Jabari Fowler | Jade Gee | John Cruz

Call Center Database

Abstract

This call center database project serves as a collection of call center interactions for a general company. This application will allow employees to log their interactions with customers to document problems and/or have a recommended solution given to them. When a customer calls in, it will record their interactions and see if they have previously contacted the company for the same or similar reason. With this, we can document any troubleshooting steps that were taken to solve the customer’s issue; and, by compiling records of solutions attached to specific problems, we will be able to find a correlation or most effective method for rectifying these issues. While the employee helps the customer, the employee will have access to recommended solutions to solve the issue for the customer more efficiently.

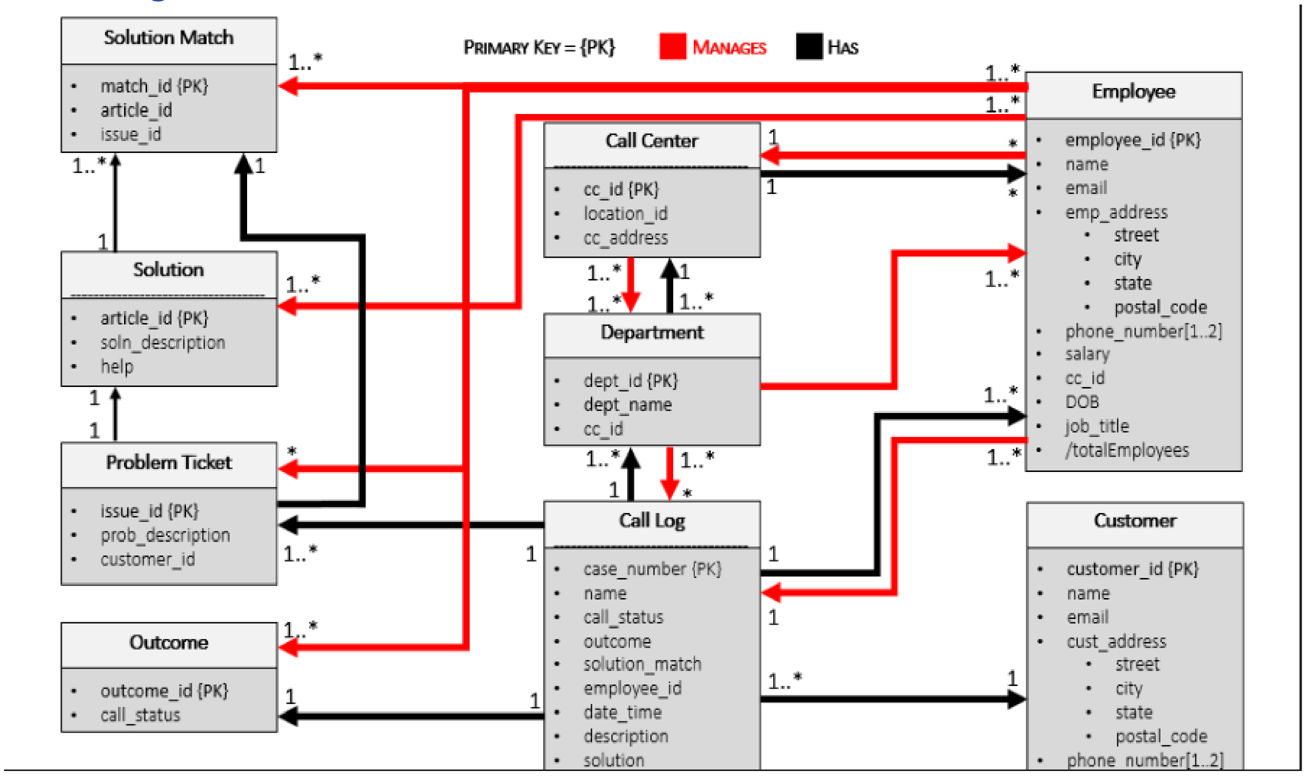
Mission Statement

The purpose of the Call Center Database system is to maintain a record of generated problems and solutions to troubleshoot and provide a resolution to customers and developers.

Mission Objectives

1. To maintain (enter, update, and delete) data on customers.
2. To maintain (enter, update, and delete) data on employees.
3. To maintain (enter, update, and delete) data on departments.
4. To maintain (enter, update, and delete) data on problems.
5. To maintain (enter, update, and delete) data on solutions.
6. To maintain (enter, update, and delete) data on outcomes.
7. To maintain (enter, update, and delete) data on statuses.
8. To maintain (enter, update, and delete) data on matches.
9. To maintain (enter, update, and delete) call centers.
10. To maintain (enter, update, and delete) call logs.
11. To perform searches on customers.
12. To perform searches on employees.
13. To perform searches on departments.
14. To perform searches on call centers.
15. To perform searches on call logs.
16. To perform searches on problems,.
17. To perform searches on solutions.
18. To perform searches on outcomes.
19. To perform searches on statuses.
20. To perform searches on matches.
21. To track the status of call outcomes.
22. To report the outcome of customer issues.
23. To report correlations between certain problems and solutions.
24. To report the employee’s success rate at solving customer issues.

E/R Diagram


PRIMARY KEY = {PK}
🟥 MANAGES
⬛ HAS
Solution Match
match_id {PK}
article_id
issue_id
Solution
article_id {PK}
soln_description
help
Problem Ticket
issue_id {PK}
prob_description
customer_id
Outcome
outcome_id {PK}
call_status
Call Center
cc_id {PK}
location_id
cc_address
Department
dept_id {PK}
dept_name
cc_id
Call Log
case_number {PK}
name
call_status
outcome
solution_match
employee_id
date_time
description
solution
Employee
employee_id {PK}
name
email
emp_address
street
city
state
postal_code
phone_number[1..2]
salary
cc_id
DOB
job_title
/totalEmployees
Customer
customer_id {PK}
name
email
cust_address
street
city
state
postal_code
phone_number[1..2]
1..*
1
*

July 26, 2020
CS 4318 Phase 2

Relational Model

Call Log

case_number{PK}	Name{FK}	outcome_id{FK}	Description	match_id{FK}	date	emplo
100001	Johnny Dang	98	Will Not power on	101	7/30/2020	4
100002	Lewis Carlton	100	Broken Screen	102	9/20/2020	4
100003	Jordan Davis	100	Broken Screen	103	9/27/2020	4
100004	Ritz Rich	98	Camera Not Functioning	104	11/16/2020	4
100005	Chester Chet	95	Broken Screen	105	12/4/2020	4
100006	Pierre Davis	102	Will Not Power On	106	8/2/2020	4
100007	Dahlia Allison	100	Slow Performance	107	10/5/2020	4
100008	Katarina Moreno	95	Camera Not Functioning	108	11/20/2020	4
100009	Jake Cannon	98	Broken Screen	109	12/3/2020	4
100010	Clarice Mcneill	100	Broken Screen	110	12/30/2020	4
100011	Sophia Andrew	98	Camera Not Functioning	111	8/6/2020	4
100012	Faizah Palmer	98	Broken Screen	112	9/4/2020	4
100013	Mercedes Maddox	95	Slow Performance	113	11/14/2020	4

Customer

customer_id{AK}	name{PK}	email	phone_number	street	state	city	zip
2001	Johnny Dang	johnny.d@gmail.com	832-999-9813	3555 Graystone	GA	Macon	31201
2002	Lewis Carlton	lews.c@yahoo.com	832-789-9814	2068 Lonely Oak Dr.	AL	Mobile	36575
2003	Jordan Davis	jordan.d@hotmail.com	602-548-1612	3100 Coplin Av.	AZ	Phoenix	85023
2004	Ritz Rich	ritz@gmail.com	910-343-8437	80 Ray Court	MO	Ellington	63638
2005	Chester Chet	chester.c@yahoo.com	616-529-8587	2992 Howard Street	MO	Belgrade	63622
2006	Pierre Davis	pierre.d@gmail.com	508-243-5310	3601 Stadium Dr.	MA	Taunton	12780
2007	Dahlia Allison	dahliaA@gmail.com	570-639-0103	2957 Coal Rd.	PA	Harveys Lake	18618
2008	Katarina Moreno	kata.M@gmail.com	352-498-0511	1857 George Street	FL	Cross City	32628
2009	Jake Cannon	jakeC@yahoo.com	856-461-6642	4677 Briarwood Dr.	NJ	Riverside	18075
2010	Clarice Mcneill	clarice@mail.com	603-842-7809	4628 Peck St.	NH	Dover	13820
2011	Sophia Andrew	sophia@mail.com	409-275-8855	4652 Brookview Dr.	TX	Beaumont	77701
2012	Faizah Palmer	faizah@mail.com	586-457-4023	1153 D Street	MI	Bloomfield	48302
2013	Mercedes Maddox	mercedes@gmail.com	217-439-8097	2121 Isaacs Creek Rd.	MN	Greenwald	56335

Problem Ticket

issue_id{PK}	description	customer_id{FK}	
205	Will Not power on	2001	2001
206	Broken Screen	2002	2002
207	Broken Screen	2003	2003
208	Camera Not Functioning	2004	2004
209	Broken Screen	2005	2005
210	Will Not Power On	2006	2006
211	Slow Performance	2007	2007
212	Camera Not Functioning	2008	2008
213	Broken Screen	2009	2009
214	Broken Screen	2010	2010
215	Camera Not Functioning	2011	2011
216	Broken Screen	2012	2012
217	Slow Performance	2013	2013

Solution Match

match_id{PK}	article_id{FK}	issue_id{FK}
101	8900	205
102	8901	206
103	8901	207
104	8901	208
105	8901	209
106	8900	210
107	8902	211
108	8901	212
109	8901	213
110	8901	214
111	8901	215
112	8901	216
113	8902	217

Issue	Solution
0 - WNP	100 - restart
1 - BS	101 - repair
2 - CNF	102 - erase and restore
3 - SP	

Jabari Fowler | Jade Gee | John Cruz

July 26, 2020
CS 4318 Phase 2

Call_Status Codes	Outcome
0 - Close 1 - Ongoing 2 - pending repair 3 - manager takeover	0 - Dissatisfied 1 - No answer 2 - Satisfied

Call Centers
895 - Southern Call Center 795 - Northern Call Center 1005 - Corporate Office

Employee

emloyee_id{PK}	name	email	street	city	state	zip	phone_number	salary	cc_id{FK}
4500	Maria Campbell	maria.c@helpc.com	2759 Roguski Rd.	Natchitoches	LA	71457	318-352-0772	25,000$	895
4600	Vincent Talley	vince.t@helpc.com	2460 Willson St.	Andover	MN	55304	763-413-6693	27,000$	795
4700	Donald Everitt	donald.e@helpc.com	4017 Hedge St.	Andover	MN	55306	908-620-6680	45,000$	795
4800	Robert Bennett	rob.b@helpc.com	4334 Williams Ave.	Natchitoches	LA	71457	661-330-6710	75,000$	895
5000	David Mitchell	david.m@helpc.com	4648 Shinn Ave.	Gibsonia	PA	15044	724-558-5918	120,000$	1005

Call Center

cc_id{PK}	cc_address
795	2919 Sugar Camp Rd. MN
895	3585 Sara Dr., LA
1005	4648 Shinn Av., PA

Department

dept_id{PK}	dept_name	cc_id{FK}
746	Tech-NCC	795
747	Repairs-NCC	795
846	Tech-SCC	895
847	Repairs-SCC	895

Outcome

outcome_id{PK}	outcome_desc
95	Ongoing
98	Closed
100	Pending Repair
102	Manager Escalation

Solution

article_id{PK}	soln_desc	help
8900	restart device first	erase if needed
8901	setup repair	
8902	erase and restore	

Call Log

Attribute	Domain Name	Meaning	Data Constraints	Defaults
Case_Number	Case Number	The set of all possible case numbers	Not Null, Unique	
Name	Name	The set of all customer names	Not Null	char = 'No Customer'
outcome_id	Outcome	The set of all possible values of outcomes	Not Null	int = 95
match_id	MatchID	The set of all possible matchID numbers(from Solution Match)	Not Null, Unique	no default
employee_id	Employee	The set of all possible employee ID's (from Employee)	Not Null	no default
date_time	Date/Time	Possible values of call date and time	Not Null	current date and time
description	Description	The set of all possible call descriptions	Null	null
issue_id	Issue	The set of all possible Issues (from problem ticket)	Not Null, Unique	

Customer

Attribute	Domain Name	Meaning	Data Constraints	Defaults
customer_id	Customer ID	The set of all possible customer ID Numbers	Not Null, Unique	
name	Name	The set of all customer names	Not Null, Unique	char = 'No Name'
email	email	The set of all customer emails in the US	Null	Null
phone_number	Phone Number	The set of all customer phone numbers in US	Not Null	no default
street	street	The set of all customer street names in the US	Not Null	no default
state	state	The set of all customer states in the US	Not Null	no default
city	city	The set of all customer cities in the US	Null	null
zip	ZipCode	The set pf all customer zip codes in the US	Not Null	no default

Problem Ticket

Attribute	Domain Name	Meaning	Data Constraints	Defaults
issue_id	Issue	The set of all possible Issues	Not Null, Unique	no default
description	DescriptionTick	The set of all possible ticket descriptions	Null	char = "No additional Notes"

customer_id	Customer ID	The set of all customer ID Numbers(from customer)	Not Null	no default	

Call Center

Attribute	Domain Name	Meaning	Data Constraints	Defaults	D
cc_id	Call Center ID	The set of all Call center ID's in the company	Not Null	no default	i
cc_address	CCAddress	The set of all Call Center Addresses for call centers in the company	Not Null	char = "No Address Entered"	c

Department

Attribute	Domain Name	Meaning	Data Constraints	Defaults	D
dept_id	DepartmentID	The set of all Departments within the call center	Not Null, Unique	no default	i
dept_name	DepartmentName	The set of all names of the department within the call center	Not Null, Unique	no default	c
cc_id	Call Center ID	The set of all Call center ID's in the company (from Call Center)	Not Null	no default	i

Outcome

Attribute	Domain Name	Meaning	Data Constraints	Defaults	D
outcome_id	Outcome	The set of all possible values of call outcomes	Not Null	int = 95	in
outcome_desc	OutcomeDesc	Possible descriptions for call outcomes	Not Null	char = "Ongoing"	c

Solution

Attribute	Domain Name	Meaning	Data Constraints	Defaults	D
article_id	Article ID	The set of all possible article ID numbers	Null	int = 8900	i
soln_desc	SolutionDescription	Possible descriptions for solutions for calls	Not Null	no default	c
help	help	Possible help or additional descriptions for solutions for calls	null	char = "No additional Notes"	c

Solution Match

Attribute	Domain Name	Meaning	Data Constraints	Defaults	
match_id	MatchID	The set of all possible matchID numbers(from Solution Match)	Not Null, Unique	no default	
article_id	Article ID	The set of all possible article ID numbers (from solution)	Null	int = 8900	
issue_id	Issue	The set of all possible Issues (from problem ticket)	Not Null, Unique	no default	

Employee

Attribute	Domain Name	Meaning	Data Constraints	Defaults	
employee_id	Employee	The set of all possible employee ID's (from Employee)	Not Null	no default	
Name	EmpName	The set of all employee names in the company	Not Null		
email	email	The set of all Employee Email in the company	Null	null	
street	street	The set of all Employees Address's in the company	Not Null	no default	
state	state	The set of all Employees State in the company	Not Null	no default	
city	city	The set of all Employees City in the company	Null	null	
zip	ZipCode	The set of all Employee Zip Code's in the company	Not Null	no default	
salary	Salary	Possible values of Employee Salaries	Not Null	no default	
cc_id	Call Center ID	The set of all Call center ID's (from Call Center) in the company	Null	null	
DOB	Dateof_Birth	Possible Values of employee birth dates	not null	1/1/1990	
job_title	JobTitle	The set of all job titles for employees inside the company	Not Null	char = "Employee"	

Use Cases

Insertion

1. **Entering a new employee**
    a. Actor: Manager
    b. Steps:
        i. User clicks “new” button; user selects employee
        ii. A new employee ID is generated and displayed
        iii. Prompts user to enter first name, last name, job title, salary, call center information, DOB, email, and phone number
        iv. All information is displayed; ask for confirmation
        v. User clicks “Confirm” button to save the new employee
    c. SQL statement:
        i. INSERT INTO Employee VALUES (employee_id, name, email, emp_address, phone_number, salary, cc_id, DOB, job_title)
    d. Use case realization:
        i. This will allow a manager to enter an employee and their contact information when they have a new hire
2. **Entering a new customer**
    a. Actor: Employee
    b. Steps:
        i. User clicks “new” button; user selects customer
        ii. A new customer ID is generated and displayed
        iii. Prompts user to enter name, address, email, and phone number
        iv. All information is displayed and asks for confirmation
        v. User clicks “confirm” button to save the new customer
    c. SQL statement:
        i. INSERT INTO Customer VALUES (customer_id, name, email, cost_address, phone_number, prob_description)
    d. Use case realization:
        i. This will allow the user to input a new customer and their information so that they can reference the customer if they call again or to address their issue
3. **Entering a new problem ticket**
    a. Actor: Employee
    b. Steps:
        i. User clicks “New”; user selects problem ticket
        ii. A new issue ID is generated and displayed
        iii. Prompts user to search for/select a customer and enter problem description.
        iv. All information is displayed and asks for confirmation
        v. User clicks “confirm” button to save the new problem ticket
    c. SQL statement:
        i. INSERT INTO Problem Ticket VALUES (Issue_id, prob_description, customer_id)
    d. Use case realization:
        i. This will allow the user to input a new problem ticket for a customer
4. **Entering a new solution**
    a. Actor: Employee, Supervisor
    b. Steps:
        i. User clicks “new” button; user selects solution
        ii. A new article ID is generated and displayed
        iii. Prompts user to enter a description of the solution and the problem issue ID
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button” to save the new solution
    c. SQL statement:

            i. INSERT INTO Solution VALUES (article_id, soln_description, help)
    d. Use case realization:
            i. Allows the user to enter a new solution to a problem ticket

5. **Entering a new call log**
    a. Actor: Employee, Supervisor
    b. Steps:
        i. User clicks “new” button; user selects call log
        ii. A new case number is generated and displayed
        iii. Prompts user to enter or select a customer, date and time, problem description, status, and outcome
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button” to save the new solution
    c. SQL statement:
        i. INSERT INTO Call Log VALUES (case_number, name, call_status , outcome_id, solution_match, employee_id, date_time, description, solution, issue_id)
    d. Use case realization:
        i. Allows user to insert a new call log when a customer calls in
6. **Entering a new call center**
    a. Actor: Director
    b. Steps:
        i. User clicks “new” button; user selects call center
        ii. A new call center ID is generated and displayed
        iii. Prompts user to enter the location id and address of the new call center
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button” to save the new call center
    c. SQL statement:
        i. INSERT INTO Call Center VALUES (cc_id, location_id, cc_address)
    d. Use case realization:
        i. Allows the director to enter a new call center when they open a new one
7. **Entering a new outcome**
    a. Actor: Employee, Supervisor
    b. Steps:
        i. User clicks “new” button; user selects outcome
        ii. Prompts user to select an outcome of the call from a dropdown menu
        iii. All information is displayed and asks for confirmation
        iv. User clicks on “Confirm Button” to save the new outcome
    c. SQL statement:
        i. INSERT INTO Outcome VALUES (outcome_id, call_status)
    d. Use case realization:
        i. Allows user to enter a new outcome on a problem ticket
8. **Entering a new solution match**
    a. Actor: Supervisor
    b. Steps:
        i. User clicks “new” button; user selects solution match
        ii. Prompts user to select a solution and links it to the open case and problem ticket.
        iii. All information is displayed and asks for confirmation
        iv. User clicks on “Confirm Button” to save the new solution match
    c. SQL statement:
        i. INSERT INTO Solution Match VALUES (match_id, article_id, issue_id)
    d. Use case realization:
        i. Allows user to match a solution to an issue for common problems
9. **Entering a new department**
    a. Actor: Director, Manager

b. Steps:
    i. User clicks "new" button; user selects department
    ii. A new department ID is generated and displayed
    iii. Prompts user to enter the department name and decription
    iv. All information is displayed and asks for confirmation
    v. User clicks on "Confirm Button" to save the new department
c. SQL statement:
    i. INSERT INTO Department VALUES (dept_id, dept_name, cc_id)
d. Use case realization:
    i. Allows user to enter a new department when one is created

Removal

10. **Deleting an employee**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks "remove" button; user selects employee
        ii. Prompts user to select call center ID and employee
        iii. All information is displayed and asks for confirmation
        iv. User clicks on "Delete Entry Button"
    c. SQL statement:
        i. DELETE FROM Employee WHERE employee_id = 'employee_id'
    d. Use case realization:
        i. Allows user to remove an employee upon termination of employment
11. **Deleting a customer**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks "remove" button; user selects customer
        ii. Prompts user to select a customer
        iii. All information is displayed and asks for confirmation
        iv. User clicks on "Delete Entry Button"
    c. SQL statement:
        i. DELETE FROM Customer WHERE customer_id = 'customer_id'
    d. Use case realization:
        i. Allows user to remove a customer, in the event that the customer is no longer associated with the company
12. **Deleting a problem ticket**
    a. Actor: Supervisor
    b. Steps:
        i. User clicks "remove" button; user selects problem ticket
        ii. Prompts user to select a call center, then the problem ticket
        iii. All information is displayed and asks for confirmation
        iv. User clicks on "Delete Entry Button"
    c. SQL statement:
        i. DELETE FROM Problem Ticket WHERE issue_id = 'issue_id'
    d. Use case realization:
        i. Allows the user to remove a problem ticket in the event that one was created by mistake
13. **Deleting a solution**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks "remove" button; user selects solution
        ii. Prompts user to select a solution.
        iii. All information is displayed and asks for confirmation

        iv. User clicks on “Delete Entry Button”
    c. SQL statement:
        i. DELETE FROM Solution WHERE article_id = ‘article_id’
    d. Use case realization:
        i. Allows user to remove a solution in the event that the solution becomes outdated or ineffective

14. **Deleting a call log**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks “remove” button; user selects call log
        ii. Prompts user to select a call center, date and time, then the call log
        iii. All information is displayed and asks for confirmation
        iv. User clicks on “Delete Entry Button”
    c. SQL statement:
        i. DELETE FROM Call Log WHERE case_number = ‘case_number’
    d. Use case realization:
        i. Allows the user to remove a call log
15. **Deleting a call center**
    a. Actor: Director
    b. Steps:
        i. User clicks “remove” button; user selects call center
        ii. Prompts user to select a call center
        iii. All information is displayed and asks for confirmation
        iv. User clicks on “Delete Entry Button”
    c. SQL statement:
        i. DELETE FROM Call Center WHERE cc_id = ‘cc_id’
    d. Use case realization:
        i. Allows the user to remove a call center in the event of a call center closure
16. **Deleting an outcome**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks “remove”; user selects outcome
        ii. Prompts user to select a customer and problem ticket, then the outcome
        iii. All information is displayed and asks for confirmation
        iv. User clicks on “Delete Entry Button”
    c. SQL statement:
        i. DELETE FROM Outcome WHERE outcome_id = ‘outcome_id’
    d. Use case realization:
        i. Allows the user to remove an outcome in the event that an outcome was not established
17. **Deleting a solution match**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks “remove” button; user selects solution match
        ii. Prompts user to select a solution or a problem, then the solution match
        iii. All information is displayed and asks for confirmation
        iv. User clicks on “Delete Entry Button”
    c. SQL statement:
        i. DELETE FROM Solution Match WHERE match_id = ‘match_id’
    d. Use case realization:
        i. Allows user to remove a solution match, in the event that the incorrect match had been inputted
18. **Deleting a department**
    a. Actor: Director
    b. Steps:
        i. User clicks “remove” button; user selects department

   ii. Prompts user to select a call center and department ID
   iii. All information is displayed and asks for confirmation
   iv. User clicks on "Delete Entry Button"

c. SQL statement:
   i. DELETE FROM Department WHERE dept_id = 'dept_id'
d. Use case realization:
   i. Allows the user to delete a department if a department closure occurs

Management

19. **Updating an employee**
    a. Actor: Manager, Supervisor
    b. Steps:
       i. User clicks "update" button
       ii. Prompts user to select a which entity to update
       iii. User selects employee and is prompted to select a department and employee
       iv. Prompts user to update employee information
       v. All information is displayed and asks for confirmation
       vi. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Employee SET 'column' = 'new column value' WHERE employee_id = 'some_employee_id
    d. Use case realization:
       i. Allows the user to update the contact information for an employee when necessary
20. **Updating a customer**
    a. Actor: Manager, Supervisor, Employee
    b. Steps:
       i. User clicks "update" button
       ii. Prompts user to select a which entity to update
       iii. User selects customer and is prompted to select a customer
       iv. Prompts user to update customer information
       v. All information is displayed and asks for confirmation
       vi. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Customer SET 'column' = 'new column value' WHERE customer_id = 'some_customer_id
    d. Use case realization:
       i. Allows the user to update the customer contact information when necessary
21. **Updating a problem ticket**
    a. Actor: Supervisor, Employee
    b. Steps:
       i. User clicks "update" button
       ii. Prompts user to select a which entity to update
       iii. User selects problem ticket and is prompted to select a problem ticket
       iv. Prompts user to update problem ticket information
       v. All information is displayed and asks for confirmation
       vi. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Problem Ticket SET 'column' = 'new column value' WHERE issue_id = 'some_issue_id'
    d. Use case realization:
       i. Allows the user to update the information on a problem ticket
22. **Updating a solution**
    a. Actor: Manager, Supervisor
    b. Steps:

        i. User clicks “update” button”
        ii. Prompts user to select a which entity to update
        iii. User selects solution and is prompted to update the solution information
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button”
    c. SQL statement:
        i. UPDATE Solution SET ‘column’ = ‘new column value’ WHERE article_id = ‘some_article_id’
    d. Use case realization:
        i. Allows user to update a solution to problem if the solution changes

23. **Updating a call log**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks “update” button
        ii. Prompts user to select a which entity to update
        iii. User selects call log and is prompted to update the call log information
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button”
    c. SQL statement:
        i. UPDATE Call log SET ‘column’ = ‘new column value’ WHERE case_number = ‘case_number’
    d. Use case realization:
        i. Allows the user to update a call log if there are information errors or updates
24. **Updating a call center**
    a. Actor: Director
    b. Steps:
        i. User clicks “update” button
        ii. Prompts user to select a which entity to update
        iii. User selects call center and is prompted to update the call center information
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button”
    c. SQL statement:
        i. UPDATE Call Center SET ‘column’ = ‘new column value’ WHERE cc_id = ‘cc_id’
    d. Use case realization:
        i. Allows the user to update call center information in the event of address changes, etc.
25. **Updating an outcome**
    a. Actor: Manager, Supervisor
    b. Steps:
        i. User clicks “update” button
        ii. Prompts user to select a which entity to update and user selects outcomes/status
        iii. Prompts user to update the outcomes/status information
        iv. All information is displayed and asks for confirmation
        v. User clicks on “Confirm Button”
    c. SQL statement:
        i. UPDATE Outcome SET ‘column’ = ‘new column value’ WHERE outcome_id = ‘outcome_id’
    d. Use case realization:
        i. Allows the user to update the outcome if there is a change in the outcome of the problem ticket or if the incorrect outcome was entered.
26. **Updating a solution match**
    a. Actor: Director, Manager
    b. Steps:
        i. User clicks “update” button
        ii. Prompts user to select which entity to update; user selects solution match
        iii. Prompts user to update the solution match information
        iv. All information is displayed and asks for confirmation

v. User clicks on "Confirm Button"

c. SQL statement:
   i. UPDATE Solution Match SET 'column' = 'new column value' WHERE match_id = 'match_id'

d. Use case realization:
   i. Allows the user to update a match if information was entered incorrectly or a solution became outdated/ineffective.

27. **Updating a department**
    a. Actor: Director
    b. Steps:
       i. User clicks "update" button; Prompts user to select which entity to update
       ii. User selects department; prompts user to update the department information
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Department SET 'column' = 'new column value' WHERE dept_id = 'dept_id'
    d. Use case realization:
       i. Allows the user to update the information for each department

Relationships

28. **Assigning an employee to a call center**
    a. Actor: Manager
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select which employee and call center
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Department SET cc_id = 'new cc_id value' WHERE employee_id = 'employee_id'
    d. Use case realization:
       i. Allows the user to assign an employee to a specific call center

29. **Assigning a problem ticket to a customer**
    a. Actor: Employee
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select a customer and the problem ticket
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Problem Ticket SET customer_id = 'new customer_id value' WHERE issue_id = 'issue_id'
    d. Use case realization:
       i. Allows the user to assign a new problem ticket to a customer

30. **Assigning a department to a call center**
    a. Actor: Director
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select a call center and the department
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Department SET cc_id = 'new cc_id value' WHERE dept_id = 'dept_id'
    d. Use case realization:

      i. Allows the user to assign a department to a call center if a new call center is opened or a new department is formed

31. **Assigning a solution to a problem ticket**
    a. Actor: Director
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select a problem ticket and the solution
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Solution Match SET (article_id = 'new article_id value' issue_id = 'new_issue_id_value') WHERE dept_id = 'dept_id'
    d. Use case realization:
       i. Assigns a solution to a problem ticket once the issue is resolved
32. **Assigning an outcome to a problem ticket**
    a. Actor: Employee, Supervisor, Manager
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select a problem ticket and outcome option
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Problem Ticket SET outcome_id = 'new outcome_id value' WHERE issue_id = 'issue_id'
    d. Use case realization:
       i. Assigns an outcome to a problem ticket once the issue has been solved or escalated
33. **Assigning additional staff member to a case number**
    a. Actor: Supervisor
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select which entity to assign; then which entity it is being assigned to
       iii. User selects employee and case number
       iv. Information is displayed and asks for confirmation to assign an additional employee
       v. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Call Log SET employee_id = 'old_employee_id_value' + ", " + 'new_employee_id_value' WHERE case_number = 'case_number'
    d. Use case realization:
       i. Assigns an additional staff member to a case if the case gets escalated
34. **Assigning a call status to a case number**
    a. Actor: Employee, Supervisor
    b. Steps:
       i. User clicks "assign" button
       ii. Prompts user to select a case number; then prompts user to select the status of the call
       iii. All information is displayed and asks for confirmation
       iv. User clicks on "Confirm Button"
    c. SQL statement:
       i. UPDATE Department SET 'column' = 'new column value' WHERE dept_id = 'dept_id'
    d. Use case realization:
       i. Assigns a call status to a case number
35. **Query a returning customer**
    a. Actor: Employee, Manager, Supervisor
    b. Steps:
       i. User clicks "Find" button

        ii. Prompts user to select which entity to search by; user selects customer
        iii. Prompts user to enter the customer's name;
        iv. Call history is displayed for the requested customer's information
        v. User clicks on "Close Button" to exit
    c. SQL statement:
        i. SELECT name, email, phone_number FROM Customer WHERE customer_id = customer_id
    d. Use case realization:
        i. Allows the user to search and locate a customer if they are a returning customer which allows the user to bypass entering all of the information for the returning customer.

36. **Query all call center tickets (call enter -> employee -> call logs)**
    a. Actor: Manager
    b. Steps:
        i. Click a "call log" button
        ii. Displays all call logs
        iii. Prompts user to select a date filter and/or employee
        iv. Click on a call ID to pull up the outcome
    c. SQL statement:
        i. SELECT cc.cc_id FROM cc (Call Center), e (Employee),cl (Call Log) WHERE cc.cc_id = e.cc_id AND e.emp_id = cl.emp_id
    d. Use case realization
        i. In order to narrow down the amount of data collected by all the call centers into Call Logs, we can match the call logs to the employee who conducted the call by the emp_id in both relations. Further narrow down the call to a Call center we can then match the employees to their call center using the cc_id in both relations. Thus, we can see what call belongs to which Call center.

37. **Query all tickets from employee (employee -> call logs)**
    a. Actor: Employee
    b. Steps:
        i. Click a button to pull up the all call logs
        ii. Displays all call logs
        iii. Prompts user to select a date filter
        iv. Click on call ID to pull up the outcome
    c. SQL statement:
        i. SELECT e.emp_id, e.name, cl.emp_id, cl.name, cl.status, cl.date_time FROM Employee e, Call Log cl WHERE e.emp_id = cl.emp_id
    d. Use case realization:
        i. An employee must be able to review their own tickets. By using the emp_id as foreign key to the employee relation, we are able to show all the tickets that the employee has done.

38. **Query employees from departments (department -> employee)**
    a. Actor: Director
    b. Steps:
        i. User clicks a button to view all call centers
        ii. Prompts user to pick a call center
        iii. User selects a call center
        iv. Displays all employees in that call center
    c. SQL statement:
        i. SELECT cc.cc_id, cc.dept_id, d.dept_name, d.dept_id, e.cc_id, e.name, e.email FROM Call Center cc, Department d, Employee e WHERE d.dept_id = cc.dept_id AND cc.cc_id = e.cc_id
    d. Use case realization:
        i. By being able to label an employee to a call center and a call center to a department by using the department ID and call center ID, we can create a roster of employees with the respective departments and call centers along with their contact information. This would be used internally in the company by the Director.

39. **Query employees from call center (call center -> employee)**

   a. Actor: Manager
   b. Steps:
      i. Click a button that has the call center number
      ii. User is provided with contact information for the employees in that call center
   c. SQL statement:
      i. SELECT cc.cc_id, e.cc_id, e.name, e.email, e.emp_address, e.phone_number FROM Call Center c, Employee e WHERE cc.cc_id = e.cc_id
   d. Use case realization:
      i. Linking employees to a call center using the call center id will allow managers to oversee their employees and employee information

40. **Query call outcomes (call log -> outcome)**
   a. Actors: Managers, Employees
   b. Steps:
      i. User clicks on the call log
      ii. Prompts user to select a specific call
      iii. Displays the outcome of the call
   c. SQL statement:
      i. SELECT cl.outcome_id, cl.case_number, cl.employee_id, cl.date_time, o. * FROM Call Log cl, Outcome o, Where cl.outcome_id = o.outcome_id
   d. Use case realization:
      i. By linking the outcome to the call log outcome, it will allow for managers and employees to quickly glance at the call logs to see which employee services which ticket and what the outcome of the ticket was.

41. **Query call center in department (department -> call center)**
   a. Actors: Director, Manager, Employee
   b. Steps:
      i. Actor click button and is show all departments
      ii. Actor is then shown all call centers in said department and information
   c. SQL statement:
      i. SELECT cc.-*, d.dept_id FROM Department d, Call Center cc WHERE cc.dept_id = d.dept_id
   d. Use case realization:
      i. By joining Call Center and Department, we can categorize Call Center to departments

42. **Query problem tickets (call logs -> problem ticket)**
   a. Actors: Managers, Employees
   b. Steps:
      i. Actos clicks a button and is shown all calls that have a status that reflect that it is still open
      ii. Displays the details of the chat to the user.
   c. SQL statement:
      i. SELECT cl.case_number, cl.name, cl.emp_id, cl.issue_id, pt. * FROM Call Log cl, Problem Ticket pt, WHERE cl.issue_id = pt.issue_id
   d. Use case realization:
      i. Allows for technicians to quickly service customers by being able to link common problems to known common solutions

43. **Query customers from call log (call logs -> customer)—**shows all customer & removes multiple instances of repeat person/company
   a. Actors: Managers, Employees
   b. Steps:
      i. Actor clicks a button that shows all calls
      ii. Prompts user to select a service ticket
      iii. User then selects the customer's name
      iv. Displays the customer's contact information
   c. SQL statement:

i. SELECT cl. *, c.customer_id, c.name, c.email, c.phone_number FROM Call Log cl, Customer c WHERE cl.customer_id = c.customer_id

d. Use case realization:
    i. All service tickets should have customer_id which allows technicians to return a call, in the event of a disconnect, by looking up the customer's contact information.

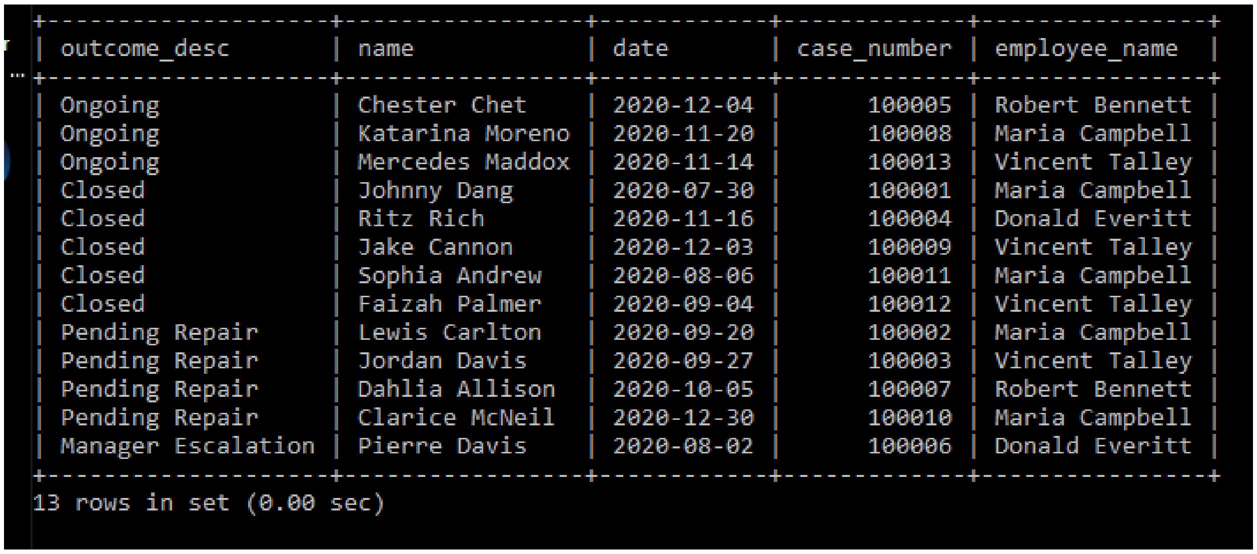
44. **Query all Call Center employee salaries in one of the Call Centers**
    a. Actor: Director
    b. Steps:
        i. User click a button that displays all call centers
        ii. Prompts the user to choose a call center
        iii. Displays all employees' salaries in the selected call center
        iv. Prompts the user to select a specific employee (optional)
        v. Displays employee's salary (optional)
    c. SQL statement:
        i. SELECT cc.cc_id, e.cc_id, e.name, e.salary FROM Call Center cc, Employee e WHERE cc.cc_id = e.cc_id
    d. Use case realization:
        i. By linking each employee to the designated call center directors, it will enable the directors to view the expenditures on employee salaries per employee or by each employee.

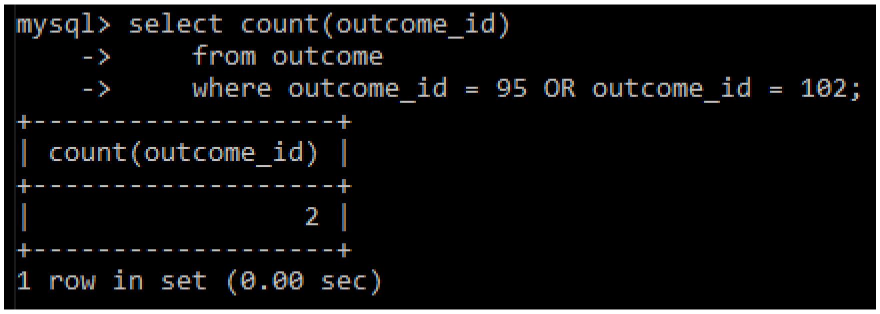
Major User Views

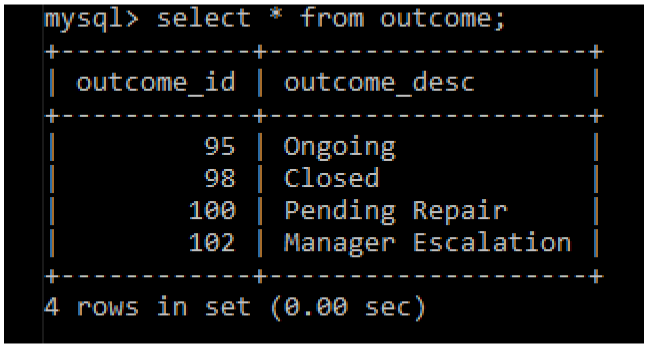
		DIRECTOR	MANAGER	SUPERVISOR	EMPLOYEE
ALL CUSTOMERS	MAINTAIN				
	QUERY	X	X		
	REPORT	X	X		
SINGLE CUSTOMER	MAINTAIN		X		
	QUERY	X	X	X	X
	REPORT	X	X	X	
ALL EMPLOYEES	MAINTAIN	X			
	QUERY	X	X		
	REPORT	X	X		
SINGLE EMPLOYEE	MAINTAIN		X		
	QUERY	X	X	X	X
	REPORT	X	X	X	
ALL CALL CENTERS	MAINTAIN	X			
	QUERY	X	X		
	REPORT	X	X		
SINGLE CALL CENTER	MAINTAIN		X		
	QUERY	X	X	X	X
	REPORT	X	X	X	
ALL DEPARTMENTS	MAINTAIN	X			
	QUERY	X	X		
	REPORT	X	X		
SINGLE DEPARTMENT	MAINTAIN		X		
	QUERY	X	X	X	X

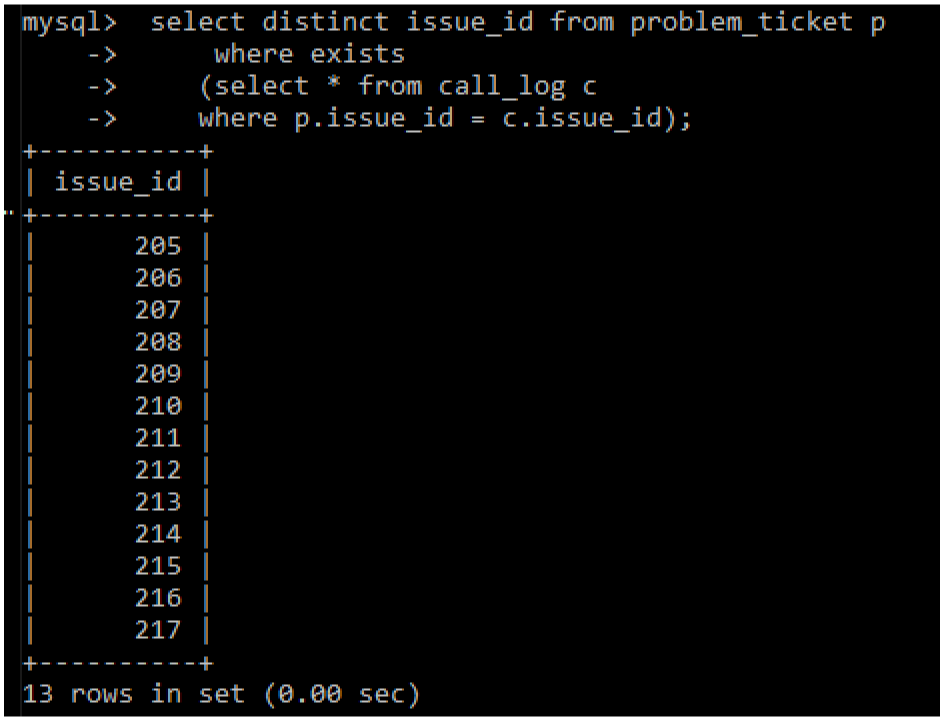
	REPORT	X	X	X	

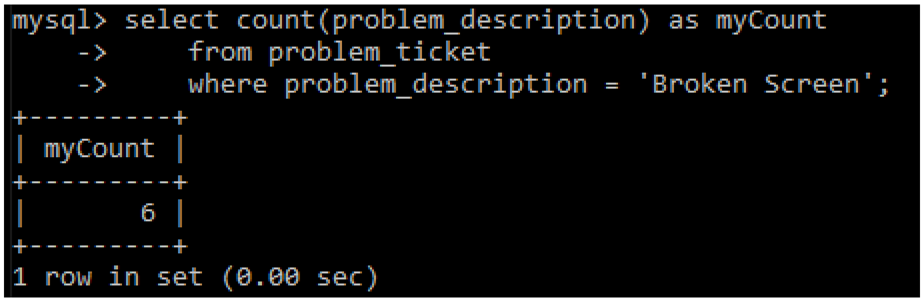
Test Cases

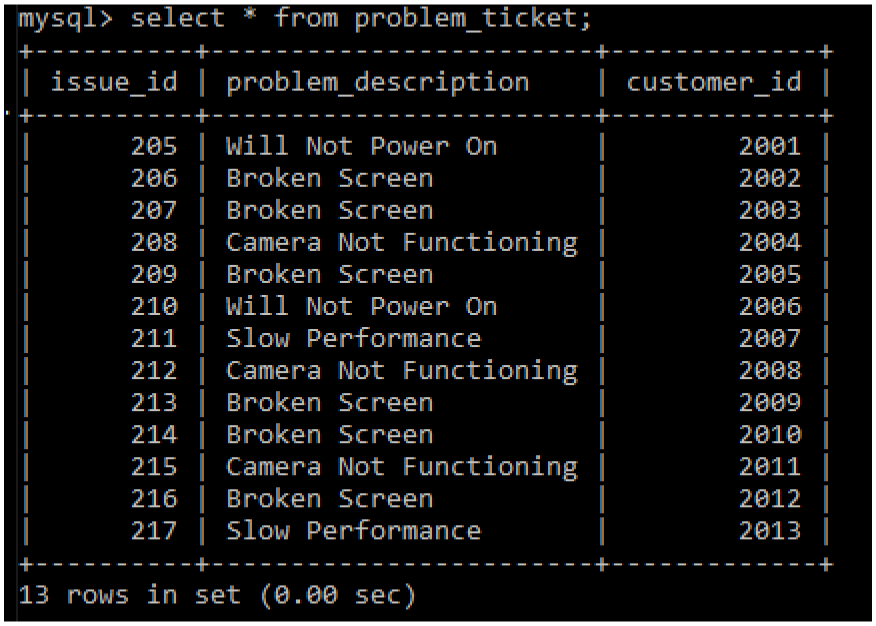

```
+--------------------+-----------------+------------+-------------+-----------------+
| outcome_desc       | name            | date       | case_number | employee_name   |
+--------------------+-----------------+------------+-------------+-----------------+
| Ongoing            | Chester Chet    | 2020-12-04 |      100005 | Robert Bennett  |
| Ongoing            | Katarina Moreno | 2020-11-20 |      100008 | Maria Campbell  |
| Ongoing            | Mercedes Maddox | 2020-11-14 |      100013 | Vincent Talley  |
| Closed             | Johnny Dang     | 2020-07-30 |      100001 | Maria Campbell  |
| Closed             | Ritz Rich       | 2020-11-16 |      100004 | Donald Everitt  |
| Closed             | Jake Cannon     | 2020-12-03 |      100009 | Vincent Talley  |
| Closed             | Sophia Andrew   | 2020-08-06 |      100011 | Maria Campbell  |
| Closed             | Faizah Palmer   | 2020-09-04 |      100012 | Vincent Talley  |
| Pending Repair     | Lewis Carlton   | 2020-09-20 |      100002 | Maria Campbell  |
| Pending Repair     | Jordan Davis    | 2020-09-27 |      100003 | Vincent Talley  |
| Pending Repair     | Dahlia Allison  | 2020-10-05 |      100007 | Robert Bennett  |
| Pending Repair     | Clarice McNeil  | 2020-12-30 |      100010 | Maria Campbell  |
| Manager Escalation | Pierre Davis    | 2020-08-02 |      100006 | Donald Everitt  |
+--------------------+-----------------+------------+-------------+-----------------+
13 rows in set (0.00 sec)
```

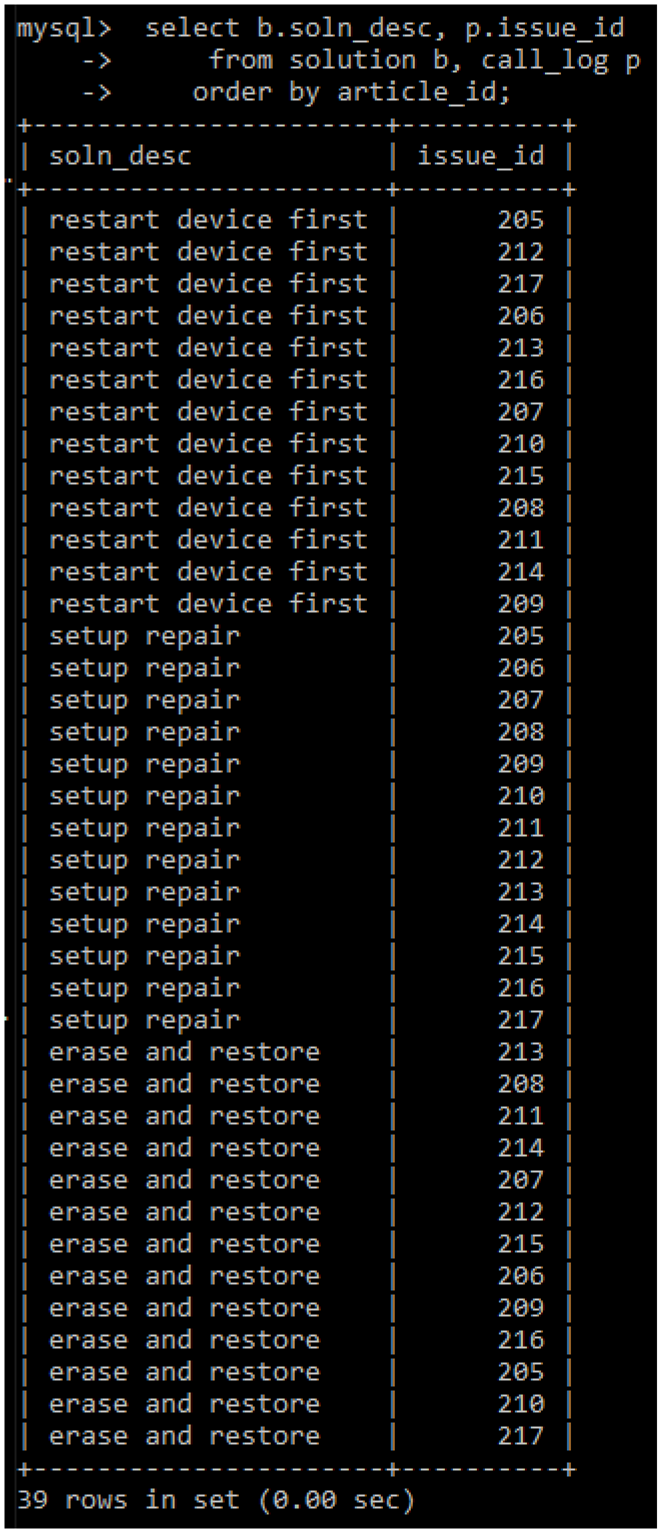

```
mysql> select count(outcome_id)
    ->     from outcome
    ->     where outcome_id = 95 OR outcome_id = 102;
+-------------------+
| count(outcome_id) |
+-------------------+
|                 2 |
+-------------------+
1 row in set (0.00 sec)
```


```
mysql> select * from outcome;
+------------+--------------------+
| outcome_id | outcome_desc       |
+------------+--------------------+
|         95 | Ongoing            |
|         98 | Closed             |
|        100 | Pending Repair     |
|        102 | Manager Escalation |
+------------+--------------------+
4 rows in set (0.00 sec)
```

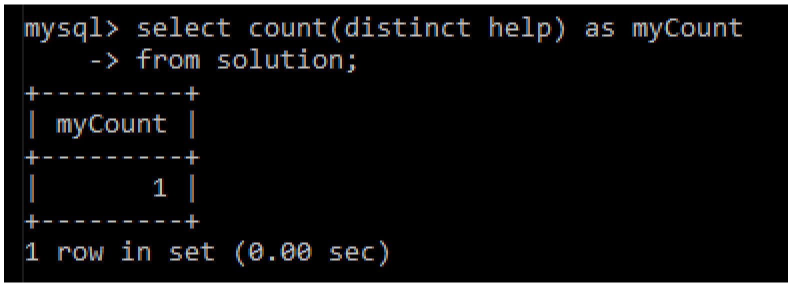

```
mysql>  select distinct issue_id from problem_ticket p
    ->      where exists
    ->     (select * from call_log c
    ->     where p.issue_id = c.issue_id);
+----------+
| issue_id |
+----------+
|      205 |
|      206 |
|      207 |
|      208 |
|      209 |
|      210 |
|      211 |
|      212 |
|      213 |
|      214 |
|      215 |
|      216 |
|      217 |
+----------+
13 rows in set (0.00 sec)
```


```
mysql> select count(problem_description) as myCount
    ->     from problem_ticket
    ->     where problem_description = 'Broken Screen';
+---------+
| myCount |
+---------+
|       6 |
+---------+
1 row in set (0.00 sec)
```

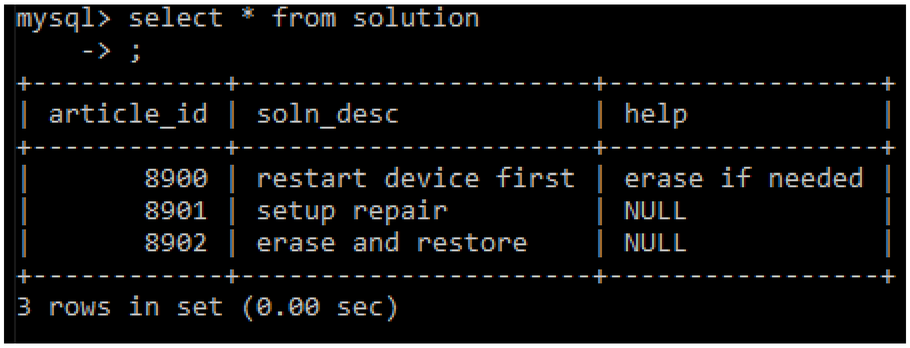

```
mysql> select * from problem_ticket;
+----------+-------------------------+-------------+
| issue_id | problem_description     | customer_id |
+----------+-------------------------+-------------+
|      205 | Will Not Power On       |        2001 |
|      206 | Broken Screen           |        2002 |
|      207 | Broken Screen           |        2003 |
|      208 | Camera Not Functioning  |        2004 |
|      209 | Broken Screen           |        2005 |
|      210 | Will Not Power On       |        2006 |
|      211 | Slow Performance        |        2007 |
|      212 | Camera Not Functioning  |        2008 |
|      213 | Broken Screen           |        2009 |
|      214 | Broken Screen           |        2010 |
|      215 | Camera Not Functioning  |        2011 |
|      216 | Broken Screen           |        2012 |
|      217 | Slow Performance        |        2013 |
+----------+-------------------------+-------------+
13 rows in set (0.00 sec)
```



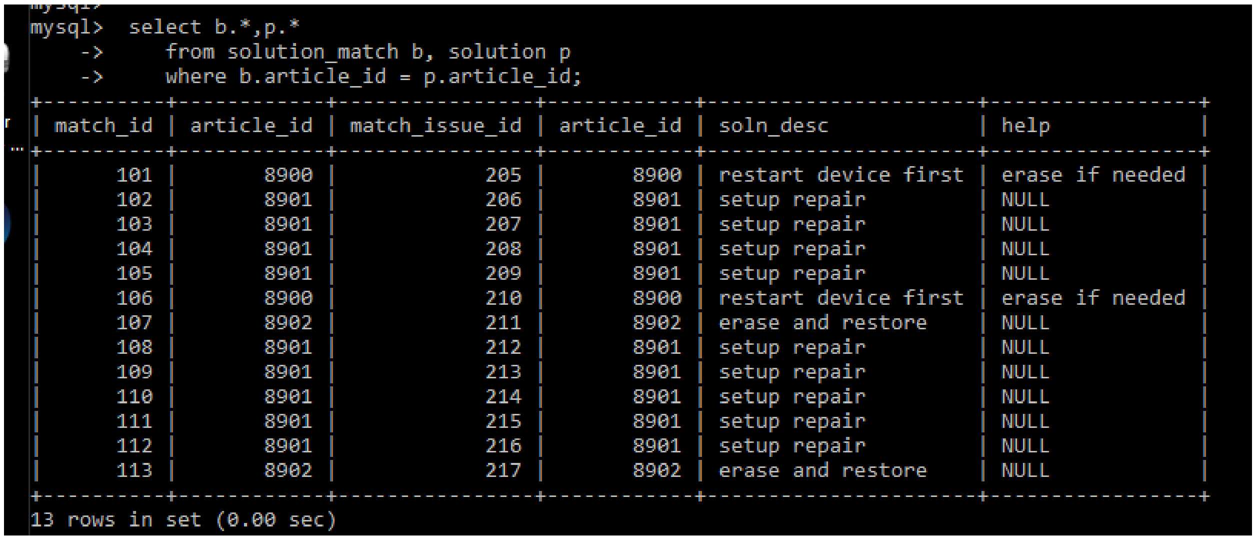
```
mysql>  select b.soln_desc, p.issue_id
    ->      from solution b, call_log p
    ->     order by article_id;
+----------------------+----------+
| soln_desc            | issue_id |
+----------------------+----------+
| restart device first |      205 |
| restart device first |      212 |
| restart device first |      217 |
| restart device first |      206 |
| restart device first |      213 |
| restart device first |      216 |
| restart device first |      207 |
| restart device first |      210 |
| restart device first |      215 |
| restart device first |      208 |
| restart device first |      211 |
| restart device first |      214 |
| restart device first |      209 |
| setup repair         |      205 |
| setup repair         |      206 |
| setup repair         |      207 |
| setup repair         |      208 |
| setup repair         |      209 |
| setup repair         |      210 |
| setup repair         |      211 |
| setup repair         |      212 |
| setup repair         |      213 |
| setup repair         |      214 |
| setup repair         |      215 |
| setup repair         |      216 |
| setup repair         |      217 |
| erase and restore    |      213 |
| erase and restore    |      208 |
| erase and restore    |      211 |
| erase and restore    |      214 |
| erase and restore    |      207 |
| erase and restore    |      212 |
| erase and restore    |      215 |
| erase and restore    |      206 |
| erase and restore    |      209 |
| erase and restore    |      216 |
| erase and restore    |      205 |
| erase and restore    |      210 |
| erase and restore    |      217 |
+----------------------+----------+
39 rows in set (0.00 sec)
```



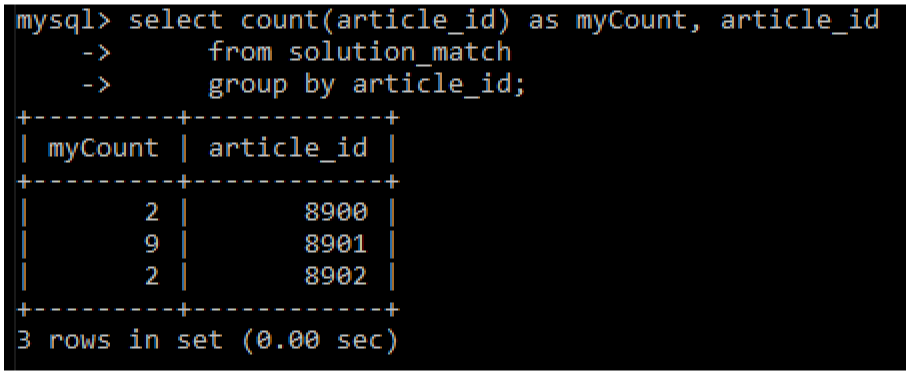
```
mysql> select count(distinct help) as myCount
    -> from solution;
+---------+
| myCount |
+---------+
|       1 |
+---------+
1 row in set (0.00 sec)
```



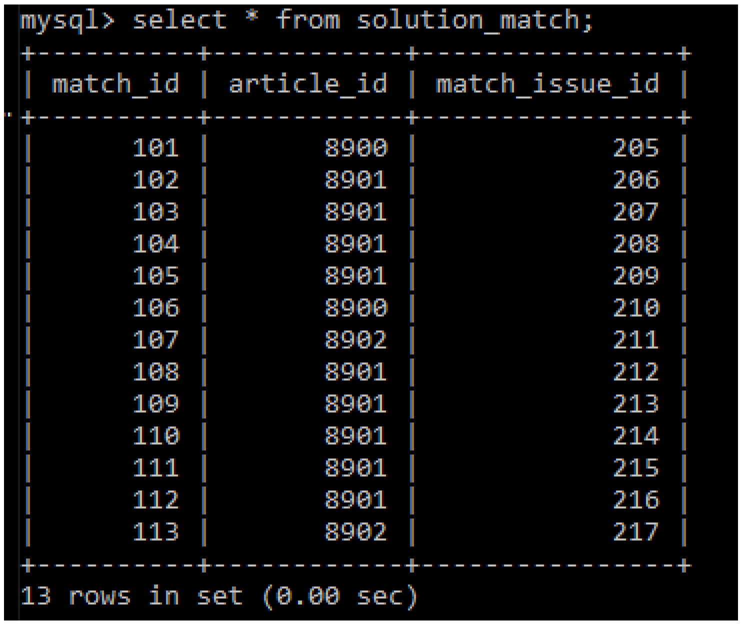
```
mysql> select * from solution
    -> ;
+------------+----------------------+-----------------+
| article_id | soln_desc            | help            |
+------------+----------------------+-----------------+
|       8900 | restart device first | erase if needed |
|       8901 | setup repair         | NULL            |
|       8902 | erase and restore    | NULL            |
+------------+----------------------+-----------------+
3 rows in set (0.00 sec)
```



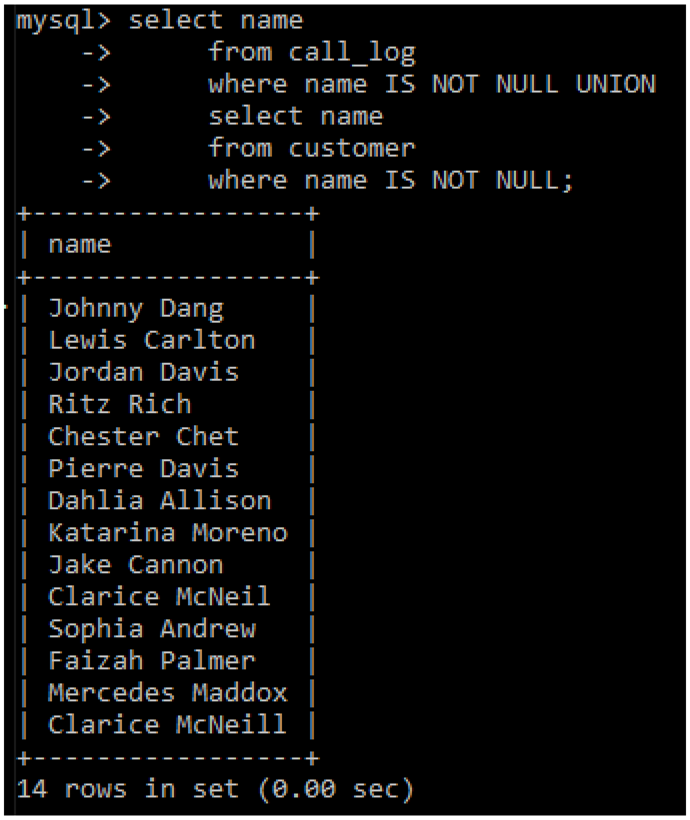
```
mysql>  select b.*,p.*
    ->     from solution_match b, solution p
    ->     where b.article_id = p.article_id;
+----------+------------+----------------+------------+----------------------+-----------------+
| match_id | article_id | match_issue_id | article_id | soln_desc            | help            |
+----------+------------+----------------+------------+----------------------+-----------------+
|      101 |       8900 |            205 |       8900 | restart device first | erase if needed |
|      102 |       8901 |            206 |       8901 | setup repair         | NULL            |
|      103 |       8901 |            207 |       8901 | setup repair         | NULL            |
|      104 |       8901 |            208 |       8901 | setup repair         | NULL            |
|      105 |       8901 |            209 |       8901 | setup repair         | NULL            |
|      106 |       8900 |            210 |       8900 | restart device first | erase if needed |
|      107 |       8902 |            211 |       8902 | erase and restore    | NULL            |
|      108 |       8901 |            212 |       8901 | setup repair         | NULL            |
|      109 |       8901 |            213 |       8901 | setup repair         | NULL            |
|      110 |       8901 |            214 |       8901 | setup repair         | NULL            |
|      111 |       8901 |            215 |       8901 | setup repair         | NULL            |
|      112 |       8901 |            216 |       8901 | setup repair         | NULL            |
|      113 |       8902 |            217 |       8902 | erase and restore    | NULL            |
+----------+------------+----------------+------------+----------------------+-----------------+
13 rows in set (0.00 sec)
```



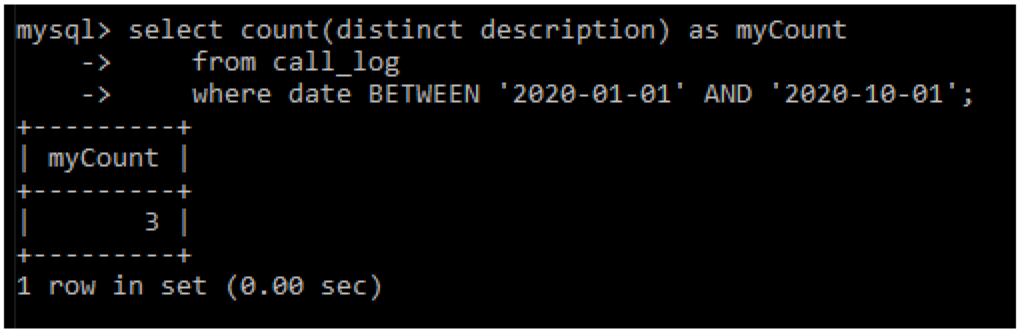
```
mysql> select count(article_id) as myCount, article_id
    ->     from solution_match
    ->     group by article_id;
+---------+------------+
| myCount | article_id |
+---------+------------+
|       2 |       8900 |
|       9 |       8901 |
|       2 |       8902 |
+---------+------------+
3 rows in set (0.00 sec)
```



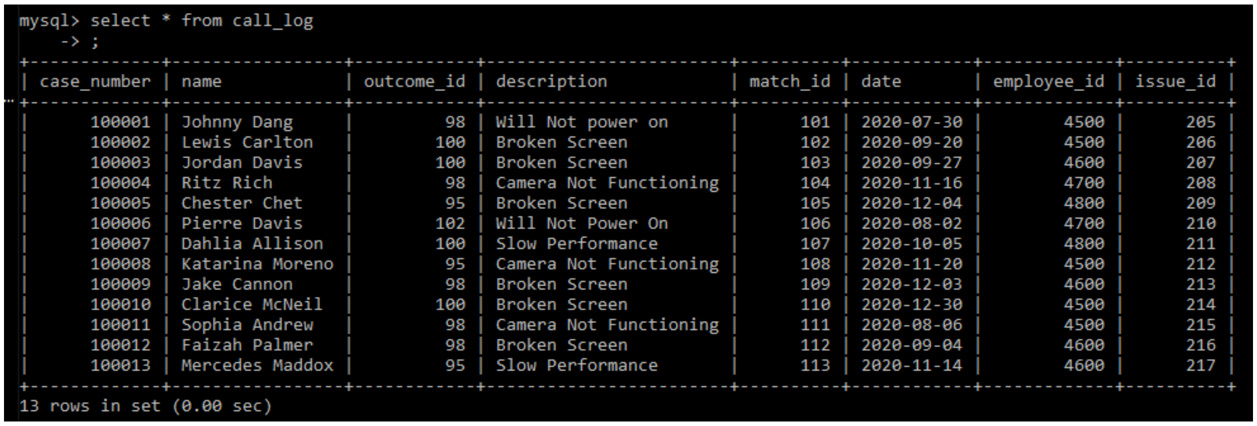
```
mysql> select * from solution_match;
+----------+------------+----------------+
| match_id | article_id | match_issue_id |
+----------+------------+----------------+
|      101 |       8900 |            205 |
|      102 |       8901 |            206 |
|      103 |       8901 |            207 |
|      104 |       8901 |            208 |
|      105 |       8901 |            209 |
|      106 |       8900 |            210 |
|      107 |       8902 |            211 |
|      108 |       8901 |            212 |
|      109 |       8901 |            213 |
|      110 |       8901 |            214 |
|      111 |       8901 |            215 |
|      112 |       8901 |            216 |
|      113 |       8902 |            217 |
+----------+------------+----------------+
13 rows in set (0.00 sec)
```



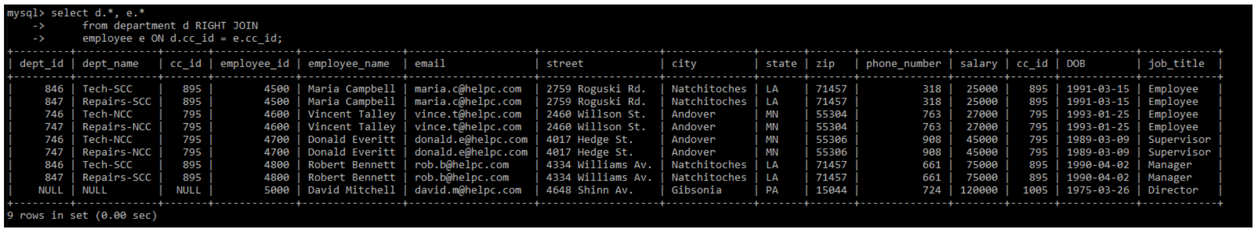
```
mysql> select name
    ->      from call_log
    ->      where name IS NOT NULL UNION
    ->      select name
    ->      from customer
    ->      where name IS NOT NULL;
+-----------------+
| name            |
+-----------------+
| Johnny Dang     |
| Lewis Carlton   |
| Jordan Davis    |
| Ritz Rich       |
| Chester Chet    |
| Pierre Davis    |
| Dahlia Allison  |
| Katarina Moreno |
| Jake Cannon     |
| Clarice McNeil  |
| Sophia Andrew   |
| Faizah Palmer   |
| Mercedes Maddox |
| Clarice McNeill |
+-----------------+
14 rows in set (0.00 sec)
```



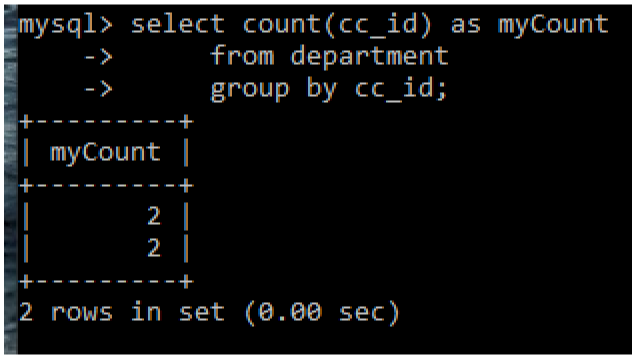
```
mysql> select count(distinct description) as myCount
    ->     from call_log
    ->     where date BETWEEN '2020-01-01' AND '2020-10-01';
+---------+
| myCount |
+---------+
|       3 |
+---------+
1 row in set (0.00 sec)
```



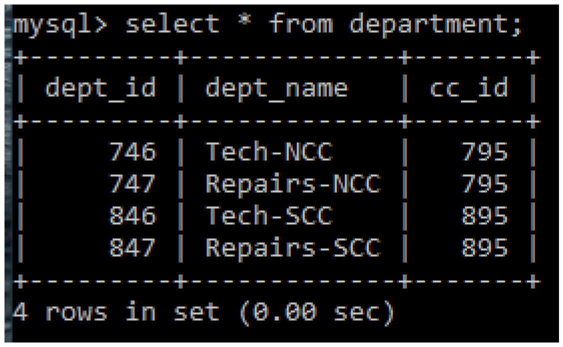
```
mysql> select * from call_log
    -> ;
+-------------+-----------------+------------+------------------------+----------+------------+-------------+----------+
| case_number | name            | outcome_id | description            | match_id | date       | employee_id | issue_id |
+-------------+-----------------+------------+------------------------+----------+------------+-------------+----------+
|      100001 | Johnny Dang     |         98 | Will Not power on      |      101 | 2020-07-30 |        4500 |      205 |
|      100002 | Lewis Carlton   |        100 | Broken Screen          |      102 | 2020-09-20 |        4500 |      206 |
|      100003 | Jordan Davis    |        100 | Broken Screen          |      103 | 2020-09-27 |        4600 |      207 |
|      100004 | Ritz Rich       |         98 | Camera Not Functioning |      104 | 2020-11-16 |        4700 |      208 |
|      100005 | Chester Chet    |         95 | Broken Screen          |      105 | 2020-12-04 |        4800 |      209 |
|      100006 | Pierre Davis    |        102 | Will Not Power On      |      106 | 2020-08-02 |        4700 |      210 |
|      100007 | Dahlia Allison  |        100 | Slow Performance       |      107 | 2020-10-05 |        4800 |      211 |
|      100008 | Katarina Moreno |         95 | Camera Not Functioning |      108 | 2020-11-20 |        4500 |      212 |
|      100009 | Jake Cannon     |         98 | Broken Screen          |      109 | 2020-12-03 |        4600 |      213 |
|      100010 | Clarice McNeil  |        100 | Broken Screen          |      110 | 2020-12-30 |        4500 |      214 |
|      100011 | Sophia Andrew   |         98 | Camera Not Functioning |      111 | 2020-08-06 |        4500 |      215 |
|      100012 | Faizah Palmer   |         98 | Broken Screen          |      112 | 2020-09-04 |        4600 |      216 |
|      100013 | Mercedes Maddox |         95 | Slow Performance       |      113 | 2020-11-14 |        4600 |      217 |
+-------------+-----------------+------------+------------------------+----------+------------+-------------+----------+
13 rows in set (0.00 sec)
```



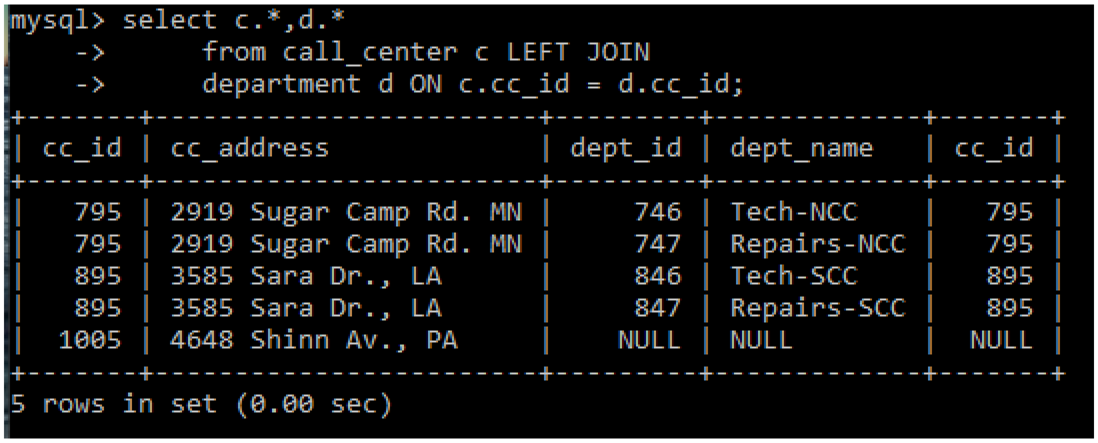
```
mysql> select d.*, e.*
    ->     from department d RIGHT JOIN
    ->     employee e ON d.cc_id = e.cc_id;
+---------+-------------+-------+-------------+----------------+--------------------+-------------------+--------------+-------+-------+--------------+--------+-------+------------+------------+
| dept_id | dept_name   | cc_id | employee_id | employee_name  | email              | street            | city         | state | zip   | phone_number | salary | cc_id | DOB        | job_title  |
+---------+-------------+-------+-------------+----------------+--------------------+-------------------+--------------+-------+-------+--------------+--------+-------+------------+------------+
|     846 | Tech-SCC    |   895 |        4500 | Maria Campbell | maria.c@helpc.com  | 2759 Roguski Rd.  | Natchitoches | LA    | 71457 |          318 |  25000 |   895 | 1991-03-15 | Employee   |
|     847 | Repairs-SCC |   895 |        4500 | Maria Campbell | maria.c@helpc.com  | 2759 Roguski Rd.  | Natchitoches | LA    | 71457 |          318 |  25000 |   895 | 1991-03-15 | Employee   |
|     746 | Tech-NCC    |   795 |        4600 | Vincent Talley | vince.t@helpc.com  | 2460 Willson St.  | Andover      | MN    | 55304 |          763 |  27000 |   795 | 1993-01-25 | Employee   |
|     747 | Repairs-NCC |   795 |        4600 | Vincent Talley | vince.t@helpc.com  | 2460 Willson St.  | Andover      | MN    | 55304 |          763 |  27000 |   795 | 1993-01-25 | Employee   |
|     746 | Tech-NCC    |   795 |        4700 | Donald Everitt | donald.e@helpc.com | 4017 Hedge St.    | Andover      | MN    | 55306 |          908 |  45000 |   795 | 1989-03-09 | Supervisor |
|     747 | Repairs-NCC |   795 |        4700 | Donald Everitt | donald.e@helpc.com | 4017 Hedge St.    | Andover      | MN    | 55306 |          908 |  45000 |   795 | 1989-03-09 | Supervisor |
|     846 | Tech-SCC    |   895 |        4800 | Robert Bennett | rob.b@helpc.com    | 4334 Williams Av. | Natchitoches | LA    | 71457 |          661 |  75000 |   895 | 1990-04-02 | Manager    |
|     847 | Repairs-SCC |   895 |        4800 | Robert Bennett | rob.b@helpc.com    | 4334 Williams Av. | Natchitoches | LA    | 71457 |          661 |  75000 |   895 | 1990-04-02 | Manager    |
|    NULL | NULL        |  NULL |        5000 | David Mitchell | david.m@helpc.com  | 4648 Shinn Av.    | Gibsonia     | PA    | 15044 |          724 | 120000 |  1005 | 1975-03-26 | Director   |
+---------+-------------+-------+-------------+----------------+--------------------+-------------------+--------------+-------+-------+--------------+--------+-------+------------+------------+
9 rows in set (0.00 sec)
```



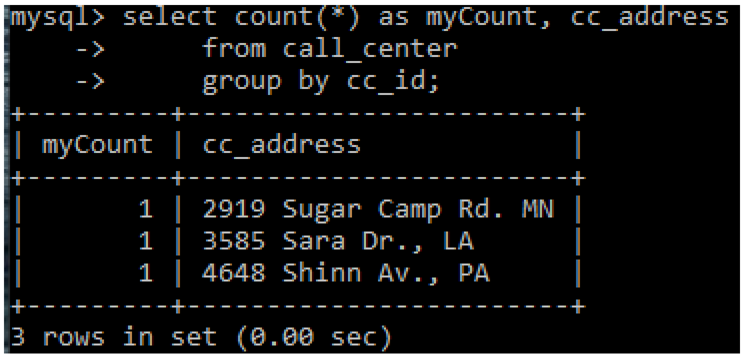
```
mysql> select count(cc_id) as myCount
    ->     from department
    ->     group by cc_id;
+---------+
| myCount |
+---------+
|       2 |
|       2 |
+---------+
2 rows in set (0.00 sec)
```



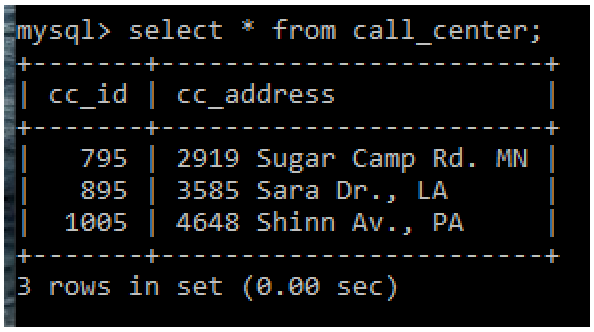
```
mysql> select * from department;
+---------+-------------+-------+
| dept_id | dept_name   | cc_id |
+---------+-------------+-------+
|     746 | Tech-NCC    |   795 |
|     747 | Repairs-NCC |   795 |
|     846 | Tech-SCC    |   895 |
|     847 | Repairs-SCC |   895 |
+---------+-------------+-------+
4 rows in set (0.00 sec)
```



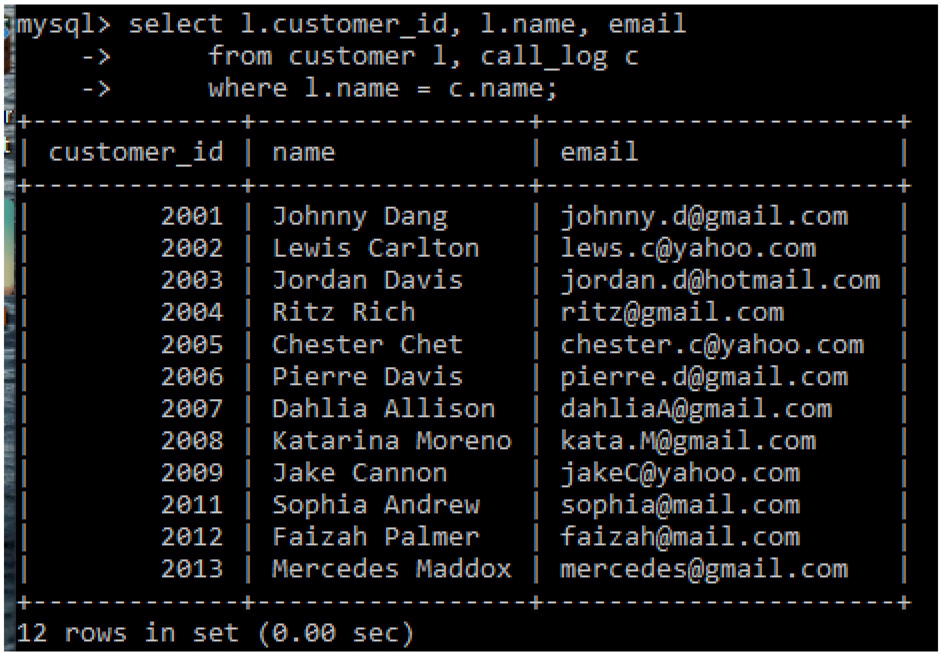
```
mysql> select c.*,d.*
    ->      from call_center c LEFT JOIN
    ->      department d ON c.cc_id = d.cc_id;
+-------+-----------------------+---------+-------------+-------+
| cc_id | cc_address            | dept_id | dept_name   | cc_id |
+-------+-----------------------+---------+-------------+-------+
|   795 | 2919 Sugar Camp Rd. MN |     746 | Tech-NCC    |   795 |
|   795 | 2919 Sugar Camp Rd. MN |     747 | Repairs-NCC |   795 |
|   895 | 3585 Sara Dr., LA     |     846 | Tech-SCC    |   895 |
|   895 | 3585 Sara Dr., LA     |     847 | Repairs-SCC |   895 |
|  1005 | 4648 Shinn Av., PA    |    NULL | NULL        |  NULL |
+-------+-----------------------+---------+-------------+-------+
5 rows in set (0.00 sec)
```



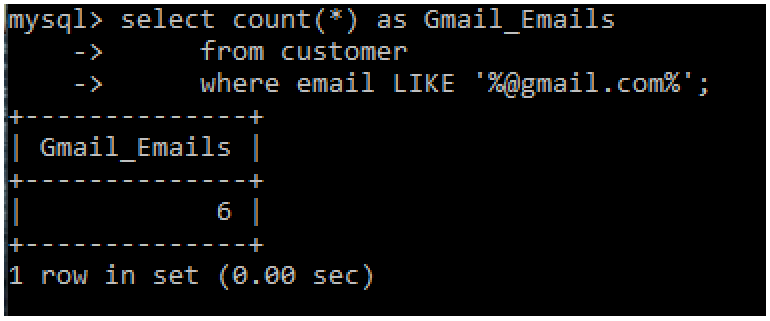
```
mysql> select count(*) as myCount, cc_address
    ->      from call_center
    ->      group by cc_id;
+---------+-----------------------+
| myCount | cc_address            |
+---------+-----------------------+
|       1 | 2919 Sugar Camp Rd. MN |
|       1 | 3585 Sara Dr., LA     |
|       1 | 4648 Shinn Av., PA    |
+---------+-----------------------+
3 rows in set (0.00 sec)
```



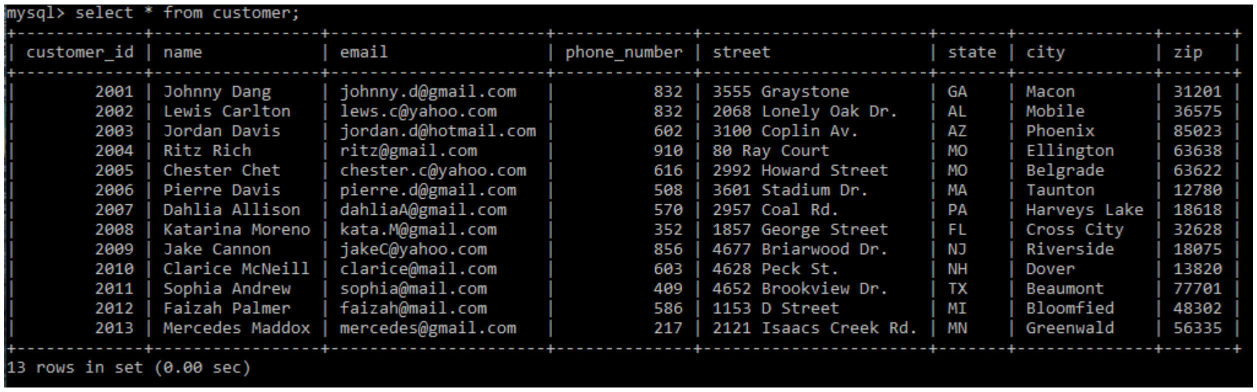
```
mysql> select * from call_center;
+-------+-----------------------+
| cc_id | cc_address            |
+-------+-----------------------+
|   795 | 2919 Sugar Camp Rd. MN |
|   895 | 3585 Sara Dr., LA     |
|  1005 | 4648 Shinn Av., PA    |
+-------+-----------------------+
3 rows in set (0.00 sec)
```



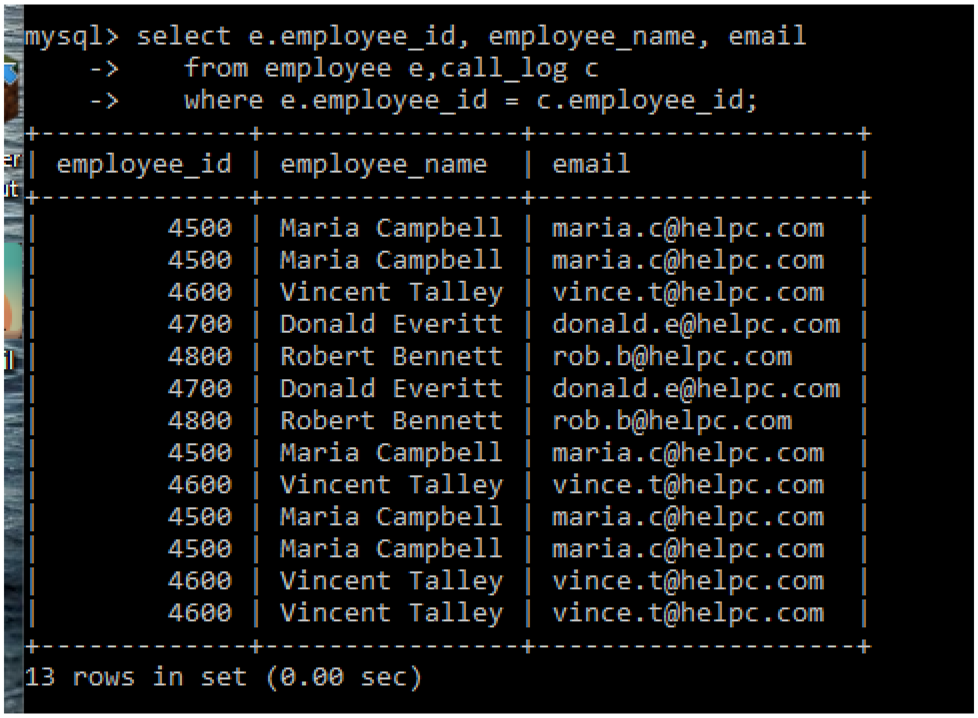
```
mysql> select l.customer_id, l.name, email
    ->      from customer l, call_log c
    ->      where l.name = c.name;
+-------------+-----------------+----------------------+
| customer_id | name            | email                |
+-------------+-----------------+----------------------+
|        2001 | Johnny Dang     | johnny.d@gmail.com   |
|        2002 | Lewis Carlton   | lews.c@yahoo.com     |
|        2003 | Jordan Davis    | jordan.d@hotmail.com |
|        2004 | Ritz Rich       | ritz@gmail.com       |
|        2005 | Chester Chet    | chester.c@yahoo.com  |
|        2006 | Pierre Davis    | pierre.d@gmail.com   |
|        2007 | Dahlia Allison  | dahliaA@gmail.com    |
|        2008 | Katarina Moreno | kata.M@gmail.com     |
|        2009 | Jake Cannon     | jakeC@yahoo.com      |
|        2011 | Sophia Andrew   | sophia@mail.com      |
|        2012 | Faizah Palmer   | faizah@mail.com      |
|        2013 | Mercedes Maddox | mercedes@gmail.com   |
+-------------+-----------------+----------------------+
12 rows in set (0.00 sec)
```



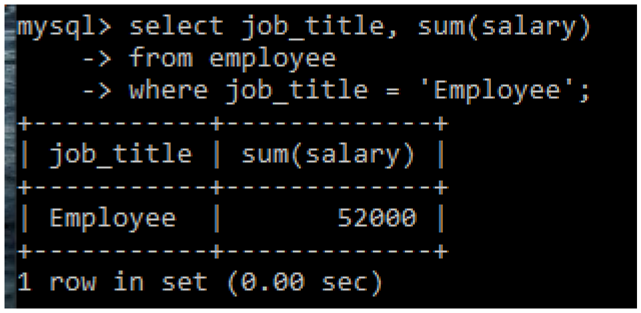
```
mysql> select count(*) as Gmail_Emails
    ->      from customer
    ->      where email LIKE '%@gmail.com%';
+--------------+
| Gmail_Emails |
+--------------+
|            6 |
+--------------+
1 row in set (0.00 sec)
```



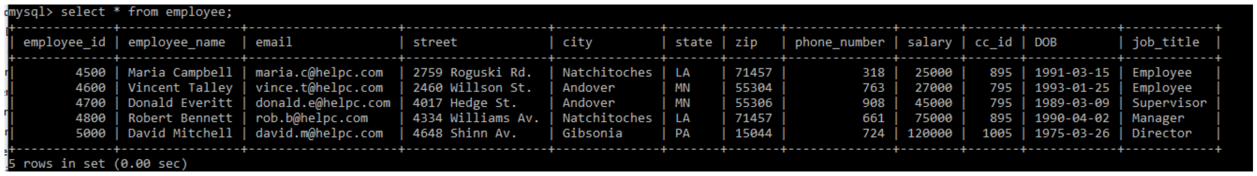
```
mysql> select * from customer;
+-------------+-----------------+----------------------+--------------+----------------------+-------+--------------+-------+
| customer_id | name            | email                | phone_number | street               | state | city         | zip   |
+-------------+-----------------+----------------------+--------------+----------------------+-------+--------------+-------+
|        2001 | Johnny Dang     | johnny.d@gmail.com   |          832 | 3555 Graystone       | GA    | Macon        | 31201 |
|        2002 | Lewis Carlton   | lews.c@yahoo.com     |          832 | 2068 Lonely Oak Dr.  | AL    | Mobile       | 36575 |
|        2003 | Jordan Davis    | jordan.d@hotmail.com |          602 | 3100 Coplin Av.      | AZ    | Phoenix      | 85023 |
|        2004 | Ritz Rich       | ritz@gmail.com       |          910 | 80 Ray Court         | MO    | Ellington    | 63638 |
|        2005 | Chester Chet    | chester.c@yahoo.com  |          616 | 2992 Howard Street   | MO    | Belgrade     | 63622 |
|        2006 | Pierre Davis    | pierre.d@gmail.com   |          508 | 3601 Stadium Dr.     | MA    | Taunton      | 12780 |
|        2007 | Dahlia Allison  | dahliaA@gmail.com    |          570 | 2957 Coal Rd.        | PA    | Harveys Lake | 18618 |
|        2008 | Katarina Moreno | kata.M@gmail.com     |          352 | 1857 George Street   | FL    | Cross City   | 32628 |
|        2009 | Jake Cannon     | jakeC@yahoo.com      |          856 | 4677 Briarwood Dr.   | NJ    | Riverside    | 18075 |
|        2010 | Clarice McNeill | clarice@mail.com     |          603 | 4628 Peck St.        | NH    | Dover        | 13820 |
|        2011 | Sophia Andrew   | sophia@mail.com      |          409 | 4652 Brookview Dr.   | TX    | Beaumont     | 77701 |
|        2012 | Faizah Palmer   | faizah@mail.com      |          586 | 1153 D Street        | MI    | Bloomfied    | 48302 |
|        2013 | Mercedes Maddox | mercedes@gmail.com   |          217 | 2121 Isaacs Creek Rd. | MN    | Greenwald    | 56335 |
+-------------+-----------------+----------------------+--------------+----------------------+-------+--------------+-------+
13 rows in set (0.00 sec)
```



```
mysql> select e.employee_id, employee_name, email
    ->    from employee e,call_log c
    ->    where e.employee_id = c.employee_id;
+-------------+-----------------+--------------------+
| employee_id | employee_name   | email              |
+-------------+-----------------+--------------------+
|        4500 | Maria Campbell  | maria.c@helpc.com  |
|        4500 | Maria Campbell  | maria.c@helpc.com  |
|        4600 | Vincent Talley  | vince.t@helpc.com  |
|        4700 | Donald Everitt  | donald.e@helpc.com |
|        4800 | Robert Bennett  | rob.b@helpc.com    |
|        4700 | Donald Everitt  | donald.e@helpc.com |
|        4800 | Robert Bennett  | rob.b@helpc.com    |
|        4500 | Maria Campbell  | maria.c@helpc.com  |
|        4600 | Vincent Talley  | vince.t@helpc.com  |
|        4500 | Maria Campbell  | maria.c@helpc.com  |
|        4500 | Maria Campbell  | maria.c@helpc.com  |
|        4600 | Vincent Talley  | vince.t@helpc.com  |
|        4600 | Vincent Talley  | vince.t@helpc.com  |
+-------------+-----------------+--------------------+
13 rows in set (0.00 sec)
```



```
mysql> select job_title, sum(salary)
    -> from employee
    -> where job_title = 'Employee';
+-----------+-------------+
| job_title | sum(salary) |
+-----------+-------------+
| Employee  |       52000 |
+-----------+-------------+
1 row in set (0.00 sec)
```



```
mysql> select * from employee;
+-------------+-----------------+---------------------+--------------------+--------------+-------+-------+--------------+--------+-------+------------+------------+
| employee_id | employee_name   | email               | street             | city         | state | zip   | phone_number | salary | cc_id | DOB        | job_title  |
+-------------+-----------------+---------------------+--------------------+--------------+-------+-------+--------------+--------+-------+------------+------------+
|        4500 | Maria Campbell  | maria.c@helpc.com   | 2759 Roguski Rd.   | Natchitoches | LA    | 71457 |          318 |  25000 |   895 | 1991-03-15 | Employee   |
|        4600 | Vincent Talley  | vince.t@helpc.com   | 2460 Willson St.   | Andover      | MN    | 55304 |          763 |  27000 |   795 | 1993-01-25 | Employee   |
|        4700 | Donald Everitt  | donald.e@helpc.com  | 4017 Hedge St.     | Andover      | MN    | 55306 |          908 |  45000 |   795 | 1989-03-09 | Supervisor |
|        4800 | Robert Bennett  | rob.b@helpc.com     | 4334 Williams Av.  | Natchitoches | LA    | 71457 |          661 |  75000 |   895 | 1990-04-02 | Manager    |
|        5000 | David Mitchell  | david.m@helpc.com   | 4648 Shinn Av.     | Gibsonia     | PA    | 15044 |          724 | 120000 |  1005 | 1975-03-26 | Director   |
+-------------+-----------------+---------------------+--------------------+--------------+-------+-------+--------------+--------+-------+------------+------------+
5 rows in set (0.00 sec)
```

Conclusion

By utilizing this database model a businesses that uses call centers to to service customers can be monumentally more successful. This database model will store all customer interactions to help improving future interactions and collect data that would help further the success of the business The database model also organize and consolidate employee data to existing and new call center branches to help view branch and department expenditures. By showing managers, directors and owners useful information they are able to interpret this data in an easy manor that will help solidify any decision to grow and increase profit margins. By collecting the customer call data we can see trends in customer call s that will help product manufactures make revisions to their product easy by using the date we collect and further solidifying business partner relation ships. The database model will greatly increase the efficiency of the employee which will increase the number of calls handled.

Reference

Connolly, T. M., and C. E. Begg. *Database Systems: a Practical Approach to Design, Implementation and Management*. Pearson Education, 2010.