Software used: Structured Query Language

**Functionalities:**

1. Checks volunteer age and whether the volunteer in current or not to segregate the volunteer based on activities.
2. Fetching volunteers based on the training ID to allocate involvement based on the training ID.
3. Identifying volunteers based on activity date and involvement type.
4. Retrieving data about volunteer based on ascending order of volunteer’s first name.
5. Retrieving volunteer data based on various criteria of performance in their activity

**STATEMENT OF OBJECTIVES**

|  |  |
| --- | --- |
| **Tables Created** | **Query Used** |
| INVOLVEMENT | DDL1 |
| TRAINING | DDL2 |
| VOLUNTEER | DDL3 |
| EMERGENCY\_CONTACT | DDL4 |
| ACTIVITY | DDL5 |

1. The club wants to differentiate the volunteers on the basis of their age because a few activities have a requirement of specific age criteria. So, we created a functionality where we would  display volunteers above the age of 18 years and then  differentiated them on whether they are current volunteers or not. Further, we would segregate them on the basis of their training Id because the club requires the volunteer to complete a level of requirements test in order for them to conduct a particular activity.

**(Query: DML1, DML3, DML4, DML8; Appendix 4,6,7,11)**

2. Volunteers could be identified on the basis of the training level that can help the club assign the volunteer to a specific activity. So, we created a functionality where the volunteer on the basis of their training level would be fetched and then the club can assign them to activity.

**(Query: DML14, Appendix 17)**

3. Volunteers could be identified based on the basis of the date of activity and the particular involvement type so the club could pay the volunteer as per the volunteer’s involved activities.

**(Query: DML12, Appendix15)**

4. Volunteer’s data including their email ID and enrollment date could be retrieved based on the volunteer’s first name in ascending order. This could lead to l faster and more efficient data retrieval.

**(Query: DML6, Appendix 9)**

5. The main function of the following queries will be to understand the activity pattern a volunteer has with respect to date, and the type of involvement s/he has participated. For example, query Q17 identifies the total hours spent by a volunteer each day regardless of the involvement, thus giving the daily hours spent with the club. Similarly, queries Q18 and Q19 have uses in understanding the minimum time spent by each volunteer across all the activities and knowing the involvement in which a volunteer had spent the maximum time.

**(Queries: DML17, DML18, DML19, Appendix 20,21,22)**

6. In order to assess the volunteers on their yearly performance, we have queries Q20 and Q21.To identify the average time spent in each involvement by each volunteer based on the year. This can give us an idea of where each volunteer is spending more time over a year, and how can we schedule them. Query 20 can give us the participation of volunteers in activities after the given yeas (in our case 2020), for those who have an involvement more than 2. This way it becomes more easier to get information of volunteers and can help in scheduling them according to the availability.

**(Queries: DML20, DML21; Appendix 23,24)**

7. A string-based identification of training IDs and Volunteers. For example, query Q can help us identifying training IDs when not knowing them and query Q will help us to pick the associates with specific range of them based on the beginning alphabets of First names and not any in a given range of alphabets for their last names.

**(Queries: DML 22, DML23, Appendix 25,26)**

8. Another functionality includes the identification of volunteers based on whether they have performed an activity or not. Query Q24 picks the distinct volunteers those have done an activity, while query Q25 goes a step ahead and display all the activity details of volunteers in volunteer table, incise they didn’t perform an activity their data regarding activity will display empty data. Queries like Q 27 will assess the number of actives a volunteer had done yearly and displays them to identify the overall yearly performance of the volunteers.

**(Queries: DML24, DML25, DML27, DML28, DML29 ; Appendix27,28,30,31,32)**

9. A training level details of volunteers and their emergency contacts can be retrieved to identify which volunteer undergoes which training in a sequential order and if no one is assigned under the training we can include more volunteers specific for that specific training.

**(Query: DML26, Appendix 29)**

**QUERIES**

**Data Definition Queries (DDL):**

**DDL1:** This query creates the Involvement table with attributes as InvolvementID and InvolvmentType, where primary key is InvolvementID.

/\*INVOLVEMNT TABLE\*/

CREATE TABLE INVOLVEMENT (

InvolvementID Int NOT NULL Identity(1,1),

InvolvmentType Char(30) NOT NULL,

CONSTRAINT INVOLVEMENT\_PK PRIMARY KEY(InvolvementID)

);

**DDL2:** This query will create the Training table with attributes as TrainingID and TrainingDescription, where primary key will be TrainingID.

/\* TRAINING TABLE\*/

CREATE TABLE TRAINING (

TrainingID Int not null identity(1,1),

TrainingDescription Char(30) not null unique,

CONSTRAINT TRAININIG\_PK PRIMARY KEY (TrainingID),

);

**DDL3:** This query will createVOLUNTEER table with attributes as VolunteerID, VolunteerFName, VolunteerLName, ContactNumber, Email, Birthdate, EnrollmentDate, TrainingID AND CurrentVolunteer, where primary key will be VolunteerID and foreign key will be TrainingID.

/\* VOLUNTEER TABLE \*/

CREATE TABLE VOLUNTEER (

VolunteerID Int NOT NULL identity(1,1),

VolunteerFName Char(30) NOT NULL,

VolunteerLName Char(30) NOT NULL,

ContactNumber Char(30) NOT NULL UNIQUE,

Email Varchar(30) NOT NULL UNIQUE,

Birthdate Date NOT NULL,

EnrollmentDate Date NOT NULL,

TrainingID Int NOT NULL DEFAULT 1,

CurrentVolunteer Char(30) NOT NULL DEFAULT 'Yes',

CONSTRAINT VOLUNTEER\_PK PRIMARY KEY(VolunteerID),

CONSTRAINT VOLUNTEER\_FK FOREIGN KEY(TrainingID)

REFERENCES TRAINING(TrainingID)

ON UPDATE CASCADE,

);

**DDL4:** This Query will create a Emergency Contact table with the attributes name as EmergencyID, ContactFNAME, ContactLNAME, ContactNumber, ContactRelationship, VolunteerID, where primary EmergencyID and volunteer ID will be the foreign key.

/\*EMERGENCY CONTACT TABLE\*/

CREATE TABLE EMERGENCY\_CONTACT (

EmergencyID Int identity(1,1),

ContactFNAME Char(30) NOT NULL,

ContactLNAME Char(30) NOT NULL,

ContactNumber Char(30) NOT NULL UNIQUE,

ContactRelationship Char(30) NOT NULL,

VolunteerID Int NOT NULL,

CONSTRAINT EMERGENCY\_PK PRIMARY KEY(EmergencyID),

CONSTRAINT EMERGENCY\_FK FOREIGN KEY(VolunteerID)

REFERENCES VOLUNTEER(VolunteerID)

ON DELETE CASCADE

ON UPDATE CASCADE

);

**DDL5:** This query will create Activity table with attributes names as ActivityID , VolunteerID, InvolvmentID, ActivityDate, InvolvementTimeIn and InvolvementTimeOut , where Primary key is ActivityID and foreign key is VolunteerID.

/\*ACTIVITY TABLE TO TRACT THE DAILY ACTIVITIES\*/

CREATE TABLE ACTIVITY (

ActivityID Int NOT NULL identity(1,1),

VolunteerID Int NOT NULL,

InvolvmentID Int NOT NULL,

ActivityDate Date NOT NULL DEFAULT GETDATE(),

InvolvementTimeIn Time NOT NULL,

InvolvementTimeOut Time NOT NULL,

CONSTRAINT ACTIVITY\_PK PRIMARY KEY (ActivityID),

CONSTRAINT ACTIVITY\_FK FOREIGN KEY(VolunteerID)

REFERENCES VOLUNTEER(VolunteerID),

CONSTRAINT ACTIVITY1\_FK FOREIGN KEY(InvolvmentID)

REFERENCES INVOLVEMENT(InvolvementID)

);

**DDL6:** This query will add the coluns o fEmergency contact tabe to the coulnteer table.

/\* ADDING THE COLUMNS OF EMERGENCY CONTACT TABLE TO MAKE IT SIMPLE WHILE RETRIVING THE OUTPUT DATA \*/

ALTER TABLE VOLUNTEER

ADD

ECFirstName Char(30) NOT NULL,

ECLastName Char(30) NOT NULL,

ECContactNumber Char(30) NOT NULL,

ECRelationship Char(30) NOT NULL;

**DDL7:** This query will add a unique constraint ContactUnique toECContactNumber

/\*ADDS UNIQUE CONSTRAINT TO CONTACT NUMBER\*/

ALTER TABLE VOLUNTEER

ADD CONSTRAINT ContactUnique UNIQUE (ECContactNumber);

**DDL8:** This query will drop the Emergency contact table on execution

/\* SINCE THE COLUMNS IN EMERGENCY TABLE ARE ADDED TO THE VOLUNTEER TABLE, EMERGENCY CONTACT TABLE IS DROPPED\*/

DROP TABLE EMERGENCY\_CONTACT;

A screenshot of a computer

Description automatically generated

**REFERENTIAL INTEGRITY:**

ON UPDATE CASCADE:

On update cascade referential integrity is incorporated in Volunteer Table with respect to the training ID in the Training Table. If a record in training table is updated, the corresponding values in the volunteer table will be updated. Similarly, if a record in the Emergency Contact table is updated, the corresponding values in the Volunteer table will also be updated (but the Emergency Contact table is dropped due to difficulties in retrieval of data)

ON DELETE CASCADE:

On delete cascade referential integrity is incorporated in the Emergency Contact table with respect to the volunteer ID in the Volunteer Table. If a record is deleted in the volunteer table, the corresponding record will be deleted in the Emergency Contact Table.

**VOLUNTEER TABLE**

**DML1:** This query will display VolunteerID, VolunteerFName,VolunteerLName, ContactNumber, Email, Birthdate, EnrollmentDate, TrainingID, CurrentVolunteer of the current volunteer.

/\*DISPLAYS VOLUNTEER INFO IF CURRENT VOLUNTEER (=) \*/

SELECT \* FROM VOLUNTEER

WHERE CurrentVolunteer = 'Yes';

**DML2:** This query will display the VolunterFname, VolunteerLName, ContcatNumber from Volunteer table for only those volunteers whose TrainingId= 2 or who is a current volunteer.

/\*DISPLAYS INFO OF VOLUNTEER USING OR FUNCTIONALITY (OR)\*/

SELECT VolunteerFName, VolunteerLName, ContactNumber

FROM VOLUNTEER

WHERE TrainingID = ' 2' OR  CurrentVolunteer= 'Yes';

**DML3:** This query will give us the result of VolunteerFName, VolunteerLName, ContactNumber from Volunteer table for those volunteer who are with the trainingid 1, 2 and 3.

/\*DISPLAYS VOLUNTEER INFO FOR TRAINING IDS 1,2 AND 3 (IN)\*/

SELECT VolunteerFName, VolunteerLName, ContactNumber

FROM VOLUNTEER

WHERE TrainingID IN ('1', '2', '3');

**DML4:** This query will give us the result of VolunteerFName, VolunteerLName, ContactNumber from Volunteer table for those volunteer who are with the  trainingid 4 and 5.

/\*DISPLAYS VOLUNTEER INFO FOR TRAINING IDS 4 AND 5 (NOT IN)\*/

SELECT VolunteerFName, VolunteerLName, ContactNumber

FROM VOLUNTEER

WHERE TrainingID NOT IN ('1', '2', '3');

**DML5:** This query will give us the result of VolunteerFName, VolunteerLName, ContactNumber from Volunteer table for those volunteer whose emailid is not null.

/\*DISPLAYS VOLUNTEER INFO IF EMAIL IS GIVEN (IS NOT NULL)\*/

SELECT VolunteerFName, VolunteerLName, ContactNumber

FROM VOLUNTEER

WHERE Email IS NOT NULL;

**DML6:** This query will provide us with  the result of the following fields by  storing it in ascending order:  VolunteerFName, VolunteerLName, Email and EnrollmentDate from Volunteer table

/\*DISPLAY NAME IN ASCENDING ORDER WITH EMAIL AND ENROLLMENT DATE (ORDER BY ASC)\*/

SELECT VolunteerFName, VolunteerLName, Email, EnrollmentDate

FROM VOLUNTEER

ORDER BY VolunteerFName ASC;

**DML7:** This query will provide us the total of number of volunteers from the Volunteer table under the column name NumberOfVolunteers

/\*DISPLAYS THE COUNT OF VOLUNTEERS (COUNT)\*/

SELECT COUNT(\*) AS NumberOfVolunteers

FROM VOLUNTEER;

**DML8:** This query will provide us with the output of  VolunteerId and their respective age (under the column name Age) from the Volunteer table.

/\*CALCULATE AGE\*/

SELECT

VolunteerID

, YEAR (Birthdate) AS [BirthYear]

,YEAR (GETDATE()) AS [ThisYear]

, YEAR (GETDATE () )-YEAR(Birthdate) AS [Age]

FROM

VOLUNTEER;

**TRAINING TABLE**

**DML9:** This query will provide with TrainingID, TrainingDescription of those volunteer whose TrainingId are 4 and 5 from the Training table.

/\*DISPLAY TRAINING INFORMATION WHOSE TRAINING ID IS GREATER THAN 3 (>)\*/

Select \* From TRAINING

Where TrainingID > 3;

**INVOLVEMENT TABLE:**

**DML10:** This query will provide us  with InvolvementID, InvolvementType of  those volunteers whose InvolvementID are 3,4,5,6 and 7 from the Involvement  table.

/\*DISPLAY INVOLVEMENT INFORMATION BASED ON INVOLVEMENT TYPE (<=)\*/

SELECT \*

FROM INVOLVEMENT

WHERE InvolvementID <= 2 ;

**DML11:** This query will provide us with the list of distinct Involvement types from the InvolvementType.

/\*DISPLAY DISTINCT INVOLVEMENT TYPES\*/

SELECT DISTINCT InvolvementType

FROM INVOLVEMENT;

**ACTIVITY TABLE:**

**DML12:** This query will provide all the information about a volunteer’s activities for activity date 2019-04-05 and Involvement ID = 1.

/\*DISPLAY THE ACTIVITY INFORMATION IF  (AND )\*/

SELECT \*

FROM ACTIVITY

WHERE ActivityDate = '2019-04-05 ' AND InvolvmentID = '1';

**DML13:** This query will display ActivityID, VolunteerID, InvolvmentID, ActivityDate of all the Volunteer on the basis of their activity date.

/\*DISPLAYS THE IDS BASED ON THE ORDER OF ACTIVITY DATE (ORDER BY)\*/

SELECT ActivityID, VolunteerID, InvolvmentID, ActivityDate

FROM ACTIVITY

ORDER BY ActivityDate;

**COMPLEX**

**DML14:** This query has performed an inner join between the following tables Volunteer table and Activity table , Activity table and Involvement table. It will display the VolunteerFName, VolunteerLName, Email, InvolvementID and InvolvementType from the Volunteer and Involvement table.

/\*DISPLAYS THE INFO USING INNER JOIN (INNER JOIN) \*/

SELECT V.VolunteerID, VolunteerFName, VolunteerLName, Email, InvolvementID, InvolvmentType

FROM VOLUNTEER AS V

   INNER JOIN

   ACTIVITY AS A

   ON V.VolunteerID = A.VolunteerID

   INNER JOIN

   INVOLVEMENT AS I

   ON A.InvolvmentID = I.InvolvementID

**DML15:** This query has performed an inner join between the following tables Volunteer table and Activity table , Activity table and Involvement table.It  provides information about Volunteer ID, Volunteer’s first and last name and email ID for those employees whose volunteer ID is between 2 and 5.

/\*DISPLAYS INFO BETWEEN INVOLVEMENT ID 2 AND 5 (INNER JOIN, WHERE BETWEEN)\*/

SELECT V.VolunteerID, VolunteerFName, VolunteerLName, Email

FROM VOLUNTEER AS V

   INNER JOIN

   ACTIVITY AS A

   ON V.VolunteerID = A.VolunteerID

   INNER JOIN

   INVOLVEMENT AS I

   ON A.InvolvmentID = I.InvolvementID

   WHERE I.InvolvementID BETWEEN 2 AND 5;

**DML16:** This query has performed a left join between the Volunteer and Training table. It will display the VolunteerID, TrainingID, VolunteerFName, VolunteerLName, Email, ContactNumber from the Volunteer table and respectively will display a null value in the TrainingId column if there is no match between Volunteer and Training table.

/\*DISPLAYS INFORMATION ABOUT VolunteerID, TrainingID, VolunteerFName, VolunteerLName, Email, ContactNumber The LEFT JOIN keyword returns all records from the VOLUNTEER table, and the matched records from the TRAINING TABLE. The result is NULL from the TRAINING side, if there is no match. (LEFT JOIN) \*/

SELECT VolunteerID, TRAINING.TrainingID, VolunteerFName, VolunteerLName, Email, ContactNumber

FROM VOLUNTEER LEFT JOIN TRAINING

ON VOLUNTEER.TrainingID = TRAINING.TrainingID;

**DML17:** The function of this query is to fetch the total of time a volunteer has spent on at a particular day

/\*SUM AND DATE DIFFERENCE FUNCTION\*/

SELECT VolunteerID, ActivityDate , SUM(DATEDIFF(HOUR, involvementTimeIn,InvolvementTimeOut)) AS "TOTAL HOURS SPENT IN A DAY "

FROM ACTIVITY

GROUP BY VolunteerID, ActivityDate;

**DML18:** The query will display the the hours spent in involevemnt in which a valounteer has worked for minimum time

/\*MIN FUNCTION\*/

SELECT VolunteerID, InvolvmentID , MIN(DATEDIFF(HOUR, InvolvementTimeIn,InvolvementTimeOut)) AS "MINIMUM HOURS SPENT IN AN INVOLVEMENT "

FROM ACTIVITY

GROUP BY VolunteerID, InvolvmentID;

**DML19:** This query will display the maximum number of hours worked by the volunteer in an involvement on a day.

/\*MAX FUNCTION\*/

SELECT VolunteerID, InvolvmentID, ActivityDate , MAX(DATEDIFF(HOUR,InvolvementTimeOut, InvolvementTimeIn)) AS "MAXIMUM HOURS SPENT IN AN INVOLVEMENT IN A DAY"

FROM ACTIVITY

GROUP BY VolunteerID, InvolvmentID, ActivityDate;

**DML20:** This query will display Volunteer First name, Volunteer Last name and contact number of the volunteer who has worked on and after 1st January 2020 and has activity id equal to or more than 2.

/\*NESTED QUERY\*/

SELECT VolunteerFName,VolunteerLName, ContactNumber

FROM VOLUNTEER

WHERE VolunteerID IN

(

SELECT VolunteerID

FROM ACTIVITY

WHERE InvolvmentID >=2 AND ActivityDate >='01-01-2020'

)

;

**DML21:** This query will display the average time spent by a volunteer across all the activities on a given day

/\*AVERAGE FUNCTION\*/

SELECT VolunteerID, YEAR(ActivityDate) YEAR , InvolvmentID, AVG(DATEDIFF(HOUR,InvolvementTimeIn,InvolvementTimeOut)) AS "AVERAGE HOURS SPENT IN A YEAR "

FROM ACTIVITY

GROUP BY VolunteerID, YEAR(ActivityDate) , InvolvmentID

ORDER BY YEAR(ActivityDate)

;

**DML22:** This query will first retrieve the training id where the training description end with ‘Instructor’ and does not start with ‘R’, then retrieves the volunteers form the volunteer table with the training id that was retrieved from the inner query.

/\*WILDCARD LIKE AND NOT LIKE FUNCTION\*/

SELECT VolunteerID, VolunteerFName, VolunteerLName , TrainingID

FROM VOLUNTEER

WHERE TrainingID IN

(

SELECT TrainingID

FROM TRAINING

WHERE TrainingDescription LIKE '%Instructor' AND TrainingDescription NOT LIKE 'R%'

);

**DML23:** Indentify all the volunteers whose First name starts from A-C and whose last name does not start from w-z.

/\*WILDCARD LIKE AND NOT LIKE FUNCTION \*/

SELECT VolunteerID, VolunteerFName, VolunteerLName

FROM VOLUNTEER

WHERE VolunteerFName LIKE '[A-C]%'

AND VolunteerLName NOT LIKE '[W-Z]%'

;

**DML24:** Displays emergency contacts of all volunteers who have participated in an activity.

/\*DISTINCT FUNCTION\*/

SELECT ECFirstName, ECLastName, ECContactNumber

FROM

VOLUNTEER

WHERE VolunteerID IN

(SELECT DISTINCT VolunteerID

FROM

ACTIVITY

)

;

**DML25:** This displays the data of volunteers, regardless of whether they have an activity or not, if they have performed an activity, then their details on the activity will be displayed, if not their activity data will be null

/\*LEFT JOIN\*/

SELECT VolunteerFName , VolunteerLName, ECFirstName, ECLastName, ECContactNumber, InvolvmentID, ActivityDate, InvolvementTimeIn, InvolvementTimeOut FROM

VOLUNTEER AS V LEFT JOIN ACTIVITY AS A ON A.VolunteerID = V.VolunteerID

;

**DML26:** This query will display all the training IDs along with the data of the volunteer who are under that training id, if no volunteer is under a training id then NULL will be displayed in the volunteer info

/\*RIGHT JOIN\*/

SELECT T.TrainingID, VolunteerFName , VolunteerLName, ECFirstName, ECLastName, ECContactNumber FROM

VOLUNTEER V RIGHT JOIN TRAINING T ON V.TrainingID= T.TrainingID

;

**DML27:** Displays the number of activity volunteer had participated in each day

/\*HAVING FUNCTION WITH YEAR BUILT IN FUNCTION\*/

SELECT VOLUNTEERID, YEAR(ACTIVITYDATE) "ACTIVITY YEAR", COUNT(ACTIVITYID) "NO. OF ACTIVITIES EACH YEAR"

FROM ACTIVITY

GROUP BY VolunteerID,YEAR(ActivityDate)

HAVING COUNT(ACTIVITYID) >=2

;

**DML28:** If a volunteer has performed an activity his data will be displayed

/\*ASC AND DESC FUNCTION\*/

SELECT V.VOLUNTEERID,A.InvolvmentID, VolunteerFName, VolunteerLName

FROM ACTIVITY A, VOLUNTEER V

WHERE A.VolunteerID= V.VolunteerID

ORDER BY InvolvmentID DESC, VolunteerLName ASC

;

**DML29:** The query will display the Volunteer information if he has performed any activity.

/\*EXISTS FUNCTION\*/

SELECT VolunteerID, VolunteerFName, VolunteerLName

FROM VOLUNTEER V

WHERE

EXISTS

(

SELECT COUNT(\*)

FROM ACTIVITY A

WHERE A.VolunteerID = V.VolunteerID

GROUP BY VolunteerID

)

ORDER BY VolunteerID

;

**Appendix AND SAMPLE DATA**

**Appendix 1:**

A screenshot of a computer

Description automatically generated

**Appendix 2:**

**A screenshot of a computer

Description automatically generated**

**Appendix 3:**

**A screenshot of a computer

Description automatically generated**

**Appendix 4:**

**A screenshot of a computer

Description automatically generated**

**Appendix 5:**

**A screenshot of a cell phone

Description automatically generated**

**Appendix 6:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 7:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 8:**

**A screenshot of a cell phone

Description automatically generated**

**Appendix 9:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 10:**

**A screenshot of a cell phone

Description automatically generated**

**Appendix 11:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 12:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 13:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 14:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 15:**

**A screenshot of a cell phone

Description automatically generated**

**Appendix 16:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 17:**

**A screenshot of a social media post

Description automatically generated**

**Appendix 18:**

**A screenshot of a cell phone

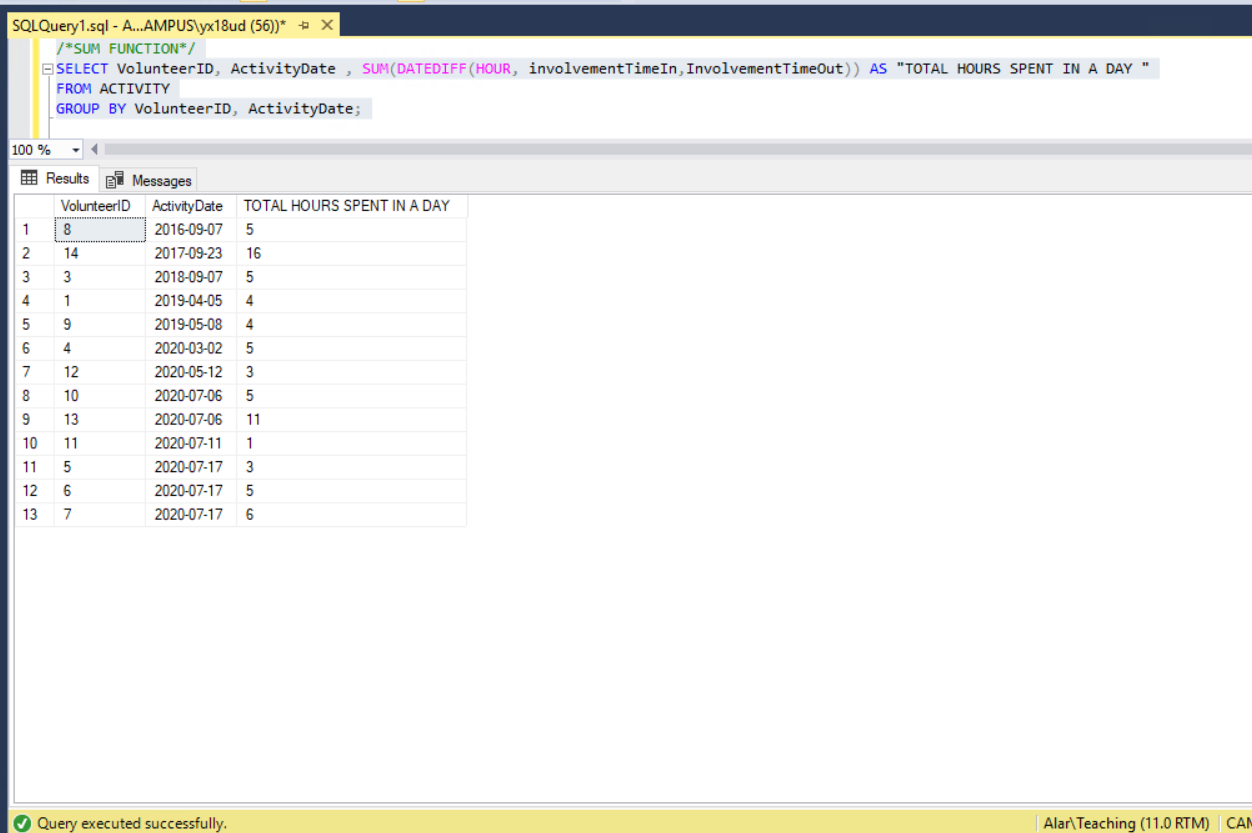
Description automatically generated**

**Appendix 19:**

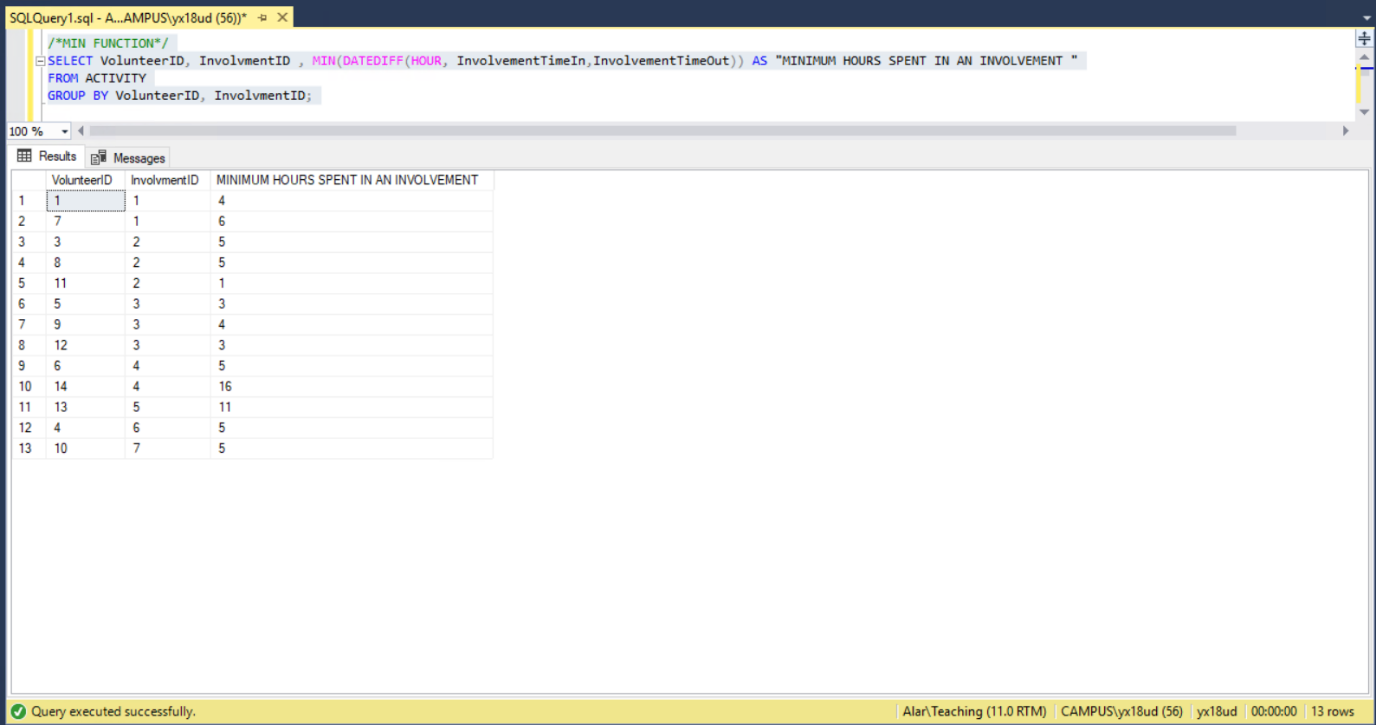
**A screenshot of a social media post

Description automatically generated**

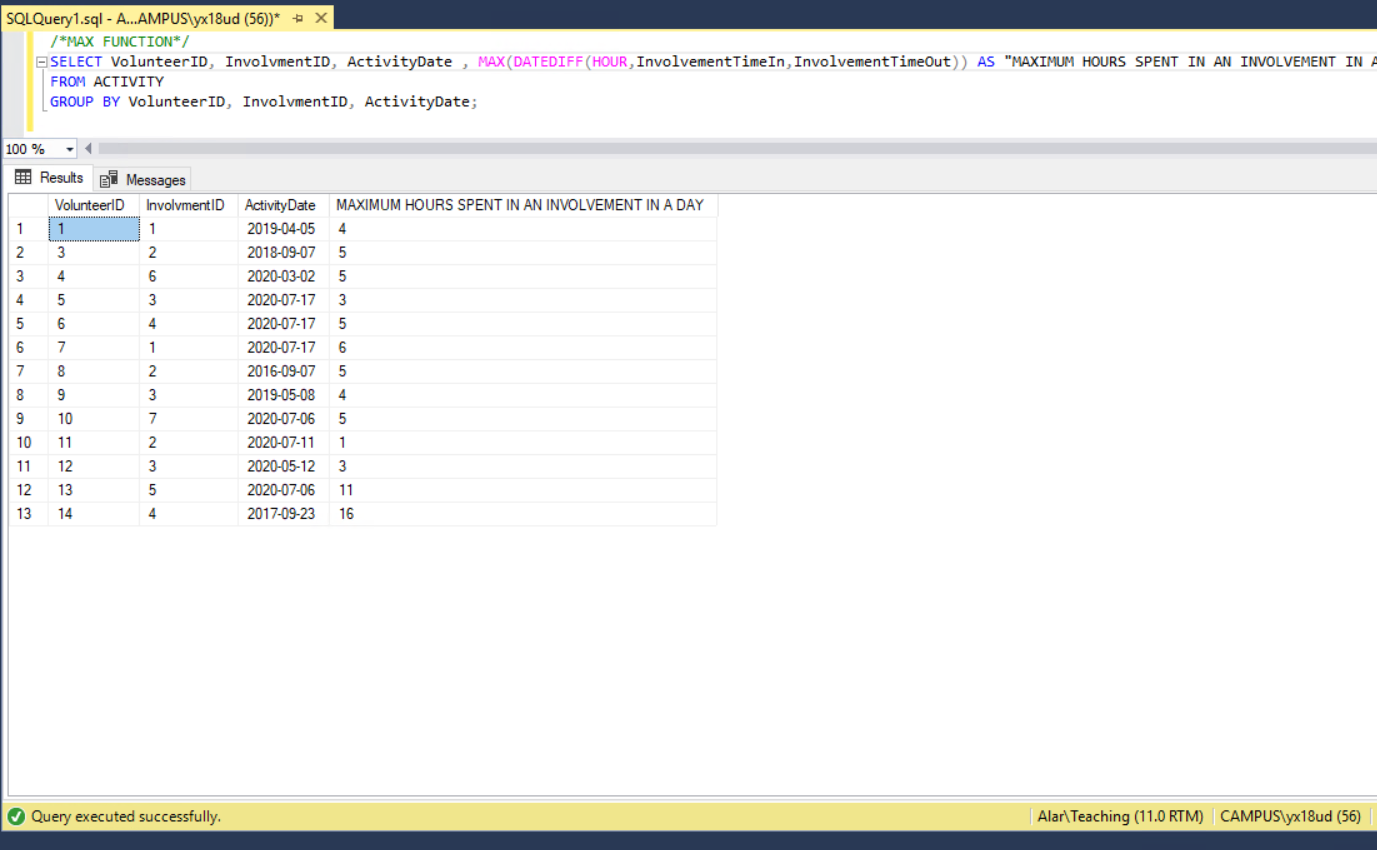
**Appendix 20:**



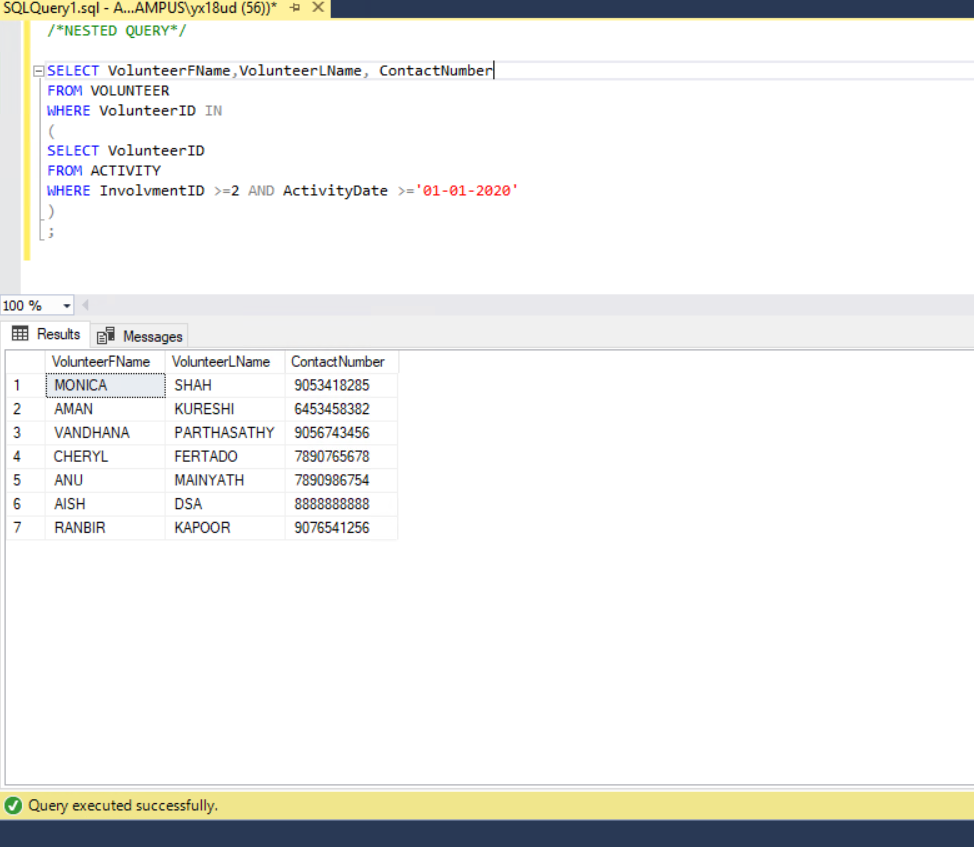
**Appendix 21:**



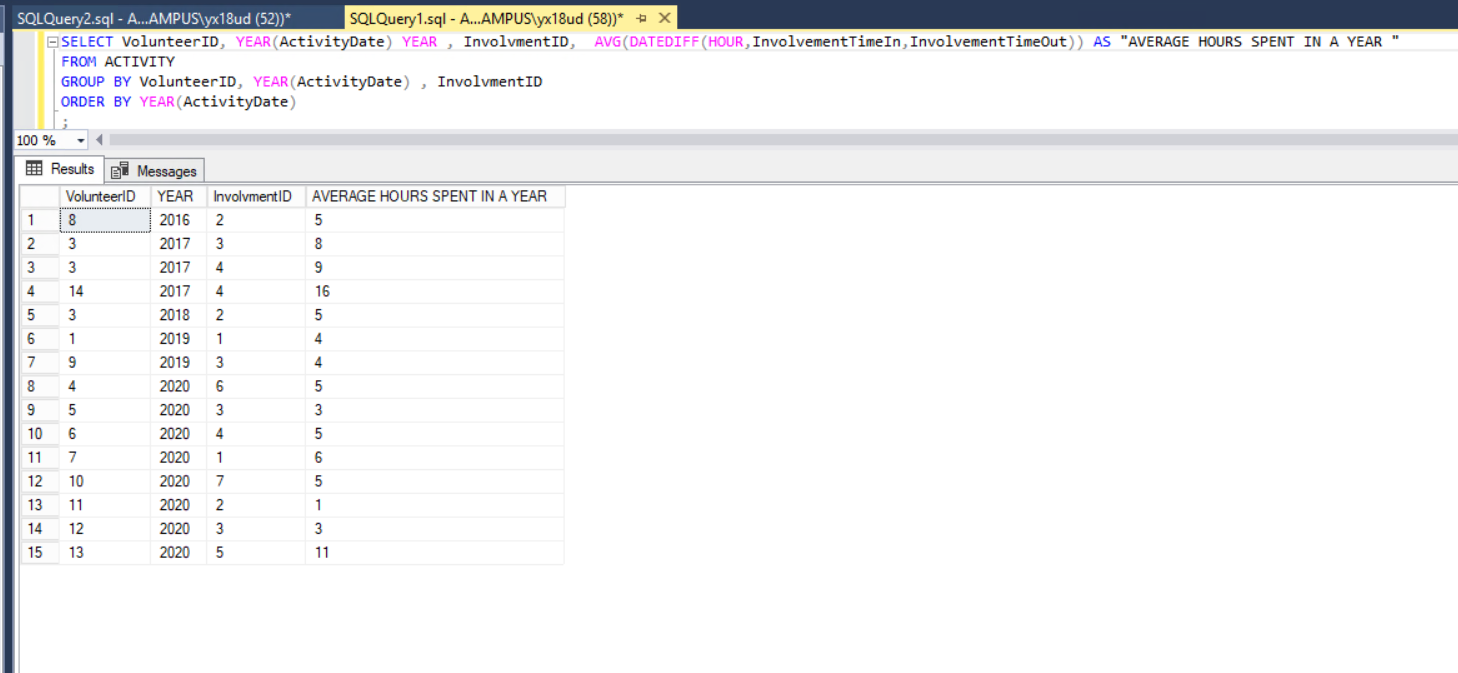
**Appendix 22:**



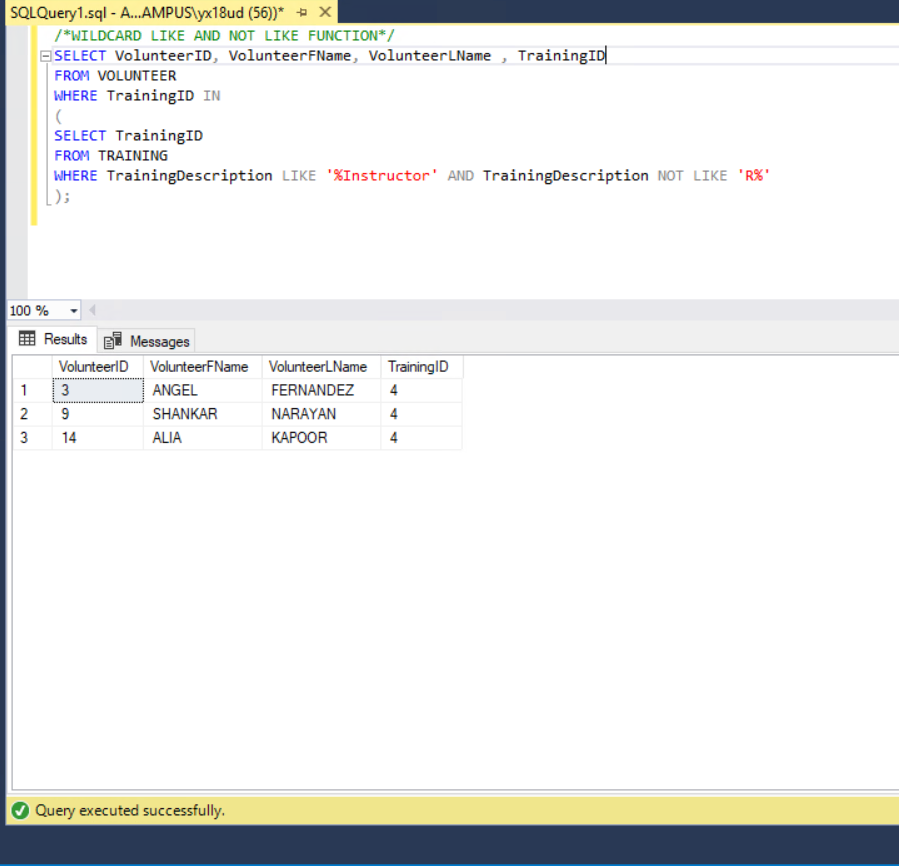
**Appendix 23:**



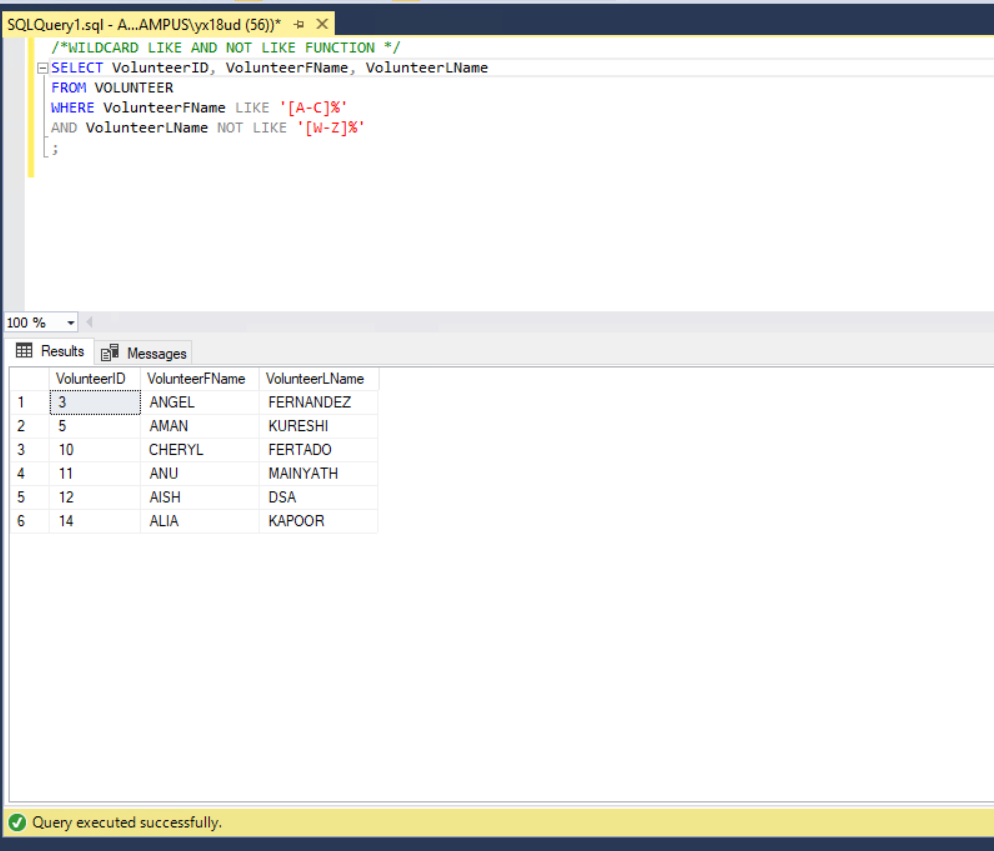
**Appendix 24:**



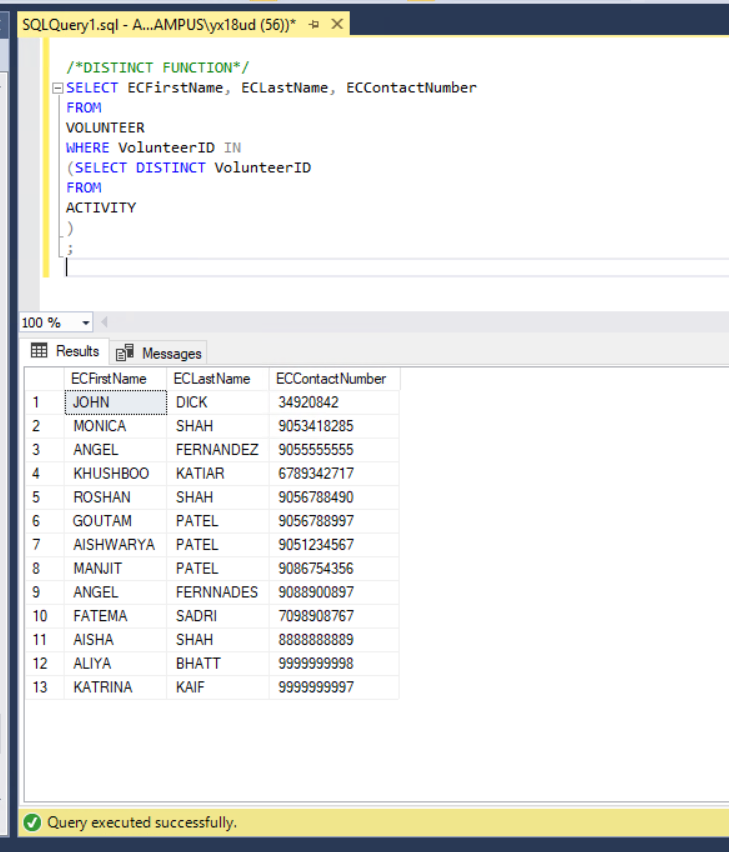
**Appendix 25:**



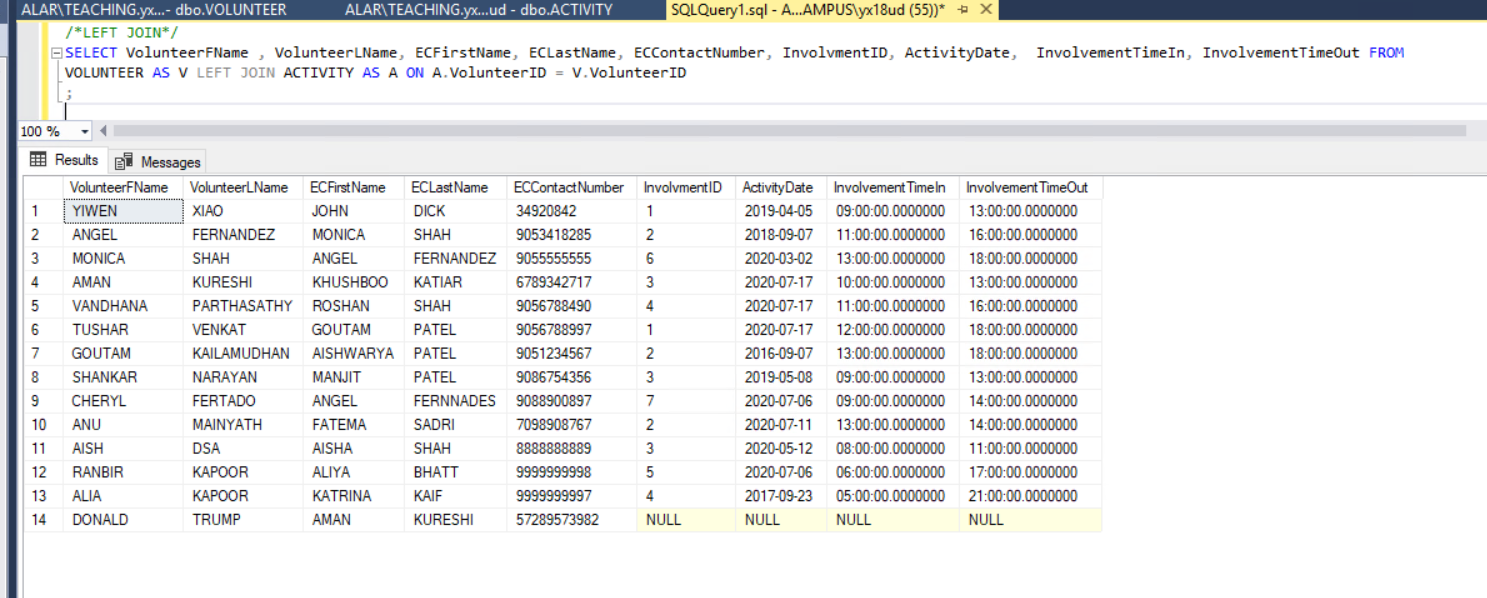
**Appendix 26:**



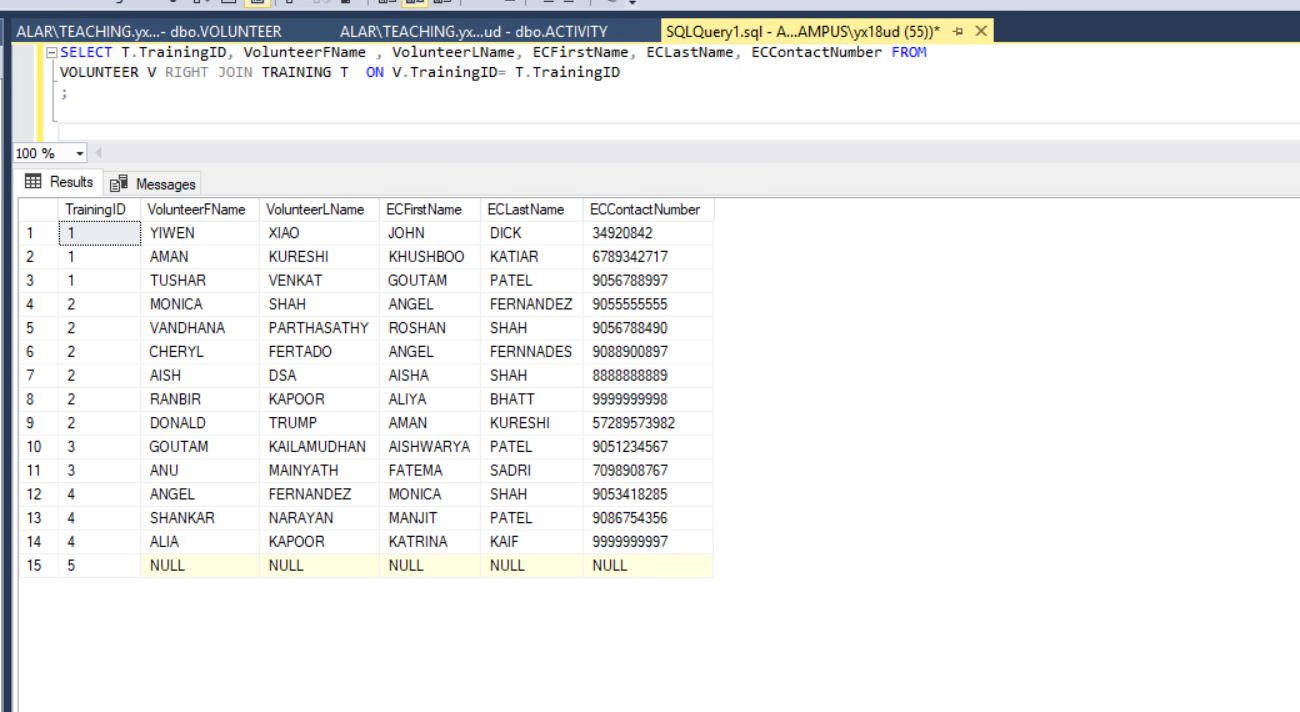
**Appendix 27:**



**Appendix 28:**



**Appendix 29:**

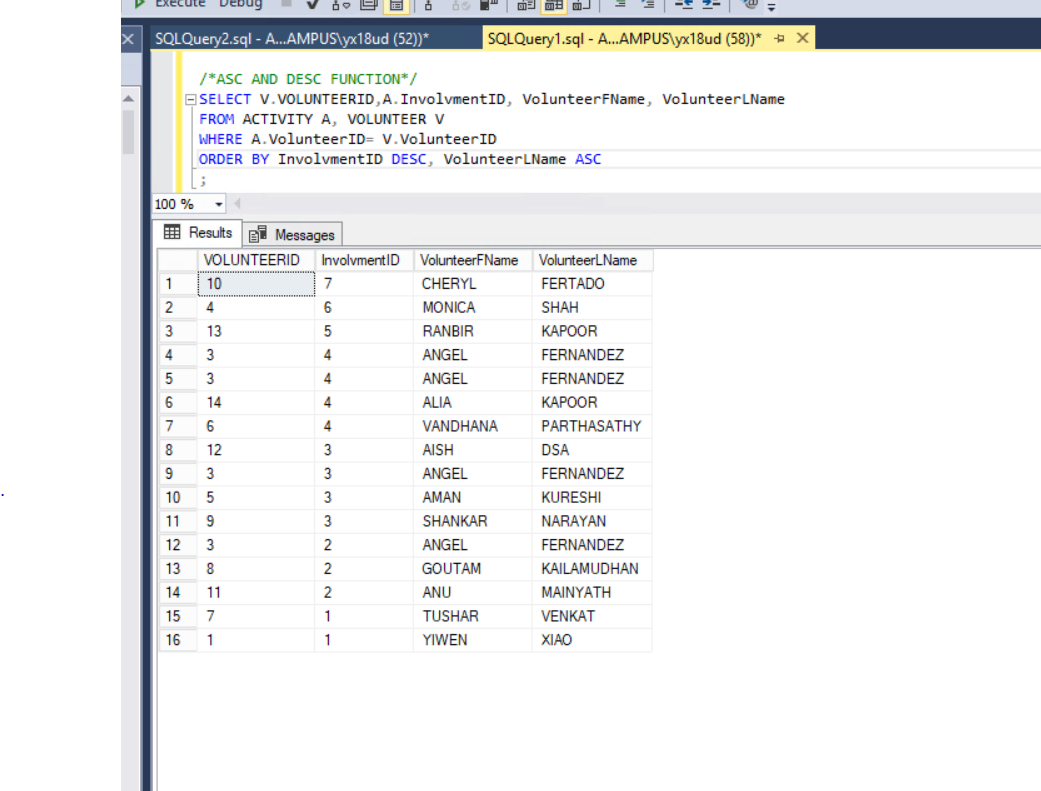


**Appendix 30:**

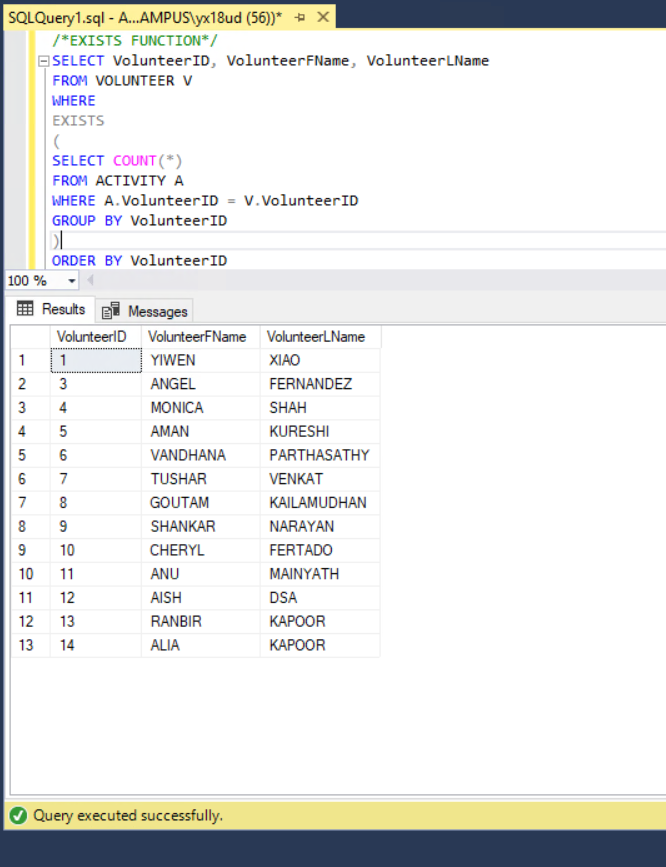
A screenshot of a social media post

Description automatically generated

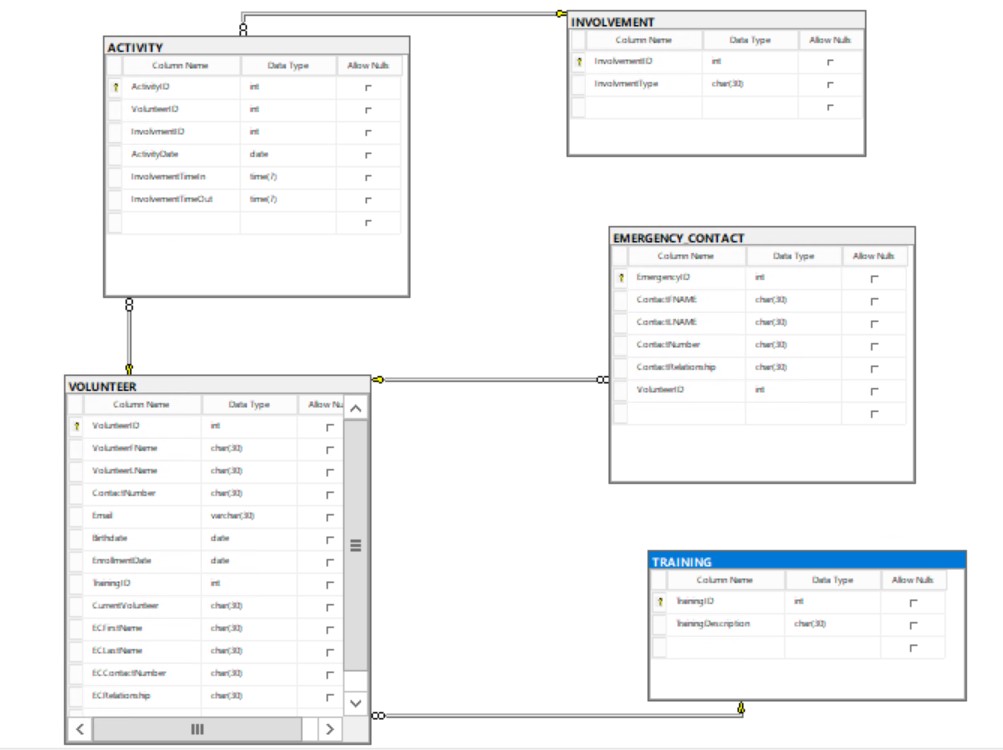
**Appendix 31:**



**Appendix 32:**



**Appendix 33:**

****