Practicum

Kanishka Parganiha

10/13/2020

Part 3:

UML Model

https://lucid.app/invitations/accept/68c20656-cdd1-4e71-a89a-7375b5ff9604
(https://lucid.app/invitations/accept/68c20656-cdd1-4e71-a89a-7375b5ff9604)

ASSUMPTIONS :

The following are the various assumptions made for each table in the database: • testingcandidate: A person who visited testing center is termed as testing candidate. He is given CandidateId key as primary key for his identification

• covidtest: Each testedcandidate is given OrderId which uniquely identifies his test result of Covid Test and this key acts as primary key for the covidtest tables.

• Testingcenter: The person where he/she gets his covid test done is the testing center. The CenterId uniquely identifies the center.

• Zipcode: Table which contain the location of the State, City and Street Name

• Selfreported: This table contains the list of individuals who have not taken Covid test but shows the symptoms of covid. They self report themselves to the Authorities.

• Taskforce: Regulatory Body which does the task of contact tracing by assigning unique ReportingId number to every individuals who has done Covid Test or is suspected or has Travel History.

• Active: table which contains the list of individuals who are Covid-Positive

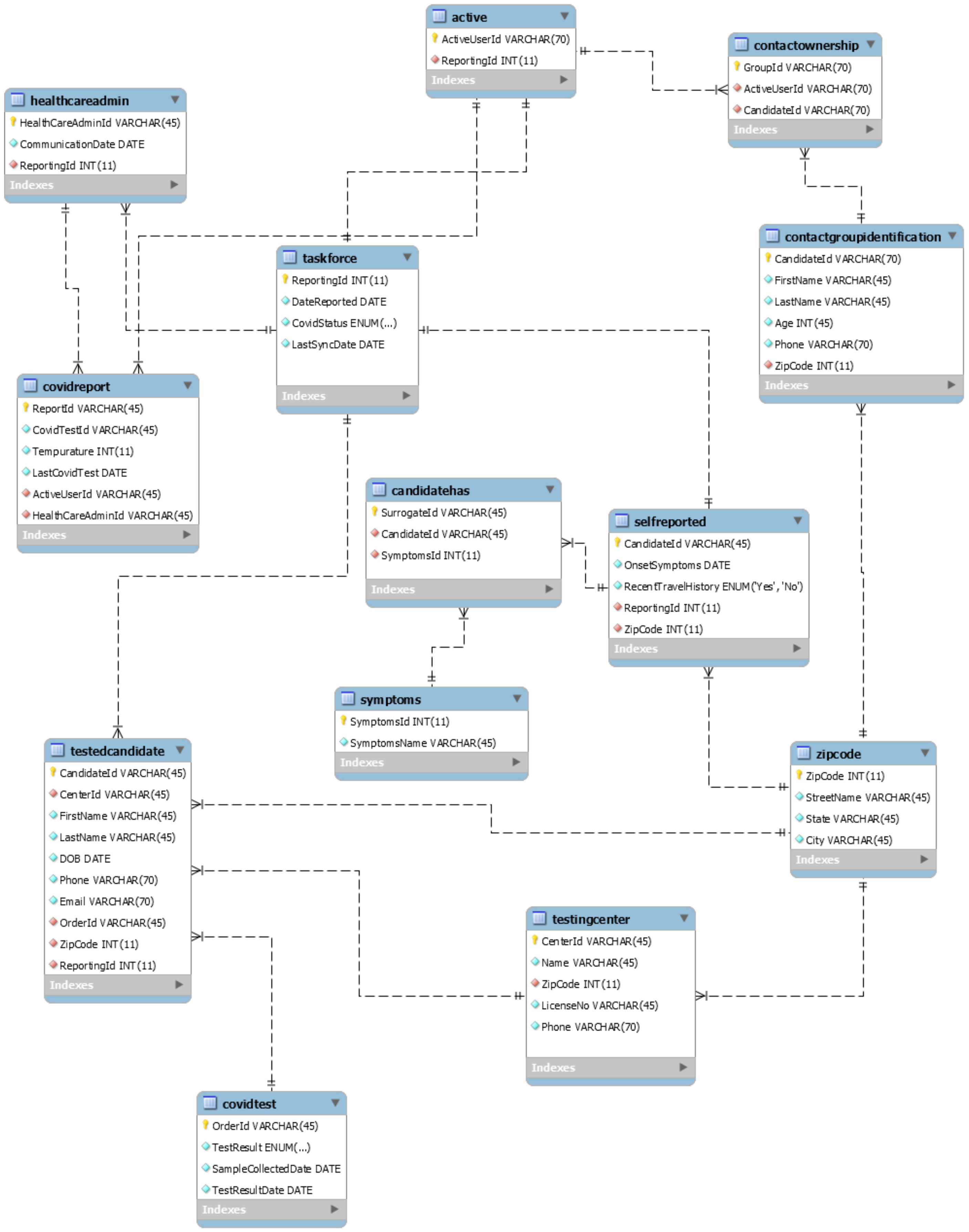
• Contactownership: Table created to map n:m multiplicity between contactgroupidentification and active table

• Contactgroupidentification: List of people who recently came in contact with Covid tested person

• Healthcareadmin: Health care worker who work along with the task force to monitor the infected patient and verifies the individual reports from them

Part 4

Logical Model



ERD Diagram Design in MySql WorkBench

• Key Icon: Primary Key • Light Red: Foreign Key

Relational Schema

```
library(DBI)
```

```
## Warning: package 'DBI' was built under R version 3.6.3
```

```
library(RMySQL)
```

```
## Warning: package 'RMySQL' was built under R version 3.6.3
```

```
mydrv <- dbDriver("MySQL")
conn <- dbConnect(mydrv, dbname="contacttracing",host="127.0.0.1",port=3306, user="root",passwor
d="Tenda@220")
```

- testingcenter

```
dbFetch(dbSendQuery(conn, 'desc testingcenter;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
CenterId	varchar(45)	NO	PRI	*NA*	
Name	varchar(45)	NO		*NA*	
ZipCode	int(11)	NO	MUL	*NA*	
LicenseNo	varchar(45)	NO		*NA*	
Phone	varchar(70)	NO		*NA*	

5 rows

- covidtest

```
dbFetch(dbSendQuery(conn, 'desc covidtest;'))
```

Field <chr>	Type <chr>	Null <chr>	... <chr>	Default <chr>	Extra <chr>
OrderId	varchar(45)	NO	PRI	*NA*	
TestResult	enum('Positive','Negative')	NO		*NA*	
SampleCollectedDate	date	NO		*NA*	
TestResultDate	date	NO		*NA*	

4 rows

- testedcandidate

```
dbFetch(dbSendQuery(conn, 'desc testedcandidate;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
CandidateId	varchar(45)	NO	PRI	NA	
CenterId	varchar(45)	NO	MUL	NA	
FirstName	varchar(45)	NO		NA	
LastName	varchar(45)	NO		NA	
DOB	date	NO		NA	
Phone	varchar(70)	NO		NA	
Email	varchar(70)	NO		NA	
OrderId	varchar(45)	NO	MUL	NA	
ZipCode	int(11)	NO	MUL	NA	
ReportingId	int(11)	NO	MUL	NA	

1-10 of 10 rows

- zipcode

```
dbFetch(dbSendQuery(conn, 'desc zipcode;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
ZipCode	int(11)	NO	PRI	NA	
StreetName	varchar(45)	NO		NA	
State	varchar(45)	NO		NA	
City	varchar(45)	NO		NA	

4 rows

- selfreported

```
dbFetch(dbSendQuery(conn, 'desc selfreported;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
CandidateId	varchar(45)	NO	PRI	NA	
OnsetSymptoms	date	NO		NA	
RecentTravelHistory	enum('Yes','No')	NO		NA	

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
ReportingId	int(11)	NO	MUL	NA	
ZipCode	int(11)	NO	MUL	NA	

5 rows

- symptoms

```
dbFetch(dbSendQuery(conn, 'desc symptoms;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
SymptomsId	int(11)	NO	PRI	NA	
SymptomsName	varchar(45)	NO		NA	

2 rows

- taskforce

```
dbFetch(dbSendQuery(conn, 'desc taskforce;'))
```

Field <chr>	Type <chr>	Null <chr>	… <chr>	Default <chr>	Extra <chr>
ReportingId	int(11)	NO	PRI	NA	
DateReported	date	NO		NA	
CovidStatus	enum('Positive','Suspected','Negative')	NO		NA	
LastSyncDate	date	NO		NA	

4 rows

- active

```
dbFetch(dbSendQuery(conn, 'desc active;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
ActiveUserId	varchar(70)	NO	PRI	NA	
ReportingId	int(11)	NO	MUL	NA	

2 rows

- healthcareadmin

```
dbFetch(dbSendQuery(conn, 'desc healthcareadmin;'))
```

Field	Type	Null	Key	Default	Extra
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
HealthCareAdminId	varchar(45)	NO	PRI	NA	
CommunicationDate	date	NO		NA	
ReportingId	int(11)	NO	MUL	NA	

3 rows

- covidreport

```
dbFetch(dbSendQuery(conn, 'desc covidreport;'))
```

Field	Type	Null	Key	Default	Extra
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
ReportId	varchar(45)	NO	PRI	NA	
CovidTestId	varchar(45)	NO		NA	
Tempurature	int(11)	NO		NA	
LastCovidTest	date	NO		NA	
ActiveUserId	varchar(45)	NO	MUL	NA	
HealthCareAdminId	varchar(45)	NO	MUL	NA	

6 rows

- contactownership

```
dbFetch(dbSendQuery(conn, 'desc contactownership;'))
```

Field	Type	Null	Key	Default	Extra
<chr>	<chr>	<chr>	<chr>	<chr>	<chr>
GroupId	varchar(70)	NO	PRI	NA	
ActiveUserId	varchar(70)	NO	MUL	NA	
CandidateId	varchar(70)	NO	MUL	NA	

3 rows

- contactgroupidentification

```
dbFetch(dbSendQuery(conn, 'desc contactgroupidentification;'))
```

Field <chr>	Type <chr>	Null <chr>	Key <chr>	Default <chr>	Extra <chr>
CandidateId	varchar(70)	NO	PRI	NA	
FirstName	varchar(45)	NO		NA	
LastName	varchar(45)	NO		NA	
Age	int(45)	NO		NA	
Phone	varchar(70)	NO		NA	
ZipCode	int(11)	NO	MUL	NA	

6 rows

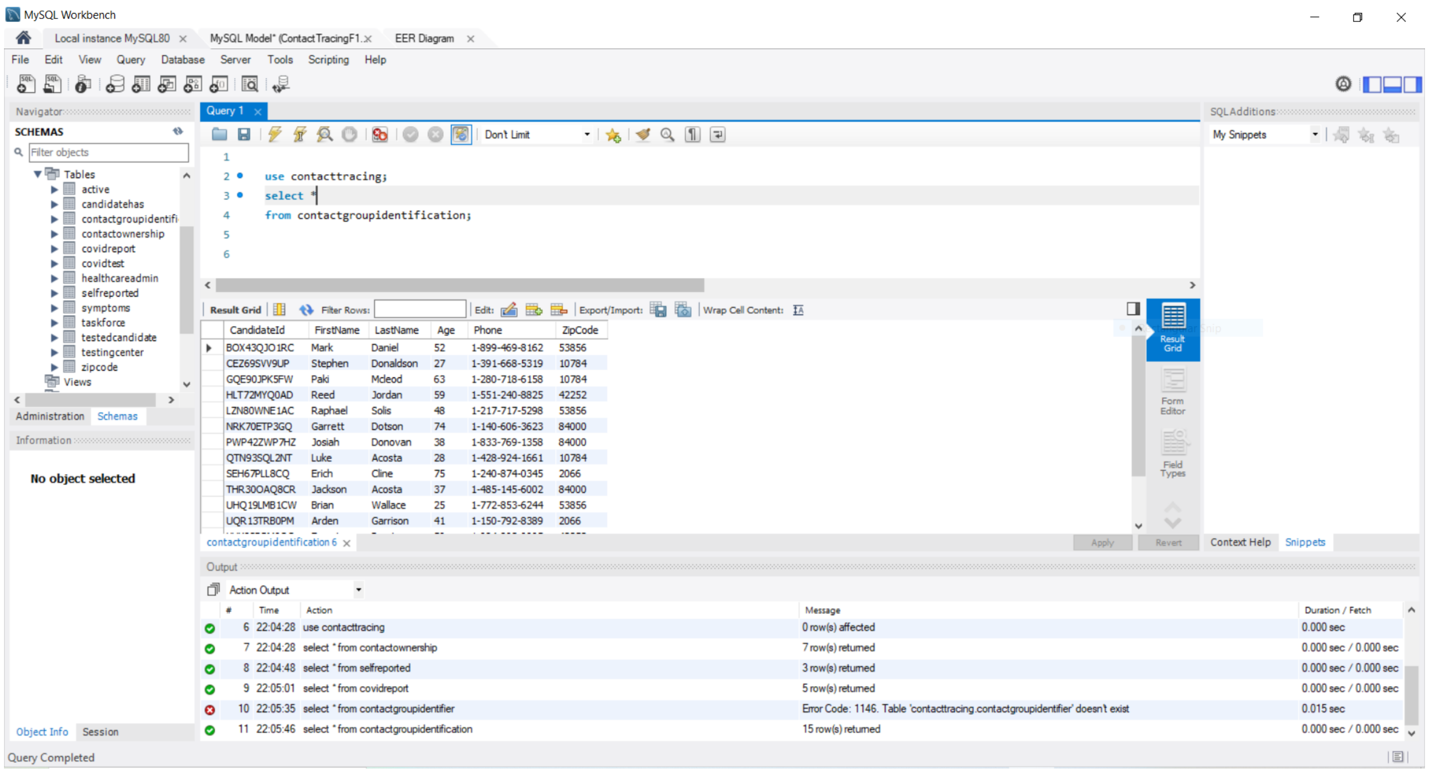
NORMALIZATION

1. testedcandidate The Functional Dependencies are, candidateId=> FirstName candidateId => Lastname candidateId => Phone candidateId => Email candidateId => DOB candidateId => OrderId candidateId => ReportingId Thus candidateId is the only candidate keys. This relation is in BCNF as the candidate key is the only determinants of every FD. There is no partial dependency as well. There are no transitive dependencies. And there are no multivalued attributes.

2. zipcode The Functional Dependencies are, Zipcode=> Streetname Zipcode => State Zipcode => City Thus Zipcode is the only candidate keys. This relation is in BCNF as the candidate key is the only determinants of every FD. There is no partial dependency as well. There are no transitive dependencies. And there are no multivalued attributes.

3. Normalized N:M association multiplicity between active and contactgroupidentification to 3NF form with GroupId key as surrogate Key

Part 5

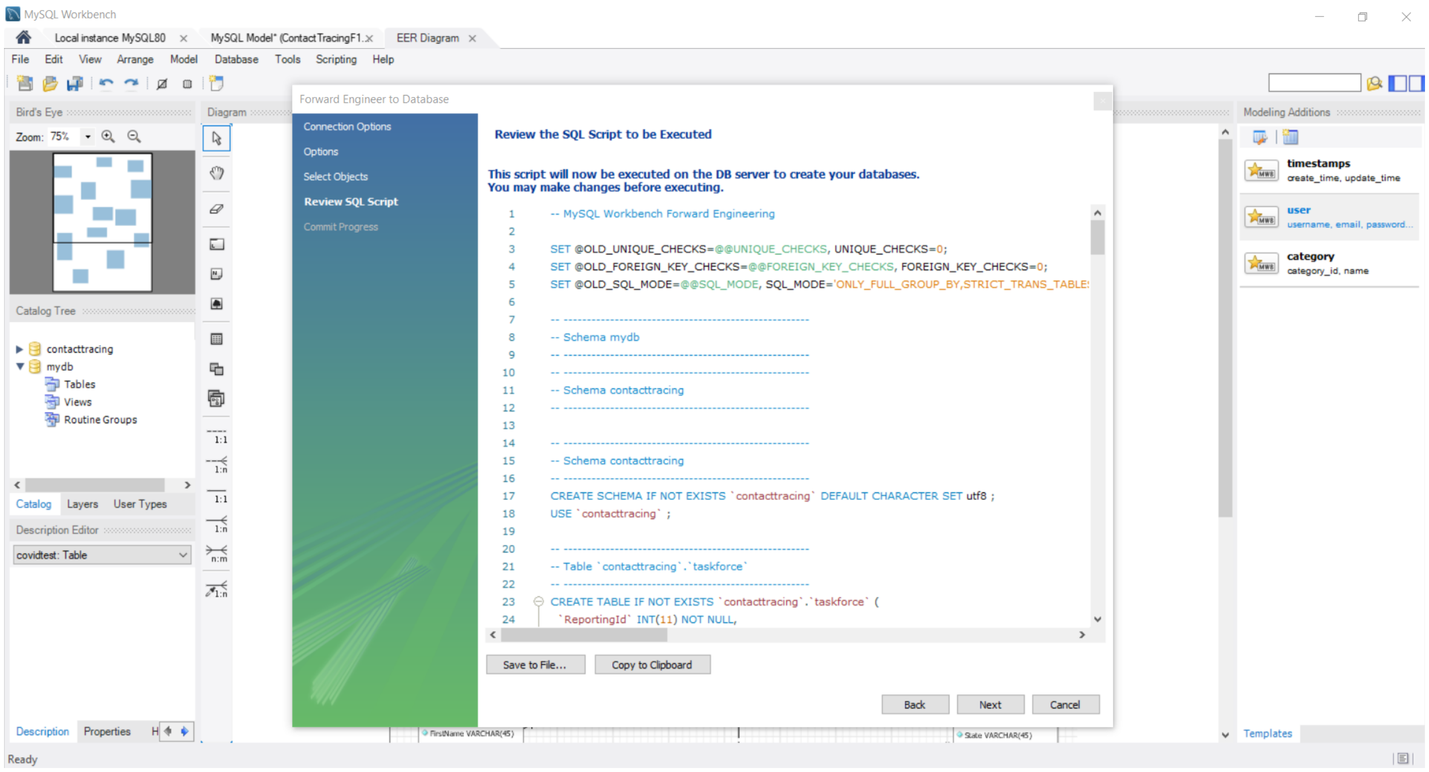


```
use contacttracing;
select *
from contactgroupidentification;
```

	CandidateId	FirstName	LastName	Age	Phone	ZipCode
▶	BOX43QJO1RC	Mark	Daniel	52	1-899-469-8162	53856
	CEZ69SVV9UP	Stephen	Donaldson	27	1-391-668-5319	10784
	GQE90JPK5FW	Paki	Mcleod	63	1-280-718-6158	10784
	HLT72MYQ0AD	Reed	Jordan	59	1-551-240-8825	42252
	LZN80WNE1AC	Raphael	Solis	48	1-217-717-5298	53856
	NRK70ETP3GQ	Garrett	Dotson	74	1-140-606-3623	84000
	PWP42ZWP7HZ	Josiah	Donovan	38	1-833-769-1358	84000
	QTN93SQL2NT	Luke	Acosta	28	1-428-924-1661	10784
	SEH67PLL8CQ	Erich	Cline	75	1-240-874-0345	2066
	THR30OAQ8CR	Jackson	Acosta	37	1-485-145-6002	84000
	UHQ19LMB1CW	Brian	Wallace	25	1-772-853-6244	53856
	UQR13TRB0PM	Arden	Garrison	41	1-150-792-8389	2066

	#	Time	Action	Message	Duration / Fetch
✓	6	22:04:28	use contacttracing	0 row(s) affected	0.000 sec
✓	7	22:04:28	select * from contactownership	7 row(s) returned	0.000 sec / 0.000 sec
✓	8	22:04:48	select * from selfreported	3 row(s) returned	0.000 sec / 0.000 sec
✓	9	22:05:01	select * from covidreport	5 row(s) returned	0.000 sec / 0.000 sec
✗	10	22:05:35	select * from contactgroupidentifier	Error Code: 1146. Table 'contacttracing.contactgroupidentifier' doesn't exist	0.015 sec
✓	11	22:05:46	select * from contactgroupidentification	15 row(s) returned	0.000 sec / 0.000 sec

Querying in MySql WorkBench



Review the SQL Script to be Executed

This script will now be executed on the DB server to create your databases.
You may make changes before executing.

```
-- MySQL Workbench Forward Engineering

SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='ONLY_FULL_GROUP_BY,STRICT_TRANS_TABLE

-- ------------------------------------------------------
-- Schema mydb
-- ------------------------------------------------------
-- ------------------------------------------------------
-- Schema contacttracing
-- ------------------------------------------------------

-- ------------------------------------------------------
-- Schema contacttracing
-- ------------------------------------------------------
CREATE SCHEMA IF NOT EXISTS `contacttracing` DEFAULT CHARACTER SET utf8 ;
USE `contacttracing` ;

-- ------------------------------------------------------
-- Table `contacttracing`.`taskforce`
-- ------------------------------------------------------
CREATE TABLE IF NOT EXISTS `contacttracing`.`taskforce` (
  `ReportingId` INT(11) NOT NULL,
```

Forward Engineering in MySql WorkBench

– MySQL Workbench Forward Engineering

```
CREATE SCHEMA IF NOT EXISTS `contacttracing` DEFAULT CHARACTER SET utf8 ;
USE `contacttracing` ;
```

– Table `contacttracing . taskforce`

CREATE TABLE IF NOT EXISTS `contacttracing . taskforce` (`ReportingId` INT(11) NOT NULL, `DateReported` DATE NOT NULL, `CovidStatus` ENUM('Positive', 'Suspected', 'Negative') NOT NULL, `LastSyncDate` DATE NOT NULL, PRIMARY KEY (`ReportingId`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;

– Table `contacttracing . active`

CREATE TABLE IF NOT EXISTS `contacttracing . active` (

`ActiveUserId` VARCHAR(70) NOT NULL,

`ReportingId` INT(11) NOT NULL,

PRIMARY KEY (`ActiveUserId`),

INDEX `fk_Active_TaskForce1_idx` (`ReportingId` ASC) VISIBLE,

CONSTRAINT `fk_Active_TaskForce1`

FOREIGN KEY (`ReportingId`)

REFERENCES `contacttracing . taskforce` (`ReportingId`))

– Table `contacttracing . active`

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

– Table `contacttracing . zipcode`

CREATE TABLE IF NOT EXISTS `contacttracing . zipcode` (`ZipCode` INT(11) NOT NULL, `StreetName` VARCHAR(45) NOT NULL, `State` VARCHAR(45) NOT NULL, `City` VARCHAR(45) NOT NULL, PRIMARY KEY (`ZipCode`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;

– Table `contacttracing . selfreported`

CREATE TABLE IF NOT EXISTS `contacttracing . selfreported` (

`CandidateId` VARCHAR(45) NOT NULL,

`OnsetSymptoms` DATE NOT NULL,

`RecentTravelHistory` ENUM('Yes', 'No') NOT NULL,

`ReportingId` INT(11) NOT NULL,

`ZipCode` INT(11) NOT NULL,

PRIMARY KEY (`CandidateId`),

INDEX `fk_SelfReported_Task Force1_idx` (`ReportingId` ASC) VISIBLE,

INDEX `fk_SelfReported_ZipCode1_idx` (`ZipCode` ASC) VISIBLE,

CONSTRAINT `fk_SelfReported_Task Force1`

FOREIGN KEY (`ReportingId`)

REFERENCES `contacttracing . taskforce` (`ReportingId`),

CONSTRAINT `fk_SelfReported_ZipCode1`

FOREIGN KEY (`ZipCode`)

REFERENCES `contacttracing . zipcode` (`ZipCode`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

– Table `contacttracing . symptoms`

CREATE TABLE IF NOT EXISTS `contacttracing . symptoms` (`SymptomsId` INT(11) NOT NULL, `SymptomsName` VARCHAR(45) NOT NULL, PRIMARY KEY (`SymptomsId`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;

– Table contacttracing . candidatehas

```
CREATE TABLE IF NOT EXISTS contacttracing . candidatehas (
 SurrogateId VARCHAR(45) NOT NULL,
 CandidateId VARCHAR(45) NOT NULL,
 SymptomsId INT(11) NOT NULL,
PRIMARY KEY ( SurrogateId ),
INDEX fk_candidatehas_selfreported1_idx ( CandidateId ASC) VISIBLE,
INDEX fk_candidatehas_symptoms_copy1_copy11 ( SymptomsId ASC) VISIBLE,
CONSTRAINT fk_candidatehas_selfreported1
FOREIGN KEY ( CandidateId )
REFERENCES contacttracing . selfreported ( CandidateId ),
CONSTRAINT fk_candidatehas_symptoms_copy1_copy11
FOREIGN KEY ( SymptomsId )
REFERENCES contacttracing . symptoms ( SymptomsId ))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

– Table contacttracing . contactgroupidentification

```
CREATE TABLE IF NOT EXISTS contacttracing . contactgroupidentification (
 CandidateId VARCHAR(70) NOT NULL, FirstName VARCHAR(45) NOT NULL,
 LastName VARCHAR(45) NOT NULL, Age INT(45) NOT NULL, Phone
VARCHAR(70) NOT NULL, ZipCode INT(11) NOT NULL, PRIMARY KEY
( CandidateId ), INDEX fk_contactgroupidentification_zipcode1_idx ( ZipCode
ASC) VISIBLE, CONSTRAINT fk_contactgroupidentification_zipcode1
FOREIGN KEY ( ZipCode ) REFERENCES contacttracing . zipcode ( ZipCode ))
ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;
```

– Table contacttracing . contactownership

```
CREATE TABLE IF NOT EXISTS contacttracing . contactownership (
GroupId VARCHAR(70) NOT NULL,
ActiveUserId VARCHAR(70) NOT NULL,
CandidateId VARCHAR(70) NOT NULL,
PRIMARY KEY ( GroupId ),
INDEX fk_ContactOwnership_ContactGroupIdentification1_idx ( CandidateId ASC) VISIBLE,
```

– Table contacttracing . contactownership

```
INDEX fk_contactownership_active1_idx ( ActiveUserId ASC) VISIBLE,
CONSTRAINT fk_ContactOwnership_ContactGroupIdentification1
FOREIGN KEY ( CandidateId )
REFERENCES contacttracing . contactgroupidentification ( CandidateId ),
CONSTRAINT fk_contactownership_active1
FOREIGN KEY ( ActiveUserId )
REFERENCES contacttracing . active ( ActiveUserId ))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

– Table contacttracing . healthcareadmin

```
CREATE TABLE IF NOT EXISTS contacttracing . healthcareadmin (
HealthCareAdminId VARCHAR(45) NOT NULL, CommunicationDate DATE NOT
NULL, ReportingId INT(11) NOT NULL, PRIMARY KEY ( HealthCareAdminId ),
INDEX fk_HealthCareAdmin_TaskForce1_idx ( ReportingId ASC) VISIBLE,
CONSTRAINT fk_HealthCareAdmin_TaskForce1 FOREIGN KEY ( ReportingId )
REFERENCES contacttracing . taskforce ( ReportingId )) ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

– Table contacttracing . covidreport

```
CREATE TABLE IF NOT EXISTS contacttracing . covidreport (
 ReportId VARCHAR(45) NOT NULL,
 CovidTestId VARCHAR(45) NOT NULL,
 Tempurature INT(11) NOT NULL,
 LastCovidTest DATE NOT NULL,
 ActiveUserId VARCHAR(45) NOT NULL,
 HealthCareAdminId VARCHAR(45) NOT NULL,
PRIMARY KEY ( ReportId ),
INDEX fk_CovidReport_Active1_idx ( ActiveUserId ASC) VISIBLE,
INDEX fk_CovidReport_HealthCareAdmin1_idx ( HealthCareAdminId ASC) VISIBLE,
CONSTRAINT fk_CovidReport_Active1
FOREIGN KEY ( ActiveUserId )
REFERENCES contacttracing . active ( ActiveUserId ),
```

– Table `contacttracing . covidreport`

CONSTRAINT `fk_CovidReport_HealthCareAdmin1`

FOREIGN KEY (`HealthCareAdminId`)

REFERENCES `contacttracing . healthcareadmin` (`HealthCareAdminId`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

– Table `contacttracing . covidtest`

CREATE TABLE IF NOT EXISTS `contacttracing . covidtest` (`OrderId` VARCHAR(45) NOT NULL, `TestResult` ENUM('Positive', 'Negative') NOT NULL, `SampleCollectedDate` DATE NOT NULL, `TestResultDate` DATE NOT NULL, PRIMARY KEY (`OrderId`)) ENGINE = InnoDB DEFAULT CHARACTER SET = utf8;

– Table `contacttracing . testingcenter`

CREATE TABLE IF NOT EXISTS `contacttracing . testingcenter` (

`CenterId` VARCHAR(45) NOT NULL,

`Name` VARCHAR(45) NOT NULL,

`ZipCode` INT(11) NOT NULL,

`LicenseNo` VARCHAR(45) NOT NULL,

`Phone` VARCHAR(70) NOT NULL,

PRIMARY KEY (`CenterId`),

INDEX `fk_TestingCenter_ZipCode1_idx` (`ZipCode` ASC) VISIBLE,

CONSTRAINT `fk_TestingCenter_ZipCode10`

FOREIGN KEY (`ZipCode`)

REFERENCES `contacttracing . zipcode` (`ZipCode`))

ENGINE = InnoDB

DEFAULT CHARACTER SET = utf8;

–Table `contacttracing . testedcandidate`

```
CREATE TABLE IF NOT EXISTS `contacttracing`.`testedcandidate` (
  `CandidateId` VARCHAR(45) NOT NULL,
  `CenterId` VARCHAR(45) NOT NULL,
  `FirstName` VARCHAR(45) NOT NULL,
  `LastName` VARCHAR(45) NOT NULL,
  `DOB` DATE NOT NULL,
  `Phone` VARCHAR(70) NOT NULL,
  `Email` VARCHAR(70) NOT NULL,
  `OrderId` VARCHAR(45) NOT NULL,
  `ZipCode` INT(11) NOT NULL,
  `ReportingId` INT(11) NOT NULL,
  PRIMARY KEY (`CandidateId`),
  INDEX `OrderId_idx` (`OrderId` ASC) VISIBLE,
  INDEX `fk_TestedCandidate_Task Force1_idx` (`ReportingId` ASC) VISIBLE,
  INDEX `fk_TestedCandidate_ZipCode1_idx` (`ZipCode` ASC) VISIBLE,
  INDEX `fk_testedcandidate_testingcenter1_idx` (`CenterId` ASC) VISIBLE,
  CONSTRAINT `OrderId0`
    FOREIGN KEY (`OrderId`)
    REFERENCES `contacttracing`.`covidtest` (`OrderId`),
  CONSTRAINT `fk_TestedCandidate_Task Force1`
    FOREIGN KEY (`ReportingId`)
    REFERENCES `contacttracing`.`taskforce` (`ReportingId`),
  CONSTRAINT `fk_TestedCandidate_ZipCode1`
    FOREIGN KEY (`ZipCode`)
    REFERENCES `contacttracing`.`zipcode` (`ZipCode`),
  CONSTRAINT `fk_testedcandidate_testingcenter1`
    FOREIGN KEY (`CenterId`)
    REFERENCES `contacttracing`.`testingcenter` (`CenterId`))
ENGINE = InnoDB
DEFAULT CHARACTER SET = utf8;
```

SET SQL_MODE=@OLD_SQL_MODE (mailto:SQL_MODE=@OLD_SQL_MODE); SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS (mailto:FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS); SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS (mailto:UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS);

Part 6 Insertion Queries

INSERT INTO zipcode (ZipCode,StreetName,State,City) VALUES (“10784”,“Opal Village”, “California”,“Los Angeles”), (“50835”,“Cartwright Meadow”,“Minnesota”,“Saint Paul”), (“19379”,“Pagac Ramp”,“Tennessee”,“Nashville”), (“42252”,“Rogers Extension”,“Georgia”,“Atlanta”), (“84000”,“Marielle Coves”,“California”,“Los Angeles”), (“06774”,“Crooks Roads”,“California”,“Los Angeles”), (“12422”,“Anibal Stream”,“New York”,“Buffalo”), (“53856”,“Daphne Plains”,“New York”,“New York City”), (“10906”,“Wyman Inlet”,“Missouri”,“Kansas City”), (“02066”,“Camron Flat”,“Missouri”,“Kansas City”)

INSERT INTO testingcenter (CenterId,Name,ZipCode,LicenseNo,Phone) VALUES (“GGIY930”,“Gravida”,“50835”,“CCQ63YSI9YM”,“1-257-419-2125”), (“CEBY227”,“Art Avenue”,“10906”,“IIE27FLI1SU”,“1-694-581-9531”), (“TXEY205”,“Dolor”,“50835”,“IIZ44UHK7MZ”,“1-572-524-4960”), (“EYXM942”,“Elite”,“12422”,“YXM70XGK6FD”,“1-613-416-2970”) ,(“ZXME144”,“996 Hawk”,“53856”,“TSZ13NHE3OV”,“1-455-963-8898”) ,(“ANWV749”,“Auctor”,“06774”,“FIU46JOV5DK”,“1-561-744-6198”) ,(“COOK457”,“Porttitor”,“19379”,“CPC43TNY2BP”,“1-517-122-5863”);

```
INSERT INTO taskforce (ReportingId,DateReported,CovidStatus,LateSyncDate) VALUES (“251697”,“2020-08-
01”,“Positive”,“2020-09-07”), (“978362”,“2020-08-07”,“Negative”,“2020-09-16”) ,(“880509”,“2020-08-
14”,“Negative”,“2020-09-24”) ,(“989349”,“2020-08-04”," Positive “,”2020-09-12 “), ("985882“,”2020-08-05“,” Positive
“,”2020-10-01 “) ,(”597538“,”2020-08-25“,”Negative“,”2020-09-20 “) ,(”381416“,”2020-08-19“,”Negative“,”2020-10-
01 ");
```

```
INSERT INTO covidtest (OrderId,TestResult,SampleCollectedDate,TestResultDate) VALUES
(“920317”,“Positive”,“2020-09-05”,“2020-09-18”) ,(“795954”,“Negative”,“2020-09-13”,“2020-09-22”) ,
(“143503”,“Negative”,“2020-09-01”,“2020-09-27”), (“153649”,“Negative”,“2020-09-03”,“2020-09-24”),
(“342421”,“Positive”,“2020-09-04”,“2020-10-01”) ,(“280492”,“Positive”,“2020-09-06”,“2020-09-18”) ,
(“871028”,“Negative”,“2020-09-10”,“2020-09-17”);
```

```
INSERT INTO testedcandidate
(CandidateId,CenterId,FirstName,LastName,DOB,Phone,Email,OrderId,ZipCode,ReportingId) VALUES
(“SJG20231”,“GGIY930”,“Nissim”,“Dyer”,“1969-12-31”,“1-736-566-4425”,“eget.ipsum.Donec@mollis.org
(mailto:eget.ipsum.Donec@mollis.org)”,“920317”,“50835”,“251697”),
(“RQV30513”,“GGIY930”,“Tanek”,“Jackson”,“1969-12-31”,“1-379-748-3323”,“consequat@diamDuismi.ca
(mailto:consequat@diamDuismi.ca)”,“153649”,“50835”,“978362”) ,
(“UZK77843”,“COOK457”,“Howard”,“Murray”,“1969-12-31”,“1-469-932-4425”,“Donec.porttitor.tellus@risus.co.uk
(mailto:Donec.porttitor.tellus@risus.co.uk)”,“143503”,“12422”,“880509”),
(“QKN80556”,“ANWV749”,“Kennedy”,“Burks”,“1969-12-31”,“1-423-727-1379”,“nulla.Integer@non.com
(mailto:nulla.Integer@non.com)”,“280492”,“06774”,“989349”) ,(“EWK36743”,“ANWV749”,“Armando”,“Byrd”,“1969-
12-31”,“1-433-304-0459”,“Mauris.magna.Duis@accumsan.ca
(mailto:Mauris.magna.Duis@accumsan.ca)”,“342421”,“10906”,“985882”) ,
(“HKV04560”,“COOK457”,“Zahir”,“Carrillo”,“1969-12-31”,“1-242-445-5926”,“at@Nullamsuscipitest.edu
(mailto:at@Nullamsuscipitest.edu)”,“871028”,“10906”,“597538”);
```

```
INSERT INTO active(ActiveUserId,ReportingId) VALUES (“YLHR6228”, “251697”) ,(“VFDW5805”, “985882”) ,
(“GSPA4057”, “880509”) ,(“DHFO0615”, “989349”) ,(“AZUV4790”, “597538”) ;
```

```
INSERT INTO contactownership(GroupId,ActiveUserId,CandidateId) VALUES
(“YPRF8254”,“YLHR6228”,“GQE90JPK5FW”) ,(“QIXA9762”,“YLHR6228”,“CEZ69SVV9UP”),
(“ZAUX3675”,“AZUV4790”,“QTN93SQL2NT”) ,(“YRFX0967”,“GSPA4057”,“BOX43QJO1RC”) ,
(“FVDY0232”,“GSPA4057”,“UHQ19LMB1CW”) ,(“UIQK1684”,“DHFO0615”,“HLT72MYQ0AD”) ,
(“EALB5226”,“DHFO0615”,“SEH67PLL8CQ”);
```

```
INSERT INTO contactgroupidentifation (CandidateId,FirstName,LastName,Age,Phone,ZipCode) VALUES
(“GQE90JPK5FW”,“Paki”,“Mcleod”,63,“1-280-718-6158”,“10784”),
(“CEZ69SVV9UP”,“Stephen”,“Donaldson”,27,“1-391-668-5319”,“10784”), (“QTN93SQL2NT”,“Luke”,“Acosta”,28,“1-
428-924-1661”,“10784”) ,(“BOX43QJO1RC”,“Mark”,“Daniel”,52,“1-899-469-8162”,“53856”),
(“LZN80WNE1AC”,“Raphael”,“Solis”,48,“1-217-717-5298”,“53856”), (“UHQ19LMB1CW”,“Brian”,“Wallace”,25,“1-
772-853-6244”,“9140 BQ”), (“XNF54NCA9ND”,“Hunter”,“English”,67,“1-535-613-8886”,“84000”) ,
(“NRK70ETP3GQ”,“Garrett”,“Dotson”,74,“1-140-606-3623”,“84000”) ,(“PWP42ZWP7HZ”,“Josiah”,“Donovan”,38,“1-
833-769-1358”,“84000”) ,(“HLT72MYQ0AD”,“Reed”,“Jordan”,59,“1-551-240-8825”,“42252”),
(“VUY82RPM9OQ”,“Francis”,“Preston”,59,“1-984-893-9905”,“42252”) ,(“SEH67PLL8CQ”,“Erich”,“Cline”,75,“1-240-
874-0345”,“02066”), (“WWL05CBK8KK”,“Brady”,“Roach”,66,“1-199-822-7523”,“02066”) ,
(“UQR13TRB0PM”,“Arden”,“Garrison”,41,“1-150-792-8389”,“02066”),
(“THR30OAQ8CR”,“Jackson”,“Acosta”,37,“1-485-145-6002”,“84000”);
```

```
INSERT INTO healthcareadmin (HealthCareAdminId,CommunicationDate,ReportingId) VALUES
(“SL833UT3GQ”,“2020-10-03”,“251697”) ,(“SW436HG6OL”,“2020-09-27”,“989349”) ,(“VE799NF5RI”,“2020-10-
06”,“985882”) ,(“QH791SY9TR”,“2020-10-06”,“381416”) ,(“GH647AI9BB”,“2020-10-11”,“597538”);
```

INSERT INTO covidreport (ReportId,CovidTestId,Tempurature,LastCovidTest,ActiveUserId,HealthCareAdminId) VALUES (“KF405IX4MJ”,“SPM54UIR2GS”,116,“2020-10-28”," AZUV4790“,” SL833UT3GQ“), (”EI508SI3RW“,”XJI99EVA8IY“,91,”2020-10-07“,” DHFO0615“,” SW436HG6OL“) , (”HO155LT3TG“,”UZQ14MPQ9CD“,116,”2020-10-23“,” GSPA4057“,” SW436HG6OL “) , (”OC706CP9PJ“,”BBJ31TZM7TD“,94,”2020-10-01“,” VFDW5805“,” SW436HG6OL“) , (”KL540BY8LF“,”MRL95LMA0SY“,116,”2020-10-19“,” YLHR6228“,” GH647AI9BB ");

INSERT INTO selfreported (CandidateId,OnsetSymptomsDate,RecentTravelHistory,ReportingId,ZipCode) VALUES (“HS018AS3EE”,“2020-09-09”,“Yes”,“597538”,“10784”), (“AN574IW6EQ”,“2020-09-26”,“Yes”,“985882”,“10784”) ,(“EF588GK2NQ”,“2020-09-30”,“No”,“880509”,“42252”); INSERT INTO candidatehas (SurrogateId,CandidateId,SymptomsId) VALUES (“TB540YJ0ZC”,“HS018AS3EE”,1) , (“HU221RQ1QY”,“AN574IW6EQ”,6), (“VE308AH6ES”,“EF588GK2NQ”,5);

INSERT INTO candidatehas (SymptomsId,SymptomsName) VALUES (2,“Fever”), (5,“Sore throat”), (4,“Cough”) , (3,“Shortness of breath”) ,(1,“Nausea”);

Part 7.

```
library(DBI)
library(RMySQL)

mydrv <- dbDriver("MySQL")
conn <- dbConnect(mydrv, dbname="contacttracing",host="127.0.0.1",port=3306, user="root",passwor
d="Tenda@220")
```

QUERIES

1. Query to display the Name, Covid Test Result, the location of their Testing center along with the Date the Task Force Authority has their data entry by using multiple join statements.

```
dbFetch(dbSendQuery(conn, 'select tc.FirstName,tc.LastName,ct.TestResult,zc.State,tf.DateReporte
d from testedcandidate tc inner join covidtest ct on tc.OrderId=ct.OrderId inner join testingcen
ter ts on ts.CenterId=tc.CenterId inner join zipcode zc on zc.zipCode=ts.zipCode inner join task
force tf on tf.ReportingId=tc.ReportingId ;'))
```

FirstName <chr>	LastName <chr>	TestResult <chr>	State <chr>	DateReported <chr>
Armando	Byrd	Positive	California	2020-08-05
Zahir	Carrillo	Negative	Tennessee	2020-08-25
Kennedy	Burks	Positive	California	2020-08-04
Tanek	Jackson	Negative	Minnesota	2020-08-07
Nissim	Dyer	Positive	Minnesota	2020-08-01
Howard	Murray	Negative	Tennessee	2020-08-14

6 rows

2. Query to display the Name and contact Details of the potential Covid infected persons who has contacted the Person whose covid Status was ‘Positive’ in the database of Task Force.

```
dbFetch(dbSendQuery(conn, 'select FirstName,LastName,Age,Phone,ZipCode from contactgroupidentifi
cation where CandidateId  in (select co.CandidateId from contactownership co inner join active a
c on ac.ActiveUserId=co.ActiveUserId inner join taskforce tf on tf.ReportingId=ac.ReportingId wh
ere tf.CovidStatus="Positive");'))
```

FirstName	LastName	Age	Phone	ZipCode
<chr>	<chr>	<int>	<chr>	<int>
Stephen	Donaldson	27	1-391-668-5319	10784
Paki	Mcleod	63	1-280-718-6158	10784
Erich	Cline	75	1-240-874-0345	2066
Reed	Jordan	59	1-551-240-8825	42252

4 rows

3. Query to display the Streetwise count of Covid testing.

```
dbFetch(dbSendQuery(conn, 'select StreetName,count(*) as TestingCount from zipcode  group by Sta
te having count(*)>1;'))
```

StreetName	TestingCount
<chr>	<dbl>
Camron Flat	2
Crooks Roads	3
Anibal Stream	2

3 rows

4. Query to display the Id of the Patient who is Covid Negative but shows Body Tempurature higher than 90 from the database that is held with HealthCare Monitoring Body.

```
dbFetch(dbSendQuery(conn, 'select hc.ReportingId,cr.Tempurature,tf.CovidStatus from covidreport
 cr inner join healthcareadmin hc on cr.HealthCareAdminId=hc.HealthCareAdminId inner join taskfo
rce tf on tf.ReportingId=hc.ReportingId  where cr.Tempurature >90 and tf.CovidStatus = "Positiv
e" ;'))
```

ReportingId	Tempurature	CovidStatus
<int>	<int>	<chr>
989349	91	Positive
989349	116	Positive
251697	116	Positive
989349	94	Positive

4 rows

5. Query to display risk-wise status patient according to their age and CovidStatus of their recent Contact Person carrying Covid.

```
dbFetch(dbSendQuery(conn, 'select FirstName,LastName,Age,
                    CASE
                          WHEN Age > 50 THEN "Higher Risk"
                          WHEN Age < 50 THEN "Mild Risk"

                          END as Risk

                        from contactgroupidentification where CandidateId  in (select co.Candida
teId from contactownership co inner join active ac on ac.ActiveUserId=co.ActiveUserId inner join
taskforce tf on tf.ReportingId=ac.ReportingId where tf.CovidStatus="Positive");'))
```

FirstName <chr>	LastName <chr>	Age <int>	Risk <chr>
Stephen	Donaldson	27	Mild Risk
Paki	Mcleod	63	Higher Risk
Erich	Cline	75	Higher Risk
Reed	Jordan	59	Higher Risk

4 rows