DBMS Project Report

PES University

Database Management Systems

UE18CS252

Submitted By

|  |  |
| --- | --- |
| PES1201801896 | KRISHNA P HEGDE |

|  |
| --- |
| **CallCentre** is a database for managing the records that concerns with different employees, customers, problems, departments and call records etc..  This is basically used to store the records of different employees and the departments they belong to along with their other details like the address they belong to.  This database also stores the record of different customers along with their information of address, email\_id, name.  So, when a customer calls with some problem, a new call record is being created in the database under call table. And each call has its own call\_log with details regarding the type of problem and the duration of the call along with when the call was made (date and time).  Each problem has a unique solution\_id given by a specialist who is actually one among the employees.  This is a database SCHEMA for a single CallCentre, it can be replicated for many CallCentres with suitable modifications if required. |

**Index**

**[Introduction](#_gwqiqapyxr0x) 2**

**[Data Model](#_p451stoz4u81) 2**

**[FD and Normalization](#_wc0v2611q54c) 2**

**[DDL](#_pn441yprtk2f) 3**

**[Triggers](#_a9bswny9ffj) 3**

**[SQL Queries](#_exvna2br67rz) 3**

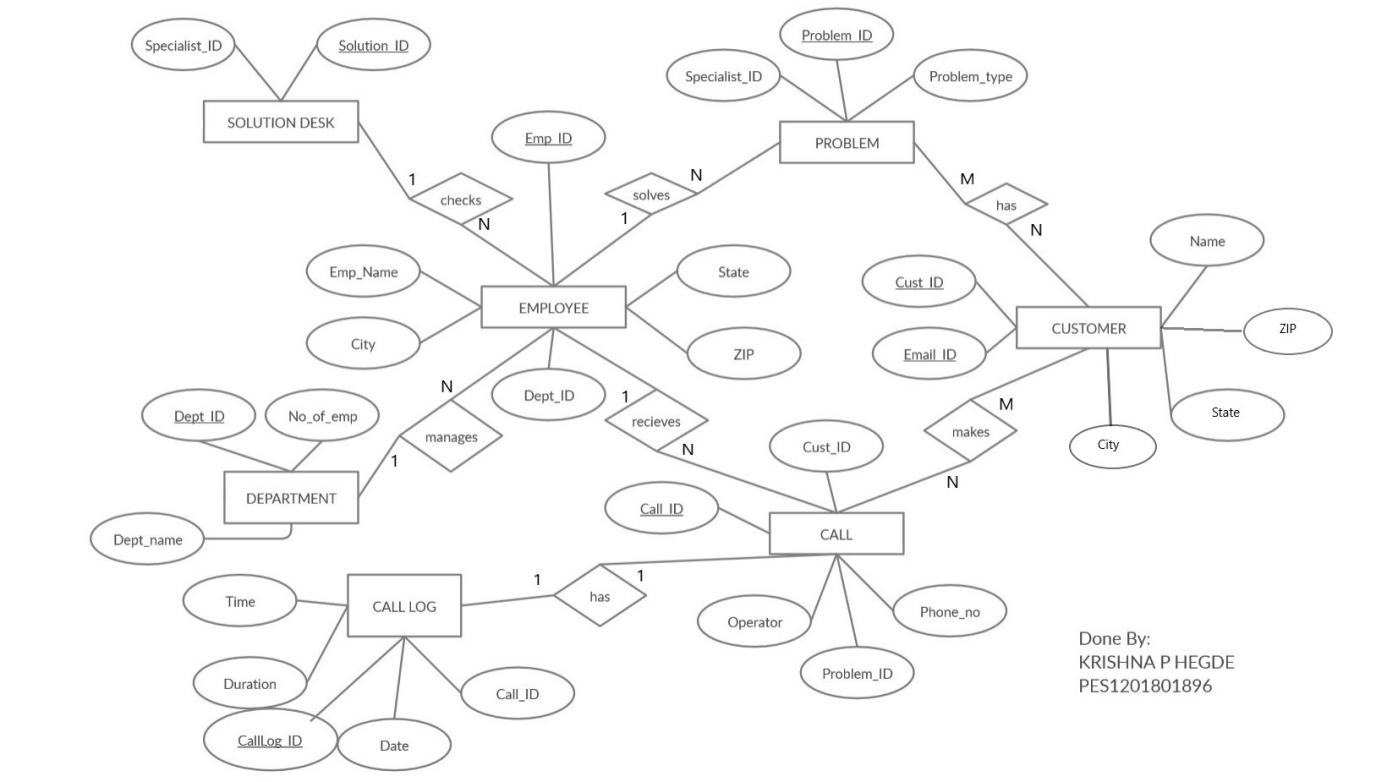
**[Conclusion](#_k6bbxckcf51j) 3**

# Introduction

CallCentre Database has 7 tables namely DEPARTMENT, EMPLOYEE, SOLUTION\_DESK, PROBLEM, CUSTOMER, CALL, CALL\_LOG. Each table has its own primary key and one/two foreign keys for maintaining relationship between tables. We can store the records of each employee’s details like emp\_id, name, address, the department he belongs to. And also it stores the records of each customer’s details like their name, address, email\_id, their cust\_id. Each call made by the customer is also recorded with information like the problem they are facing and other details like the duration of the call and the time and date the call was made.

# Data Model

This is the ER diagram of the database model.



# SCHEMA:

DEPARTMENT

|  |  |  |
| --- | --- | --- |
| DEPT\_ID | DEPT\_NAME | NO\_OF\_EMP |

EMPLOYEE

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| EMP\_ID | EMP\_NAME | DEPT\_ID | SALARY | CITY | STATE | ZIP |

SOLUTION\_DESK

|  |  |
| --- | --- |
| SOLUTION\_ID | SPECIALIST\_ID |

PROBLEM

|  |  |  |
| --- | --- | --- |
| PROBLEM\_ID | PROBLEM\_TYPE | SOLUTION\_ID |

CUSTOMER

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| CUST\_ID | NAME | CITY | STATE | ZIP | EMAIL\_ID |

CALL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CALL\_ID | CUST\_ID | OPERATOR | PH\_NO | PROBLEM\_ID |

CALL\_LOG

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| CALLLOG\_ID | DATE | TIME | DURATION(m) | CALL\_ID |

CANDIDATE KEYS IN EACH RELATION:

**DEPARTMENT**: DEPT\_ID DEPT\_NAME  
**EMPLOYEE**: EMP\_ID

**SOLUTION\_DESK**: SOLUTION\_ID

**PROBLEM**: PROBLEM\_ID PROBLEM\_TYPE

**CUSTOMER**: CUST\_ID EMAIL\_ID

**CALL**: CALL\_ID

**CALL\_LOG**: CALLLOG\_ID CALL\_ID

# FD and Normalization

FD in each relation can be calculated by:

If A -> B, then

SELECT A

FROM RELATION

GROUP BY A

HAVING COUNT (DISTINCT B) > 1;

Will return nothing

Else, FD is violated

1. **DEPARTMENT**

DEPT\_ID -> DEPT\_NAME NO\_OF\_EMP

DEPT\_NAME -> DEPT\_ID NO\_OF\_EMP

This relation is in BCNF since for every A -> B, A+ = DEPARTMENT

1. **EMPLOYEE**

EMP\_ID -> EMP\_NAME DEPT\_ID CITY STATE ZIP

This is also in BCNF

1. **SOLUTION\_DESK**

SOLUTION\_ID -> SPECIALIST\_ID

This is also in BCNF

1. **PROBLEM**

PROBLEM\_ID -> PROBLEM\_TYPE SPECIALIST\_ID

PROBLEM\_TYPE -> PROBLEM\_ID SPECIALIST\_ID

This is also in BCNF

1. **CUSTOMER**

CUST\_ID -> NAME CITY STATE ZIP EMAIL\_ID

EMAIL\_ID -> NAME CITY STATE ZIP CUST\_ID

This is also in BCNF

1. **CALL**

CALL\_ID -> CUST\_ID OPERATOR PH\_NO PROBLEM\_ID

This is also in BCNF

1. **CALL\_LOG**

CALLLOG\_ID -> CALL\_ID DATE TIME DURATION(m)

CALL\_ID -> CALLLOG\_ID DATE TIME DURATION(m)

This is also in BCNF

**Normalization**

**1NF**

For 1NF , we need atomic data. In our database all columns are single valued, hence all the data is atomic. Hence all our functional dependencies are already in 1NF.

**2NF and 3NF**

A relation schema R is in 2NF if every non-prime attribute A in R is fully functionally dependent on the primary key of R.

A relation schema R is in 3NF if it satisfies 2NF and no non-prime attribute of R is transitively dependent on the primary key.

**BCNF**

A relation is in BCNF if for every FD in R, from A -> B A is a super key, i.e A+ = R

# LOSSLESS JOIN PROPERTY:

To check for lossless join decomposition after normalization, the following conditions must hold:

* Union of Attributes of R1, R2 … Rn must be equal to the attributes of R.
* Each attribute of R must be present in the decomposed relations.

When we apply a natural join to all the relations we should get back the original data without any redundant tuples coming in.

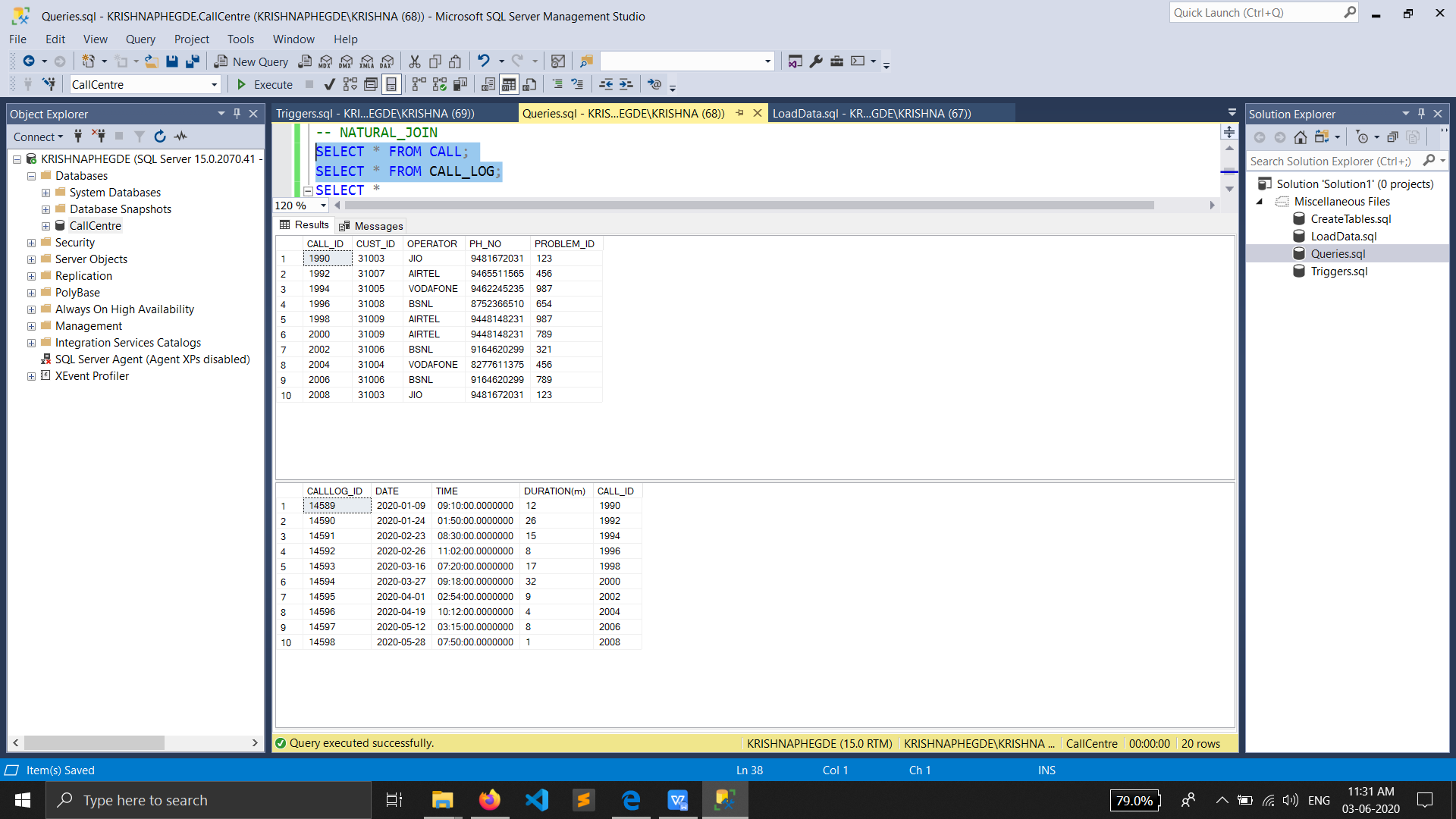
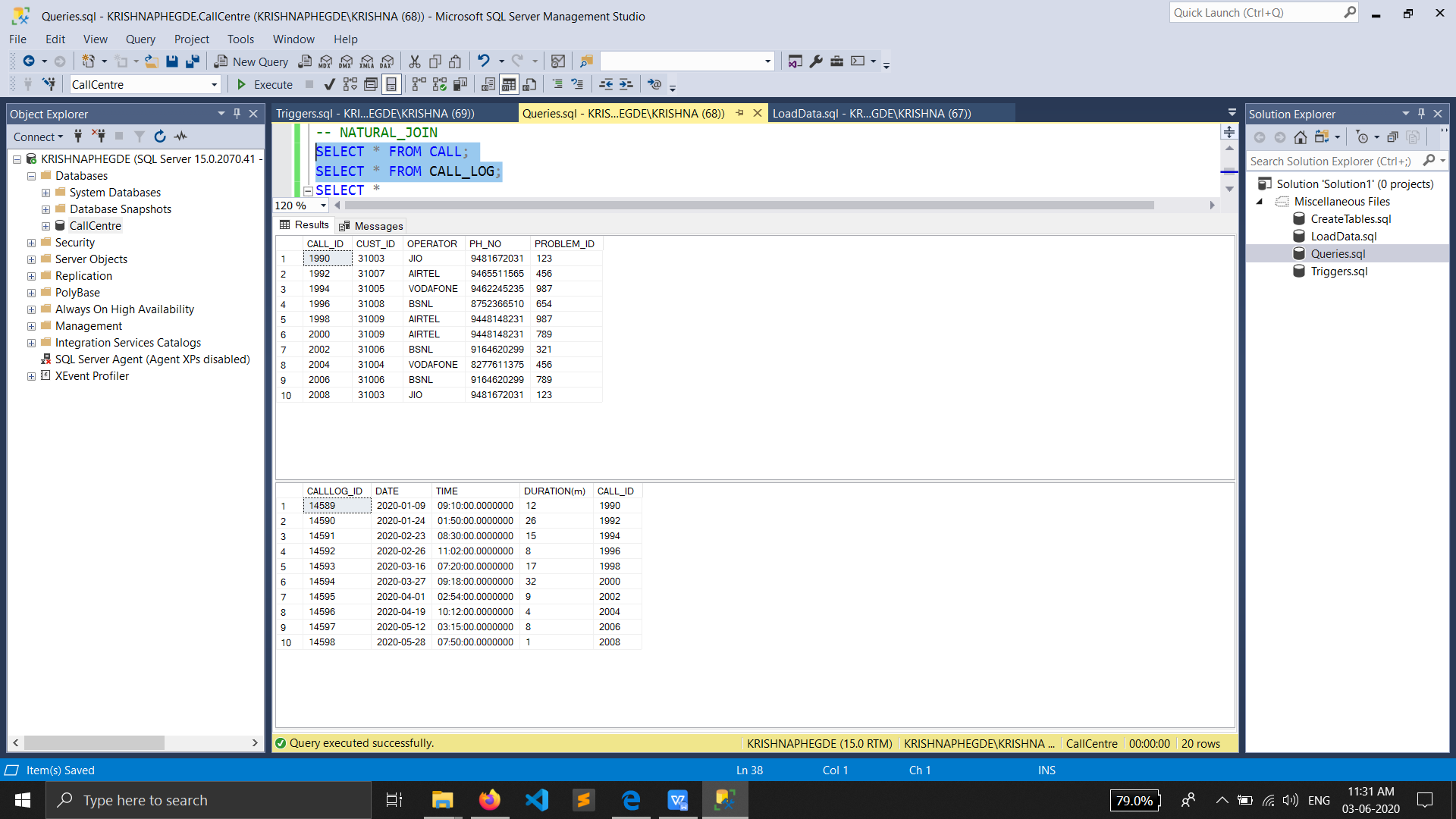
-- NATURAL\_JOIN

SELECT \*

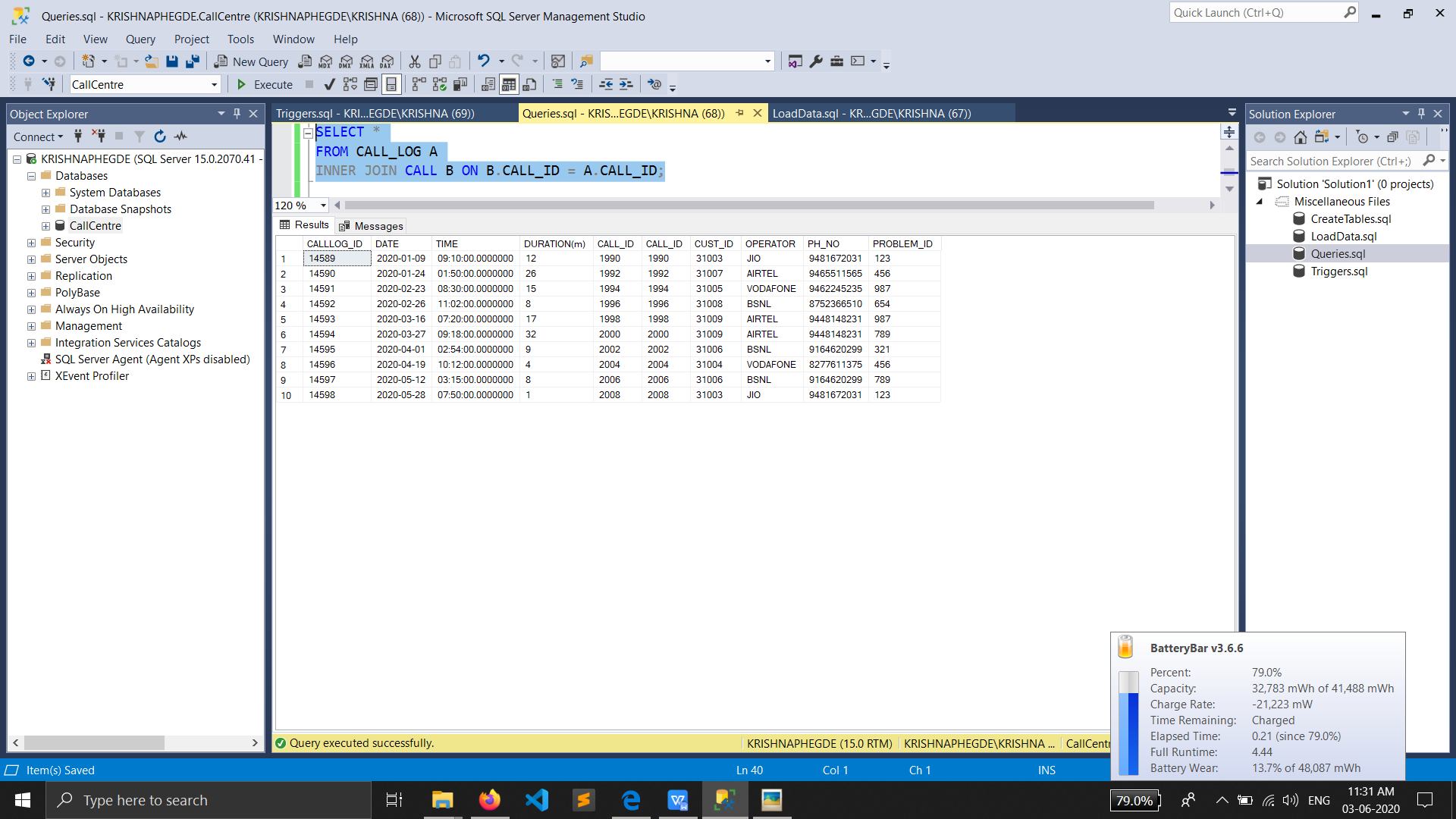
FROM CALL\_LOG A

INNER JOIN CALL B ON B.CALL\_ID = A.CALL\_ID;

HERE R1 and R2 are:



AND THE NATURAL JOIN OF R1 and R2 results in Lossless join



# DDL

-- Creating the table with check constraint

CREATE TABLE DEPARTMENT (

DEPT\_ID INT NOT NULL PRIMARY KEY,

DEPT\_NAME VARCHAR(100) UNIQUE,

NO\_OF\_EMP INT CHECK(NO\_OF\_EMP>=10)

);

CREATE TABLE EMPLOYEE (

EMP\_ID INT NOT NULL PRIMARY KEY,

EMP\_NAME VARCHAR(100) NOT NULL,

DEPT\_ID INT FOREIGN KEY REFERENCES DEPARTMENT(DEPT\_ID) ON DELETE SET NULL, -- Referential Integrity Constraint

CITY VARCHAR(100),

STATE VARCHAR(100),

ZIP INT

);

CREATE TABLE SOLUTION\_DESK (

SPECIALIST\_ID INT NOT NULL PRIMARY KEY,

SOLUTION\_ID INT NOT NULL UNIQUE

);

CREATE TABLE PROBLEM (

PROBLEM\_ID INT NOT NULL PRIMARY KEY,

SPECIALIST\_ID INT FOREIGN KEY REFERENCES SOLUTION\_DESK(SPECIALIST\_ID) ON UPDATE CASCADE, -- Referential Integrity Constraint

PROBLEM\_TYPE VARCHAR(100) NOT NULL

);

-- Creating the table with check constraint

CREATE TABLE CUSTOMER (

CUST\_ID INT NOT NULL PRIMARY KEY,

NAME VARCHAR(100) NOT NULL,

CITY VARCHAR(100),

STATE VARCHAR(100),

ZIP INT,

EMAIL\_ID VARCHAR(100) UNIQUE CHECK (EMAIL\_ID LIKE '%\_@\_\_\_\_%.com')

);

CREATE TABLE CALL (

CALL\_ID INT IDENTITY(1990, 2) NOT NULL PRIMARY KEY,

CUST\_ID INT FOREIGN KEY REFERENCES CUSTOMER(CUST\_ID) ON DELETE CASCADE, -- Referential Integrity Constraint

OPERATOR VARCHAR(100) DEFAULT 'JIO',

PH\_NO VARCHAR(10) NOT NULL,

PROBLEM\_ID INT FOREIGN KEY REFERENCES PROBLEM(PROBLEM\_ID) ON DELETE CASCADE -- Referential Integrity Constraint

);

CREATE TABLE CALL\_LOG (

CALLLOG\_ID INT IDENTITY(14589, 1) NOT NULL PRIMARY KEY,

DATE DATE,

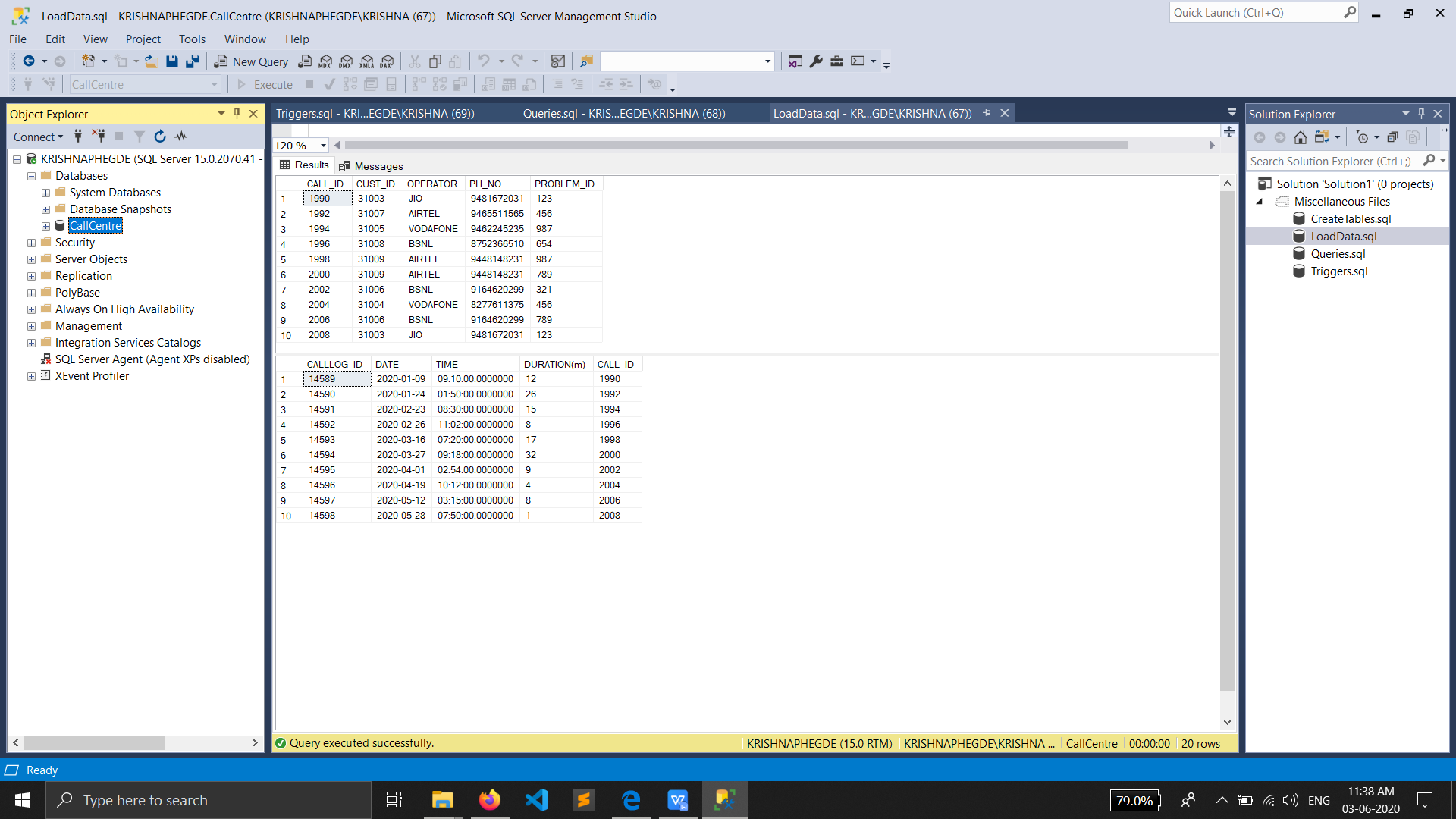
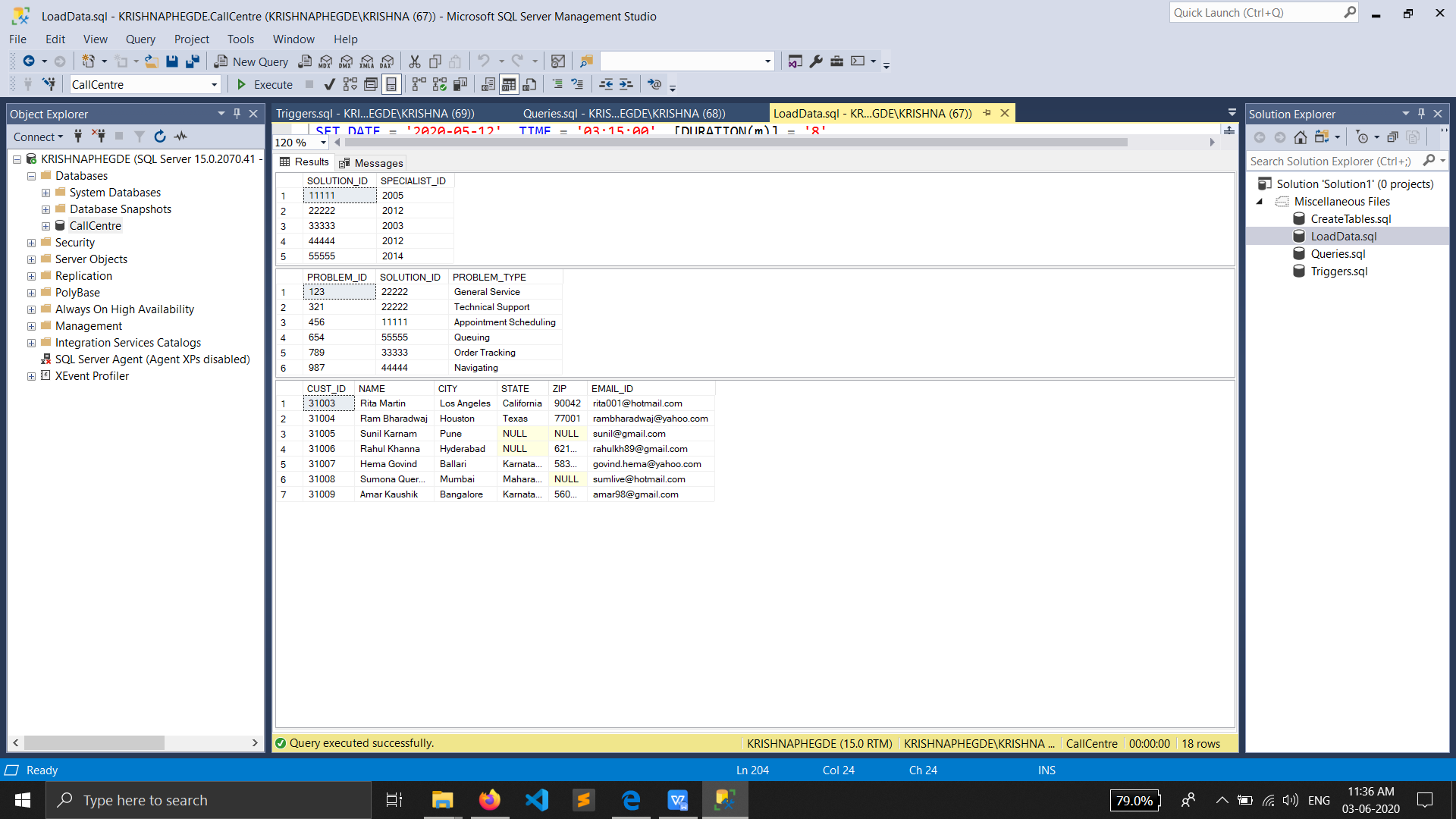
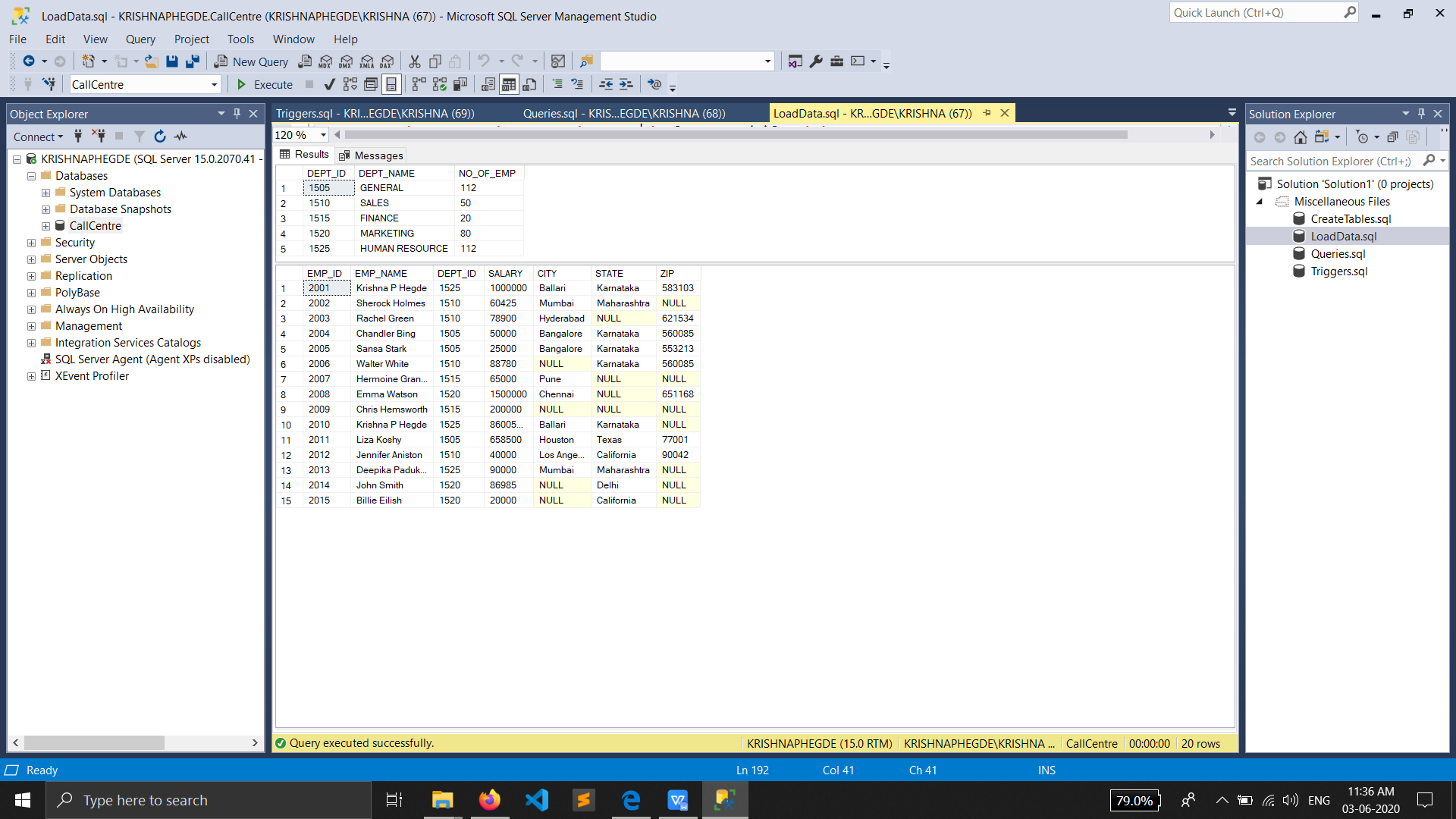
TIME TIME,

[DURATION(m)] INT,

CALL\_ID INT FOREIGN KEY REFERENCES CALL(CALL\_ID) ON DELETE CASCADE -- Referential Integrity Constraint

# );

# After loading all the data records in the table:



# Triggers

I have created a AFTER INSERT trigger in the CALL table, which creates a new record in the CALL\_LOG table which stores the duration and other details of each call identified by the CALL\_ID.

-- DML Trigger

CREATE TRIGGER TRIG\_ONE ON CALL

AFTER INSERT

AS

BEGIN

INSERT INTO CALL\_LOG (CALL\_ID)

SELECT INSERTED.CALL\_ID

FROM INSERTED;

END

# SQL Queries

1. Nested Query

This query returns the department name of the employee who has the highest salary.

SELECT A.DEPT\_NAME

FROM DEPARTMENT A

WHERE A.DEPT\_ID IN ( SELECT B.DEPT\_ID

FROM EMPLOYEE B

WHERE B.SALARY IN (SELECT MAX(SALARY) FROM EMPLOYEE)

);

This query returns the operator used by customers whose EMAIL\_ID starts with ‘r’

SELECT DISTINCT A.OPERATOR

FROM CALL A

WHERE A.CUST\_ID IN ( SELECT B.CUST\_ID

FROM CUSTOMER B

WHERE B.EMAIL\_ID LIKE 'R%'

);

1. Aggregate Queries

These are the queries for finding the minimum, maximum and average duration of all the calls received by the CallCentre. And also count the total number of departments.

SELECT min([DURATION(m)]) AS Min\_duration

FROM CALL\_LOG;

SELECT max([DURATION(m)]) AS Max\_duration

FROM CALL\_LOG;

SELECT avg([DURATION(m)]) AS Avg\_duration

FROM CALL\_LOG;

SELECT count(DEPT\_ID) AS [No of departments]

FROM DEPARTMENT;

# Outer Join Query

This is a query used to find the employees and the customers who share the same city.

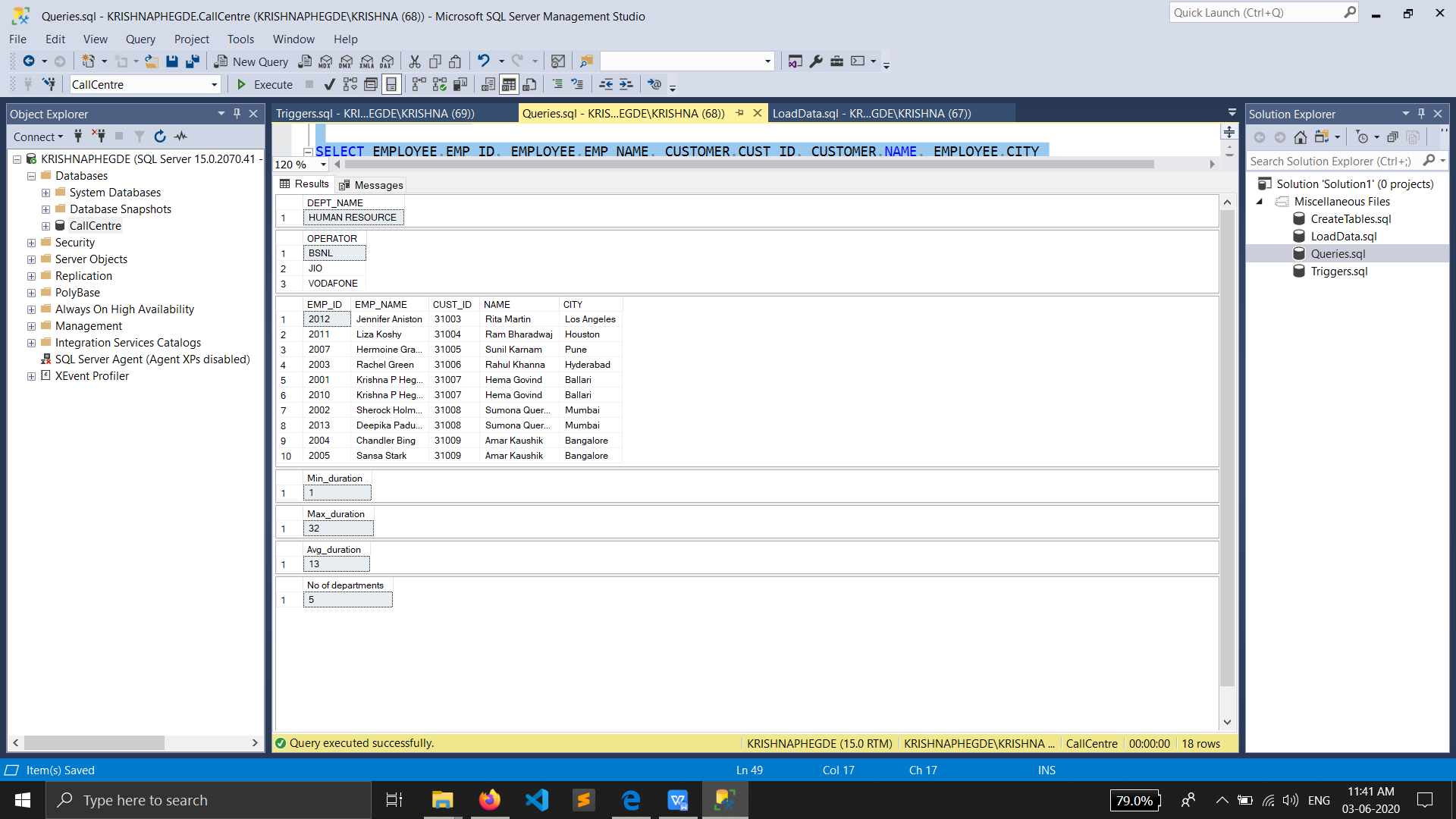
SELECT EMPLOYEE.EMP\_ID, EMPLOYEE.EMP\_NAME, CUSTOMER.CUST\_ID, CUSTOMER.NAME, EMPLOYEE.CITY

FROM CUSTOMER

LEFT JOIN EMPLOYEE ON EMPLOYEE.CITY=CUSTOMER.CITY

ORDER BY CUSTOMER.CUST\_ID;

OUTPUTS OF ALL THE QUERIES:



# Conclusion

In conclusion, the database has many capabilities of storing large amount of records with clear meaning of each record, and be used to retrieve complex queries.

It has few limitations also, like the administrator can add few more attributes in the tables like employee and customer. It is pretty simple and can be used for managing small call-centre firms.