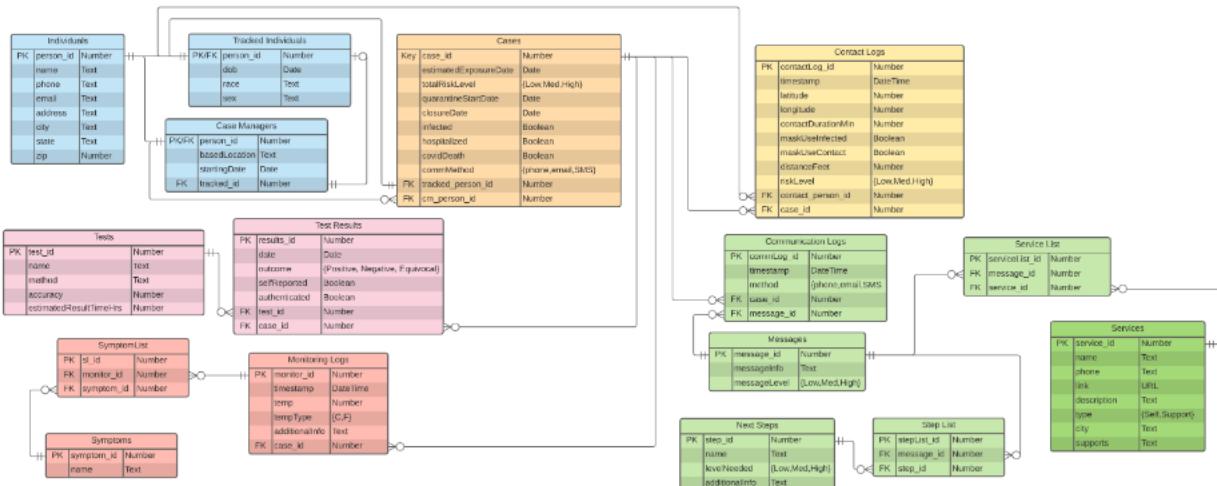
<https://app.lucidchart.com/invitations/accept/5602312e-dfc3-4423-975c-47190ce6022e>



Logical Model: ERD

View the logical model in Lucid Chart here:

<https://app.lucidchart.com/invitations/accept/7b497cbf-268d-4a03-b1a7-822b5a844fea>



Schema

View the schema in Google Docs here:

https://docs.google.com/document/d/1o8pk51aed3BJSaBcwO2EMbT8I3ru_wpGTcTN4W-DbIM/edit?usp=sharing

Individuals(person_id: Number, name: Text, phone: Text, email: Text, address: Text, city: Text, state: Text, zip: Number)

TrackedIndividuals(person_id: Number, dob: Date, race: Text, sex: Text)

CaseManagers(person_id: Number, basedCity: Text, basedState: Text, startingDate: Date, infected_id: Number)

Tests(test_id: Number, name: Text, method: Text, accuracy: Number, estimatedResultTimeHrs: Number)

TestResults(results_id: Number, date: Date, outcome: {Positive, Negative, Equivocal}, selfReported: Boolean, authenticated: Boolean, test_id: Number, case_id: Number)

MonitoringLogs(monitor_id: Number, timestamp: DateTime, temp: Number, tempType: {C, F}, additionalInfo: Text, case_id: Number)

SymptomList(sl_id: Number, monitor_id: Number, symptom_id: Number)

Symptoms(symptom_id: Number, name: Text)

Cases(case_id: Number, estimatedExposureDate: Date, totalRiskLevel: {Low, Med, High}, quarantineStartDate: Date, closureDate: Date, infected: Boolean, hospitalized: Boolean, covidDeath: Boolean, commMethod: {phone, email, SMS}, cm_person_id: Number, infected_person_id: Number)

ContactLogs(contactLog_id: Number, timestamp: DateTime, latitude: Number, longitude: Number, contactDurationMin: Number, maskUseInfected: Boolean, maskUseContact: Boolean, distanceFeet: Number, riskLevel: {Low, Med, High}, contact_person_id: Number, case_id: Number)

CommunicationLogs(commLog_id: Number, timestamp: DateTime, method: {phone, email, SMS}, case_id: Number, message_id: Number)

Messages(message_id: Number, messageLevel: {Low, Med, High}, messageInfo: Text)

StepList(stepList_id: Number, message_id: Number, step_id: Number)

NextSteps(step_id: Number, name: Text, levelNeeded: {Low, Med, High}, additionalInfo: Text)

ServiceList(serviceList_id: Number, message_id: Number, service_id: Number)

Services(service_id: Number, name: Text, phone: Text, link: Text, description: Text, type: {Self, Support}, city: Text, supports: Text)

Creating Database Tables

Should you wish to inspect the scripts that create the database and populate data, they can be found at:

<https://github.com/eldss-classwork/databases-practicum1-scripts>

The following images will show a progression from an empty database through table creation in MySQL Workbench.

The MySQL Workbench start screen.

Welcome to MySQL Workbench

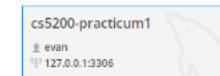
MySQL Workbench is the official graphical user interface (GUI) tool for MySQL. It allows you to design, create and browse your database schemas, work with database objects and insert data as well as design and run SQL queries to work with stored data. You can also migrate schemas and data from other database vendors to your MySQL database.

[Browse Documentation >](#)

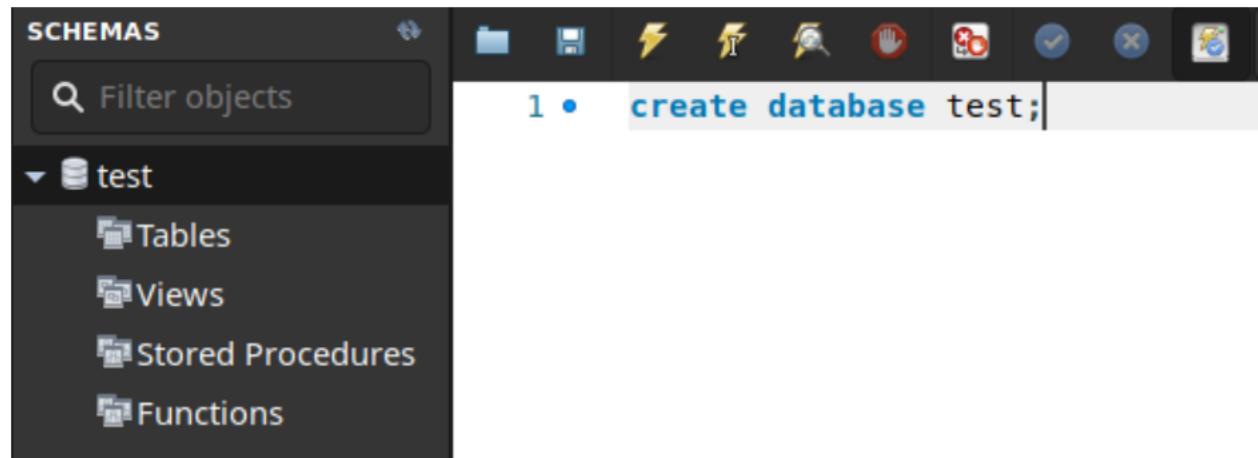
[Read the Blog >](#)

[Discuss on the Forums >](#)

MySQL Connections @@



The newly created, empty `test` database.



The **test** database after table creation.

The screenshot shows the MySQL Workbench interface. On the left, the 'SCHEMAS' tree view is expanded to show the 'test' schema, which contains tables such as CaseManagers, Cases, CommunicationLogs, CommunicationMethods, ContactLogs, Individuals, Messages, MonitoringLogs, NextSteps, PossibleTestOutcomes, RiskLevels, ServiceList, Services, StepList, SymptomList, Symptoms, TestResults, Tests, and TrackedIndividuals. Below the schema tree, there are sections for Views, Stored Procedures, and Functions. On the right, a large text area displays a SQL script for creating the 'Individuals' table. The script starts with comments about dropping existing tables and setting foreign key checks to 0. It then creates the 'Individuals' table with columns for person_id (primary key, auto-increment), name (text, not null), phone (text), email (text), address (text), city (text), and state (text). The script concludes by setting foreign key checks back to 1. The status bar at the bottom indicates 'Added new script editor'.

```
1 -- This script provides statements to build the tables required
2 -- of a contact tracing database.
3
4 -- Start by dropping tables
5 • SET foreign_key_checks = 0;
6 • DROP TABLE IF EXISTS `ContactLogs`;
7 • DROP TABLE IF EXISTS `CommunicationLogs`;
8 • DROP TABLE IF EXISTS `ServiceList`;
9 • DROP TABLE IF EXISTS `Services`;
10 • DROP TABLE IF EXISTS `StepList`;
11 • DROP TABLE IF EXISTS `Individuals`;
12 • DROP TABLE IF EXISTS `NextSteps`;
13 • DROP TABLE IF EXISTS `CaseManagers`;
14 • DROP TABLE IF EXISTS `Cases`;
15 • DROP TABLE IF EXISTS `TrackedIndividuals`;
16 • DROP TABLE IF EXISTS `Messages`;
17 • DROP TABLE IF EXISTS `SymptomList`;
18 • DROP TABLE IF EXISTS `Symptoms`;
19 • DROP TABLE IF EXISTS `MonitoringLogs`;
20 • DROP TABLE IF EXISTS `TestResults`;
21 • DROP TABLE IF EXISTS `PossibleTestOutcomes`;
22 • DROP TABLE IF EXISTS `Tests`;
23 • DROP TABLE IF EXISTS `CommunicationMethods`;
24 • DROP TABLE IF EXISTS `RiskLevels`;
25 • SET foreign_key_checks = 1;
26
27 -- Recreate the tables
28 • CREATE TABLE `Individuals` (
29     `person_id` Int PRIMARY KEY auto_increment,
30     `name` Text NOT NULL,
31     -- Tracked people demographics are voluntary
32     `phone` Text,
33     `email` Text,
34     `address` Text,
35     `city` Text,
36     `state` Text.
```

The following photos provide a detailed look at the schema of each table as it was created in MySQL.

1 **DESCRIBE CaseManagers;**

#	Field	Type	Null	Key	Default	Extra
1	person_id	int(11)	NO	PRI	NULL	
2	basedCity	text	NO		NULL	
3	basedState	text	NO		NULL	
4	startingDate	date	NO		NULL	
5	infected_id	int(11)	YES	UNI	NULL	

1 **DESCRIBE Cases;**

#	Field	Type	Null	Key	Default	Extra
1	estimatedExposureDate	date	NO		NULL	
2	quarantineStartDate	date	YES		NULL	
3	closureDate	date	YES		NULL	
4	case_id	int(11)	NO	PRI	NULL	auto_increment
5	totalRiskLevel_id	int(11)	NO	MUL	NULL	
6	commMethod_id	int(11)	NO	MUL	NULL	
7	cm_person_id	int(11)	NO	MUL	NULL	
8	tracked_person_id	int(11)	NO	UNI	NULL	
9	infected	tinyint(1)	YES		0	
10	hospitalized	tinyint(1)	YES		0	
11	covidDeath	tinyint(1)	YES		0	

1 DESCRIBE CommunicationLogs;

#	Field	Type	Null	Key	Default	Extra
1	commLog_id	int(11)	NO	PRI	NULL	auto_increment
2	timestamp	datetime	NO		NULL	
3	method_id	int(11)	NO	MUL	NULL	
4	case_id	int(11)	NO	MUL	NULL	
5	message_id	int(11)	NO	MUL	NULL	

1 DESCRIBE CommunicationMethods;

#	Field	Type	Null	Key	Default	Extra
1	commMethod_id	int(11)	NO	PRI	NULL	auto_increment
2	commMethod	text	NO	UNI	NULL	

1 DESCRIBE ContactLogs;

#	Field	Type	Null	Key	Default	Extra
1	contactLog_id	int(11)	NO	PRI	NULL	auto_increment
2	timestamp	datetime	NO		NULL	
3	latitude	decimal...	NO		NULL	
4	longitude	decimal...	NO		NULL	
5	contactDurationMin	int(11)	NO		NULL	
6	maskUseInfected	tinyint(1)	NO		NULL	
7	maskUseContact	tinyint(1)	NO		NULL	
8	distanceFeet	int(11)	NO		NULL	
9	risk_id	int(11)	NO	MUL	NULL	
10	contact_person_id	int(11)	NO	MUL	NULL	
11	case_id	int(11)	NO	MUL	NULL	

1 DESCRIBE Individuals;

#	Field	Type	Null	Key	Default	Extra
1	person_id	int(11)	NO	PRI	NULL	auto_increment
2	name	text	NO		NULL	
3	phone	text	YES		NULL	
4	email	text	YES		NULL	
5	address	text	YES		NULL	
6	city	text	YES		NULL	
7	state	text	YES		NULL	
8	zip	int(11)	YES		NULL	

1 DESCRIBE Messages;

#	Field	Type	Null	Key	Default	Extra
1	message_id	int(11)	NO	PRI	NULL	auto_increment
2	messageLevel_id	int(11)	NO	MUL	NULL	
3	messagelInfo	text	NO		NULL	

1 DESCRIBE MonitoringLogs;

#	Field	Type	Null	Key	Default	Extra
1	monitor_id	int(11)	NO	PRI	NULL	auto_increment
2	timestamp	datetime	NO		NULL	
3	temp	decimal...	NO		NULL	
4	tempType	text	NO		NULL	
5	additionalInfo	text	YES		NULL	
6	case_id	int(11)	NO	MUL	NULL	

1 DESCRIBE NextSteps;

#	Field	Type	Null	Key	Default	Extra
1	step_id	int(11)	NO	PRI	NULL	auto_increment
2	name	text	NO		NULL	
3	levelNeeded_id	int(11)	NO	MUL	NULL	
4	additionalInfo	text	NO		NULL	

1 DESCRIBE PossibleTestOutcomes;

#	Field	Type	Null	Key	Default	Extra
1	outcome_id	int(11)	NO	PRI	NULL	auto_increment
2	outcome	text	NO		NULL	

1 DESCRIBE RiskLevels;

#	Field	Type	Null	Key	Default	Extra
1	risk_id	int(11)	NO	PRI	NULL	auto_increment
2	risk	text	NO	UNI	NULL	

1 DESCRIBE ServiceList;

#	Field	Type	Null	Key	Default	Extra
1	serviceList_id	int(11)	NO	PRI	NULL	auto_increment
2	message_id	int(11)	NO	MUL	NULL	
3	service_id	int(11)	NO	MUL	NULL	

1 DESCRIBE Services;

#	Field	Type	Null	Key	Default	Extra
1	service_id	int(11)	NO	PRI	NULL	auto_increment
2	name	text	NO		NULL	
3	phone	text	NO		NULL	
4	link	text	NO		NULL	
5	description	text	NO		NULL	
6	type	text	NO		NULL	
7	city	text	NO		NULL	
8	supports	text	NO		NULL	

1 DESCRIBE StepList;

#	Field	Type	Null	Key	Default	Extra
1	stepList_id	int(11)	NO	PRI	NULL	auto_increment
2	message_id	int(11)	NO	MUL	NULL	
3	step_id	int(11)	NO	MUL	NULL	

1 DESCRIBE SymptomList;

#	Field	Type	Null	Key	Default	Extra
1	sl_id	int(11)	NO	PRI	NULL	auto_increment
2	monitor_id	int(11)	NO	MUL	NULL	
3	symptom_id	int(11)	NO	MUL	NULL	

1 DESCRIBE Symptoms;

#	Field	Type	Null	Key	Default	Extra
1	symptom_id	int(11)	NO	PRI	NULL	auto_increment
2	name	text	NO		NULL	

```
1 DESCRIBE TestResults;
```

#	Field	Type	Null	Key	Default	Extra
1	results_id	int(11)	NO	PRI	NULL	auto_increment
2	date	date	NO		NULL	
3	outcome_id	int(11)	NO	MUL	NULL	
4	selfReported	tinyint(1)	NO		NULL	
5	authenticated	tinyint(1)	NO		0	
6	test_id	int(11)	NO	MUL	NULL	
7	case_id	int(11)	NO	MUL	NULL	

```
1 DESCRIBE Tests;
```

#	Field	Type	Null	Key	Default	Extra
1	test_id	int(11)	NO	PRI	NULL	auto_increment
2	name	text	NO	UNI	NULL	
3	method	text	NO		NULL	
4	accuracy	decimal...	NO		NULL	
5	estimatedResultTimeHrs	int(11)	NO		NULL	

```
1 DESCRIBE TrackedIndividuals;
```

#	Field	Type	Null	Key	Default	Extra
1	person_id	int(11)	NO	PRI	NULL	
2	dob	date	YES		NULL	
3	race	text	YES		NULL	
4	sex	text	YES		NULL	

Populating The Database

The script for populating data into our database can be found at the following link:

<https://github.com/eldss-classwork/databases-practicum1-scripts/blob/master/populate.sql>

Demonstrations that the data was loaded correctly can be found in the queries below. That is, they return the data that was loaded.

Queries

The script for running all of the queries can be found at the following link:

<https://github.com/eldss-classwork/databases-practicum1-scripts/blob/master/queries.sql>

The following is the screenshots of the query results.

Query 1

```
1 -- How many communications have been sent to people in each exposure level group?
2 • SELECT r.risk, COUNT(cl.`timestamp`) AS numCommunications
3   FROM CommunicationLogs AS cl, Cases AS c, RiskLevels AS r
4   WHERE cl.case_id=c.case_id AND c.totalRiskLevel_id=r.risk_id
5   GROUP BY r.risk
6   ORDER BY r.risk_id;
```

#	risk	numCommunications
1	Low	39
2	Medium	32
3	High	29

Query 2

```
1 -- List the number of COVID deaths by state, in descending order of deaths.
2 • SELECT i.state, COUNT(*) AS deaths
3   FROM Cases AS c, Individuals AS i
4   WHERE c.tracked_person_id=i.person_id AND c.covidDeath=1
5   GROUP BY i.state
6   ORDER BY deaths DESC;
```

#	state	deaths
1	Imo	3
2	São Paulo	1
3	Gye	1

Query 3

```
1 -- List the individual name, type of test, test method, test result,
2 -- and date of test for all recorded tests; order by date taken.
3 • SELECT i.name, t.name, t.method, pto.outcome, tr.date
4 FROM Individuals AS i, Tests AS t, TestResults AS tr, Cases AS c, PossibleTestOutcomes AS pto
5 WHERE
6     tr.test_id=t.test_id
7     AND tr.case_id=c.case_id
8     AND tr.outcome_id=pto.outcome_id
9     AND c.tracked_person_id=i.person_id
10 ORDER BY tr.date;
```

#	name	name	method	outcome	date
1	Gareth Flores	Experimental Test 4	Nasal Swab	Positive	2020-02-05
2	Gareth Flores	Experimental Test 3	Nasal Swab	Negative	2020-02-28
3	Bradley Rocha	Experimental Test 3	Nasal Swab	Negative	2020-03-15
4	Ciaran Jacobs	Experimental Test 3	Nasal Swab	Positive	2020-03-22
5	Bradley Rocha	Experimental Test 2	Cheek Swab	Positive	2020-03-24
6	Bradley Rocha	Experimental Test 3	Nasal Swab	Negative	2020-03-28
7	Macy Wise	Experimental Test 1	Nasal Swab	Positive	2020-04-04
8	Knox Henderson	Experimental Test 3	Nasal Swab	Negative	2020-04-04
9	Macy Wise	Experimental Test 4	Nasal Swab	Negative	2020-05-03
10	Elmo Burns	Experimental Test 2	Cheek Swab	Positive	2020-05-05
11	Knox Henderson	Experimental Test 1	Nasal Swab	Negative	2020-05-13
12	Elmo Burns	Experimental Test 3	Nasal Swab	Negative	2020-06-09
13	Bertha Acosta	Experimental Test 2	Cheek Swab	Negative	2020-06-30
14	Bertha Acosta	Experimental Test 1	Nasal Swab	Negative	2020-07-09
15	Murphy Leon	Experimental Test 2	Cheek Swab	Positive	2020-07-15
16	Lydia Ramirez	Experimental Test 4	Nasal Swab	Positive	2020-08-06
17	Carl Perry	Experimental Test 3	Nasal Swab	Positive	2020-08-18
18	Lydia Ramirez	Experimental Test 2	Cheek Swab	Negative	2020-09-17
19	Carl Perry	Experimental Test 1	Nasal Swab	Positive	2020-09-18
20	Carl Perry	Experimental Test 2	Cheek Swab	Positive	2020-09-29

Query 4

```
1 -- Get the name, phone, and email of individuals who did not provide
2 -- daily check-ins while in quarantine (14 days).
3 • SELECT i.name, i.phone, i.email, COUNT(*) AS `messages sent`
4 FROM Individuals AS i, Cases AS c, MonitoringLogs AS ml
5 WHERE
6     c.tracked_person_id=i.person_id
7     AND ml.case_id=c.case_id
8     AND c.closureDate IS NOT NULL
9 GROUP BY i.name
10 HAVING `messages sent` < 14;
```

#	name	phone	email	messages sent
1	Bertha Acosta	(640) 327-1743	rutrurum@sitamet.org	11
2	Carl Perry	(426) 177-3217	orci.lacus.vestibulum@euismodma...	8
3	Ciaran Jacobs	(960) 864-9603	Ut.tincidunt.orci@vel.co.uk	10
4	Elmo Burns	(521) 702-4435	Donec.at@in.edu	11
5	Gareth Flores	(140) 387-5572	facilisi.Sed@Quisque.org	9
6	Knox Henderson	(377) 684-0047	sodales.elit.erat@aliquetdiamSed.ca	6
7	Lydia Ramirez	(309) 418-8256	taciti.sociosqu.ad@odioPhasellusat...	9
8	Macy Wise	(878) 442-1017	dolor.dolor@Nullainterduum.net	13
9	Murphy Leon	(327) 972-7628	vitae.purus.gravida@feugiatnec.com	8

Query 5

- Get the id and number of cases assigned to each case manager.

The screenshot shows the MySQL Workbench interface with two query panes and a results pane.

Query 1:

```

1 • SELECT cn.person_id AS CM_ID, CASE WHEN CaseCount IS NULL THEN 0 ELSE CaseCount END AS CaseCount
2   FROM casemanagers cn
3   LEFT JOIN (SELECT cn_person_id,COUNT(cn_person_id) AS CaseCount FROM cases GROUP BY cn_person_id) counts
4     ON cn_person_id = cn.person_id
    
```

Result Grid:

CM_ID	CaseCount
96	6
97	5
98	5
99	6
100	0

Query 2:

Action Output:

- 144 15:42:53 INSERT INTO ServiceLog (message_id, service_id) VALUES (0,5),(5,0),(0,0),(4,3) 5 rows affected Records: 5 Duplicates: 0 Warnings: 0 Duration / Item: 0.000 sec
- 145 15:42:53 INSERT INTO ServiceLog (message_id, service_id) VALUES (4,4),(4,5),(5,5),(7,4),(6,6) 5 rows affected Records: 5 Duplicates: 0 Warnings: 0 Duration / Item: 0.000 sec
- 146 15:42:53 INSERT INTO ServiceLog (message_id, service_id) VALUES (2,4),(4,7),(7,3),(1),(6,1),(3,1) 6 rows affected Records: 6 Duplicates: 0 Warnings: 0 Duration / Item: 0.000 sec
- 147 16:01:13 SELECT cases.case_id tracked_person_id,infected,closedate,timestamp FROM cases JOIN contactlogs ON tra... 2 rows returned Duration / Item: 0.015 sec / 0.000 sec
- 148 16:01:45 SELECT cn.person_id AS CM_ID,CASE WHEN CaseCount IS NULL THEN 0 ELSE CaseCount END AS CaseCount... 5 rows returned Duration / Item: 0.000 sec / 0.000 sec

Result 2:

Output:

Action Output:

- 144 15:42:53 INSERT INTO ServiceLog (message_id, service_id) VALUES (0,5),(5,0),(0,0),(4,3) 5 rows affected Records: 5 Duplicates: 0 Warnings: 0 Duration / Item: 0.000 sec
- 145 15:42:53 INSERT INTO ServiceLog (message_id, service_id) VALUES (4,4),(4,5),(5,5),(7,4),(6,6) 5 rows affected Records: 5 Duplicates: 0 Warnings: 0 Duration / Item: 0.000 sec
- 146 15:42:53 INSERT INTO ServiceLog (message_id, service_id) VALUES (2,4),(4,7),(7,3),(1),(6,1),(3,1) 6 rows affected Records: 6 Duplicates: 0 Warnings: 0 Duration / Item: 0.000 sec
- 147 16:01:13 SELECT cases.case_id tracked_person_id,infected,closedate,timestamp FROM cases JOIN contactlogs ON tra... 2 rows returned Duration / Item: 0.015 sec / 0.000 sec
- 148 16:01:45 SELECT cn.person_id AS CM_ID,CASE WHEN CaseCount IS NULL THEN 0 ELSE CaseCount END AS CaseCount... 5 rows returned Duration / Item: 0.000 sec / 0.000 sec

Query 6

- Get the id, name, phone, and email of individuals who have been in contact with a tracked individual but do not have a case yet.

The screenshot shows the MySQL Workbench interface with two query panes and a results pane.

Query 1:

```

1 • SELECT person_id, ind.name, ind.phone, ind.email
2   FROM individuals ind
3   JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id
4     FROM contactlogs
5     LEFT JOIN cases ON contact_person_id = tracked_person_id WHERE cases.case_id IS NULL) logged
6     ON person_id = contact
    
```

Result Grid:

person_id	name	phone	email
15	Lane Branson	(102) 674-0287	Integer.product@maleadagae.com
23	Jonah Chen	(708) 770-0255	Chen.eulogia@solitairesquarrel.edu
25	Dominic Penny	(6,9) 554-0302	wouter@neelworts.co.uk
32	Vernon Schwartz	(5,8) 206-5796	neque@utem.org
33	Wyatt Orr	(7,6) 613-7739	Wealthy@orenheviaur.com
34	Marcus Sanderson	(2,8) 622-8581	melius@mavamun.co.uk
36	Stewart Thomas	(5,8) 568-7056	equitas@ermitac.net
43	Emmale Parker	(2,9) 859-7341	paul@primitiv.ca
44	Lorena Hayden	(6,1) 766-5887	ncc@Mauris@sturdy.co.uk
50	Maxwell Hartman	(9,8) 696-1168	qua.dan.Palentosque@ante@econasmil.com
52	Reed Didson	(8,4) 257-3396	justo@modul.com
58	Brianna Eaton	(7,2) 432-0970	pelentesque@nkoac.net
59	Dana Sawyer	(4,0) 464-4451	neantid@att@integer.net

Query 2:

Action Output:

- 147 16:01:13 SELECT cases.case_id tracked_person_id,infected,closedate,timestamp FROM cases JOIN contactlogs ON tra... 2 rows returned Duration / Item: 0.015 sec / 0.000 sec
- 148 16:01:45 SELECT person_id AS CM_ID,CASE WHEN CaseCount IS NULL THEN 0 ELSE CaseCount END AS CaseCount... 5 rows returned Duration / Item: 0.000 sec / 0.000 sec
- 149 16:01:12 SELECT person_id,ind.name,ind.phone,ind.email,cse_id FROM individuals ind JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id... 15 rows returned Duration / Item: 0.016 sec / 0.000 sec
- 150 16:05:27 SELECT person_id,ind.name,ind.phone,ind.email FROM individuals ind JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id... 15 rows returned Duration / Item: 0.000 sec / 0.000 sec
- 151 16:07:20 SELECT person_id,ind.name,ind.phone,ind.email FROM individuals ind JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id... 15 rows returned Duration / Item: 0.000 sec / 0.000 sec

Result 2:

Action Output:

- 147 16:01:13 SELECT cases.case_id tracked_person_id,infected,closedate,timestamp FROM cases JOIN contactlogs ON tra... 2 rows returned Duration / Item: 0.015 sec / 0.000 sec
- 148 16:01:45 SELECT person_id AS CM_ID,CASE WHEN CaseCount IS NULL THEN 0 ELSE CaseCount END AS CaseCount... 5 rows returned Duration / Item: 0.000 sec / 0.000 sec
- 149 16:01:12 SELECT person_id,ind.name,ind.phone,ind.email,cse_id FROM individuals ind JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id... 15 rows returned Duration / Item: 0.016 sec / 0.000 sec
- 150 16:05:27 SELECT person_id,ind.name,ind.phone,ind.email FROM individuals ind JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id... 15 rows returned Duration / Item: 0.000 sec / 0.000 sec
- 151 16:07:20 SELECT person_id,ind.name,ind.phone,ind.email FROM individuals ind JOIN (SELECT DISTINCT(contact_person_id) AS contact, cases.case_id... 15 rows returned Duration / Item: 0.000 sec / 0.000 sec

Query 7

- Get the case, name, preferred communication method, contact information, and case closure date for tracked individuals who have a temperature over 99 degrees F and are experiencing a cough or breathing problems.

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query for Query 7, which joins multiple tables to find cases with temperatures over 99°F and symptoms like cough or breathing problems, along with their communication methods and contact info.
- Result Grid:** Displays the results of the query, showing columns: case_id, name, commMethod, contactInfo, and closureDate. The results list six individuals with their respective details.
- Action Output:** Shows the execution log with 15 rows returned for each of the five SELECT statements in the query.

```

1 • SELECT cases.case_id, ind.name, commMethod,
2     CASE WHEN commMethod='Email' THEN email ELSE phone END AS contactInfo, closureDate
3     FROM cases
4     JOIN (SELECT DISTINCT(case_id) FROM monitoringlogs ml
5           JOIN symptomlist sl ON sl.monitor_id = ml.monitor_id
6           WHERE temp > 99 AND tempType = 'F' AND symptom_id IN (3,4,5)) log
7     ON cases.case_id = log.case_id
8     JOIN communicationmethods cm ON cases.commMethod_id = cm.commMethod_id
9     JOIN individuals ind ON tracked_person_id = person_id

```

case_id	name	commMethod	contactInfo	closureDate
9	Lyle Ramirez	Email	lcooper@roaring-edifice.com	2020-05-07
3	Mona Vilas	Email	dolor dolor@antibacterium.net	2020-04-02
10	Cef Perry	Phone	(408) 177-3027	2020-05-01
2	Bridley Roche	Phone	(979) 279-2592	2020-05-01
7	Bertha Areola	Phone	(840) 927-1743	2020-05-15
6	Eino Burns	SMS	(521) 202-4435	2020-04-15

Query 8

- Find the id of individuals that were not infected and have been re-exposed.
- Include the date the original case was closed, and the new exposure date.

The screenshot shows the MySQL Workbench interface with the following details:

- Query Editor:** Contains the SQL query for Query 8, which joins cases and contactlogs to find individuals who were not infected but have a newer timestamp than their original closure date.
- Result Grid:** Displays the results of the query, showing columns: tracked_person_id, originalClosureDate, and exposureDate. The results show two individuals with their respective dates.
- Action Output:** Shows the execution log with 15 rows returned for each of the five SELECT statements in the query.

```

1 • SELECT tracked_person_id, closureDate AS originalClosureDate, timestamp AS exposureDate
2     FROM cases JOIN contactlogs ON tracked_person_id = contact_person_id
3     WHERE infected = 0 AND timestamp > closureDate

```

tracked_person_id	originalClosureDate	exposureDate
81	2020-07-11	2020-08-25 06:15:53
54	2020-05-07	2020-08-24 14:40:18