# Technical Documentation

# Introduction

In this document you will find a brief explanation about the new Social Ministry database system which will be used mainly to store and retrieve information; beneficiary’s, trainers, organizers, head organizer associated with our new system, Where each user we will identify what they are accessed to do in the system and show how the system implemented all there accessibilities from privilege and no privilege, thus shows the effectiveness of all requirements and how different implementations will be held by users and its need

This documentation will describe functionalities and architecture of our database system, it will focus on different main sides such as the system requirements and how it was handled, the system design(GUI) and its implementation process and the system architecture. This documentation main process is to show how the system main functionalities work and why it was implemented, provide instructions for the software technologies used and as well provide instructions of procedures and current access to specific users such as the ministry organizers and head. In this technical report we focused on everything from the user’s point of view, and it’s a consistent form that users can refer to when needed.

# Physical Schema



# Database Development

## Database Overview

|  |  |  |
| --- | --- | --- |
| **Table** | **Name** | **Description** |
|  | Beneficiary | This table mainly focuses on storing Beneficiary data such as:  **Beneficiary\_ID**: Is a PK, INT and not null uniquely identified for each beneficiary used mainly to retrieve each beneficiary record.  **fname, midname, lname:** Three attributes varchar(20) and not null main aim is to maintain with the primary key a unique record for each beneficiary and are stored in the UQ\_Beneficary\_Fullname,  **sex**: A char(1) , not null that enters one character and checks if its M or F.  **date\_of\_birth:** This stores the date of birth for each beneficiary with the data type DATE and is not null.  **dept:** Stores the dept for each beneficiary with the data type varchar(50) and is not null due to the importance of storing the beneficiary dept. |
|  | Course | This table stores the necessary data of each course such as:  **Course\_ID:** Is a PK, INT and not null uniquely identified for each course which is used mainly to retrieve each course record.  **Name:** Stores the courses name with the varchar(30) and not null to maintain all courses records are stored with names.  **Credit\_Hours**: Stores credit hours for each course in INT form maintaining all courses records are stored with its hours and its not null.  **Prerequisites**: Stores the courses, skills, or knowledge that a beneficiary must have completed or acquired before enrolling in the course with a varchar(50) and a null constraint since it’s not necessary to be completed in order to register in a course.  **Description:** Enters a description with the data type text and null allowing for empty description for the course.  **Delivery\_Method:** Stores the method thought in the course with the data type varchar(15) and not null to maintain for each table a delivering method.  **Enrollment\_Capacity:** Stores the max num of beneficary allowed to be stored in each course with the data type int and not null ensuring each course has a enrolment capacity. |
|  | Organizer | This table stores all the necessary data of each organizer such as:  **Organizer\_ID:** Is a PK, INT and not null uniquely identified for each organizer which is used mainly to retrieve each organizer record.  **fname, lname**: Two attributes stores the first and last name of the organizer with the varchar(20) each and a not null constraint not allowing for organizer’s records to be nullable names.  **starting\_Date:** stores the starting date of the last date enrolled in a role with the date and not null attribute ensuring entered dates for each employee.  **salary**: Stores the salary for each employee in a decimal form of 6,2 and not null.  **date\_Employeed:** Stores the date of employment for each organizer with a date data type and a not null constraint ensuring entered data. |
|  | Program | This table stores the necessary data for each program with the following attributes:  **Program\_ID**: Is a PK, INT and not null uniquely identified for each program which is used mainly to retrieve each program record.  **Name**: Stores the name of the program with the varchar(50) uniquely ensuring for each program one name.  **Level:** Stores the general level of the program with the varchar(30) and null since its not a main requirement to have a level and if no value was entered the default constraint inserts intermediate |
|  | Schedule | Each user in our system from the beneficary, trainer and organizer should have a schedule in this table we view there schedules and store the necessary data such as:  **Schedule\_ID**: Is a PK, INT and not null uniquely identified for each Schedule which is used mainly to retrieve each Schedule record.  **Beneficary\_ID**: A INT null foreign key that connects both the beneficary and the schedule table.  **Trainer\_ID**: A INT null foreign key that connects both the Trainer and the schedule table.  **Organizer\_ID**: A INT null foreign key that connects both the Organizer and the schedule table.  **Start\_Date:** Store the date when the schedule was made with the DATE data type and not null to ensure storing the schedule first date.  **Status:** Stores the schedule status scheduled or not with a varchar(15) and not null. |
|  | Trainer | This table stores the necessary data for each Trainer with the following attributes:  **Trainer\_ID**: Is a PK, INT and not null uniquely identified for each Schedule which is used mainly to retrieve each Schedule record.  **Course\_ID:** A INT null foreign key that connects both the Course and the schedule table.  **fname, lname:** Two attributes stores the first and last name of the organizer with the varchar(20) each and a not null constraint not allowing for trainer’s records to be nullable names.  **Expertise**: Stores the trainer expertise using a varchar(50) and not null. |
|  | Beneficary\_Addresses | This table stores the data of a multi-value attribute in the beneficary table with two main attributes,  **Beneficary\_ID**; A part of a primary composite attribute with a INT data type that connects the Beneficary\_Addresses and the Beneficary table.  **Addresses:** The second part of the composite primary key where it stores addresses for the beneficary’s with a varchar(50) and not null. |
|  | Beneficary\_Emails | This table stores the data of a multi-value attribute in the beneficary table with two main attributes,  **Beneficary\_ID**; A part of a primary composite attribute with a INT data type that connects the Beneficary\_Emails and the Beneficary table.  **email:** The second part of the composite primary key where it stores addresses for the beneficary’s with a varchar(50) and not null and uniquely identified to ensure a unique email across all the records. |
|  | Beneficary\_PhoneNumbers | This table stores the data of a multi-value attribute in the beneficary table with two main attributes,  **Beneficary\_ID**; A part of a primary composite attribute with a INT data type that connects the Beneficary\_PhoneNumbers and the Beneficary table.  **phone\_number**: The second part of the composite primary key where it stores phone numbers for the beneficary’s with a varchar(15) and not null and uniquely identified to ensure a unique phone number across all the records. |
|  | Course\_Location | This table stores the data of a multi-value attribute in the Course table with two main attributes,  **Course\_Location**; A part of a primary composite attribute with a INT data type that connects the Course\_Location and the Course table.  **location:** The second part of the composite primary key where it stores location for the Courses’s with a varchar(255) and not null. |
|  | Course\_Program | This table is a production of a many-to-many relationship between the course and program table where we store:  **Course\_ID** and the **Program\_ID** with both a INT data type and is a composite primary key where each part of the key is a connection to another table and is a foreign key itself. |
|  | Course\_Schedule | This table is a production of a many-to-many relationship between the course and Schedule table where we store:  **Course\_ID** and the **Schedule \_ID** with both a INT data type and is a composite primary key where each part of the key is a connection to another table and is a foreign key itself. |
|  | Orgainzer\_PhoneNumbers | This table stores the data of a multi-value attribute in the Orgainzer table with two main attributes,  **Organizer\_ID**; A part of a primary composite attribute with a INT data type that connects the Orgainzer\_PhoneNumbers and the Orgainzer table.  **phone\_number**: The second part of the composite primary key where it stores phone numbers for the beneficary’s with a varchar(15) and not null and uniquely identified to ensure a unique phone number across all the records. |
|  | Program\_Organizer | This table is a production of a many-to-many relationship between the Organizer and program table where we store:  **Organizer \_ID** and the **Program\_ID** with both a INT data type and is a composite primary key where each part of the key is a connection to another table and is a foreign key itself. |
|  | Trainer\_PhoneNumbers | This table stores the data of a multi-value attribute in the beneficary table with two main attributes,  **Beneficary\_ID**; A part of a primary composite attribute with a INT data type that connects the Beneficary\_PhoneNumbers and the Beneficary table.  **phone\_number:** The second part of the composite primary key where it stores phone numbers for the beneficary’s with a varchar(15) and not null and uniquely identified to ensure a unique phone number across all the records. |
|  | Program\_Beneficary | This table is a production of a many-to-many relationship between the Organizer and program table where we store:  **Beneficary\_ID** and the **Program\_ID** with both a INT data type and is a composite primary key where each part of the key is a connection to another table and is a foreign key itself. |

|  |  |  |
| --- | --- | --- |
| **View** | **Name** | **Description** |
|  | Head\_Organizer\_Access | This view was Created for the head organizer where he can view all the programs organizers organize and access all the organizers info with each program, we joined all necessary tables to ensure that right output was executed, one of the main roles of the head organizer is to maintain all organizers info which this view easily helps the head organizer access it with it’s alternative programs without the need for him to right the query each time he wants to get the organizers info to its respective program. |
|  | Head\_Organizer\_ Beneficiary\_Info | In this view we gave the head organizer the ability to view all beneficiary’s information within each program and joined all necessary tables such as program\_Beneficary to ensure each program with it’s respective beneficiary, Beneficiary table to retrieve the according info for each beneficiary as well as the beneficiary\_phoneNumbers and beneficiary\_addresses tables to make the head organizer view as well there respective address and phone number and the program each beneficiary is registered to. |
|  | View\_Organizer\_Hassan\_ farsi\_Schedule | Organizers main role is in assigning courses to its respective program and add and update according for each organizer his/her respective program, so we created a view for Hassan one of the organizers to view all the programs he’s responsible for within a schedule and its respective course for him to check if there are any needed updates them. |
|  | View\_Trainer\_Jane\_Smith\_Schedule | Trainer main role is to teach courses and be able to update on the course information he/she teaches so we created a view for the trainer jane smith that helps him view all the courses information he teaches from its location, description and all detailed information of the course within his schedule, this helps the trainer have a continuous access on all course he teaches. |
|  | View\_Beneficary\_Khaled\_Mohammed\_Schedule | The beneficiary should has a schedule where it shows all his registered programs and its respective courses he is enrolled in details from the courses location and its important information he needs and the trainer who teaches these courses and be able to view his schedule information from the status and the starting date of the schedule. |
|  | View\_Courses\_To\_ Beneficary | In order for the beneficiary to view all the courses and navigate to see which program and it’s courses to register to, this view help the beneficiary to view all **needed** information such as the enrolment capacity isn’t an attribute that’s allowed for the beneficiary to view that’s why having the beneficiary have a direct access on the course table to select and view it would be not the best practice. Therefore, help him as well have a more clear picture of the programs and it’s respective courses |

|  |  |  |
| --- | --- | --- |
| **Procedure** | **Name** | **Description** |
|  | Organizer\_InsertNewProgram | Organizers have the privilege to create and add a program in thi procedure we focused on obtaining the organizer ID and the program ID and its info to add the program first and then the procedure assigns the entered organizer ID with the program to make responsible on the respective added program |
|  | Organizer\_UpdateProgram | This Procedure checks first whether the entered organizer ID is the one responsible of the entered program ID where it checks it from the program\_organizer table if it’s yes assigned to it he can update the program details else it gives him a message that he cant update the program details. |
|  | Organizer\_AssignCourseTo Program | One of the organizers main role is to assign courses to its respective program this procedure first checks whether the entered organizer ID is responsible on the wanted to assign to program ID if the organizer is responsible of it from the program\_organizer table he inserts first the course info and then connect it to the program through inserting the entered course info ID with the program he is responsible on in the course\_program table.  Noting: This procedure focuses on inserting the entered course with the program but doesn’t focus on connecting existing courses with programs. |
|  | Organizer\_UpdateCourseInfo | In this procedure we gave the organizer the ability to update the course information where he enters his ID, the program of the course he wants to update and the course ID it checks first if the organizer has an access on the perspective program course if yes and if the course ID he entered is connected to the same program it updates the course info. This procedure helps the organizer update the course info without the need of writing an entire query specifies updating the course now any organizer can access this procedure and update a course. |
|  | Trainer\_UpdateCourseInfo | Trainers have the access to update the courses they are only assigned to where first this procedure checks if the trainer is assigned to the course from the trainer table it checks if the course is assigned to it or not if yes, it allows the user trainer to update the info if not it doesn’t allow to make the respective update, this procedure helps trainers on upating there courses info and demonstrates a higher level of security where we ensure that the trainer is allowed to update the course or not. |

## Security

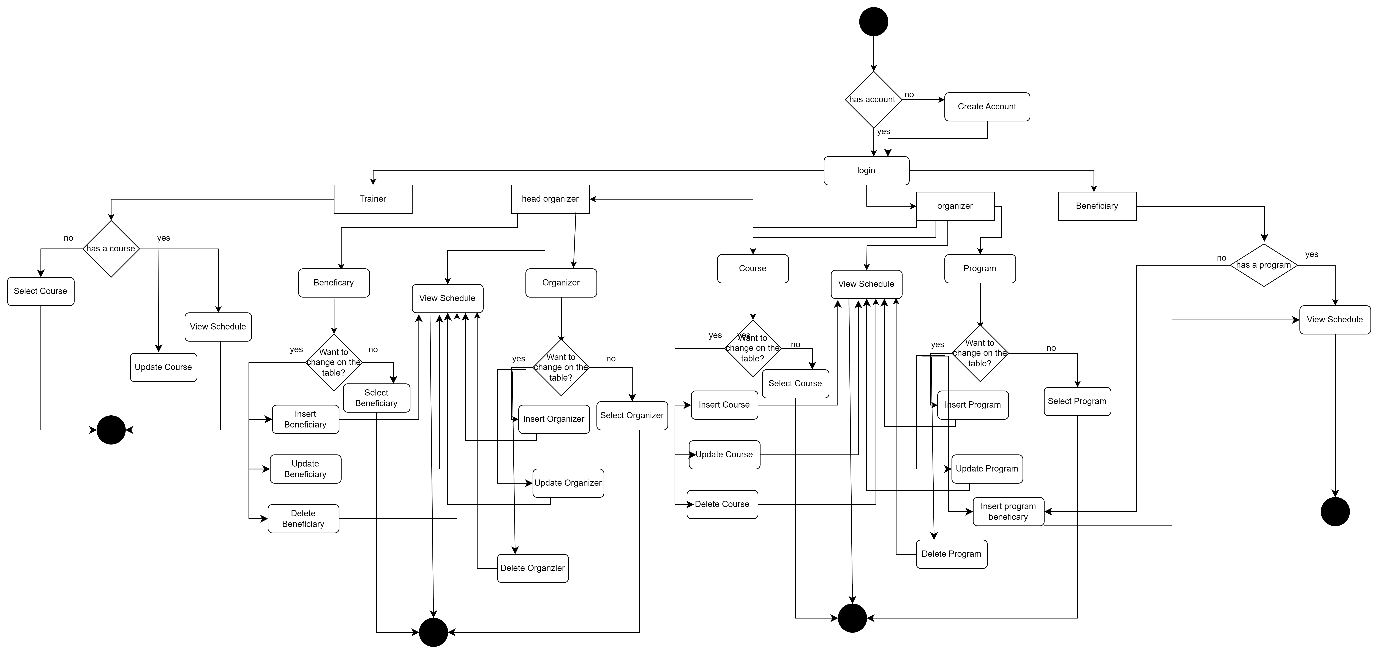
|  |  |  |  |
| --- | --- | --- | --- |
| **User name** | **Privilege Command** | **Description** | **Screenshot** |
| 6  **User Name** | **Privilege Command** | **Description** | **Screenshot** |
| Head\_Organizer | GRANT SELECT, INSERT, DELETE, UPDATE ON [dbo].[Head\_Organizer\_Beneficiary\_Info] TO Head\_Organizer; | Accessing all beneficiary info within each program helps the head organizer have a more picture of the structure within, so we granted the head organizer the access on the [dbo].[Head\_Organizer\_Beneficiary\_Info] where it enabled him to select, insert, update and delete from this view records which will respectively change in the tables used in the view. |  |
|  | GRANT SELECT, INSERT, DELETE, UPDATE ON [dbo].[Organizer] TO Head\_Organizer; | The head organizer has a complete access on the organizer selecting, inserting, updating and deleting where we granted in this command for him to do so, which despite the view [dbo].[Head\_Organizer\_Access] that does a similar attitude ,accessing the organizer table directly helps him create different query’s related to the table. |  |
|  | GRANT SELECT, INSERT, DELETE, UPDATE ON [dbo].[Beneficary] TO Head\_Organizer; | The head organizer has a complete access on the beneficary selecting, inserting, updating and deleting where we granted in this command for him to do so. Despite the view [dbo].[Head\_Organizer\_Beneficiary\_Info] that does a similar attitude ,accessing the beneficary table directly helps him create different query’s related to the table. |  |
|  | GRANT SELECT ON [dbo].[Program] TO Head\_Organizer; | Head organizer main role is to maintain the structure of the system so we granted him the access to select the program table in order for him to check the organizers updates and maintainens of the programs . |  |
|  | GRANT SELECT ON [dbo].[Course] TO Head\_Organizer; | As mentioned he maintains the structure so selecting the course table is essential for him to check the organizers updates and maintainens of the courses. |  |
| Organizer\_ Hassan\_farsi | grant execute on [dbo].[Organizer\_AssignCourseToProgram] TO Organizer\_Hassan\_farsi; | Different procedures where granted to the organizer hassan in order to help him make current action, in this command we granted him the access to use the procedure [dbo].[Organizer\_AssignCourseToProgram] where it assigns the program the organizer is reasponsible for to its inserted course. |  |
| grant execute on [dbo].[Organizer\_InsertNewProgram] TO Organizer\_Hassan\_farsi; | In this command we granted him the access to use the procedure [dbo].[Organizer\_InsertNewProgram] that inserts a program and connects the organizer to the inserted program |  |
| grant execute on [dbo].[Organizer\_UpdateProgram] to Organizer\_Hassan\_farsi; | In this command the organizer was granted the access to use the procedure [dbo].[Organizer\_UpdateProgram] that updates only the programs the organizer is responsible for. |  |
| grant execute on [dbo].[Organizer\_UpdateCourseInfo] to Organizer\_Hassan\_farsi; | In this command the organizer was granted the access to use the procedure [dbo].[Organizer\_UpdateCourseInfo] that updates only the courses of the programs the organizer is responsible for. |  |
|  | grant select, update, delete, insert on [dbo].[Course] to Organizer\_Hassan\_farsi; | In this command he could select, update, delete, insert into the course table different query’s related. |  |
|  | grant select, update, insert on [dbo].[Program] to Organizer\_Hassan\_farsi; | In this command he could select, update, delete, insert into the program table different query’s related. |  |
|  | grant select, insert, update on [dbo].[Course\_Program] to Organizer\_Hassan\_farsi; | In this command he could as well select, update, insert into the course\_program table different query’s related. |  |
|  | grant select on [dbo].[Program\_Organizer] to Organizer\_Hassan\_farsi; | In this command he could select only the program\_organizer table to view each program and its organizer but can’t change on its record. |  |
|  | grant select on [dbo].[View\_Organizer\_Hassan\_farsi\_Schedule] to Organizer\_Hassan\_farsi; | Each organizer have a schedule where he views generally all programs he’s reasponsible of and its courses info, in this command he is able to view his respective program and its courses to check and maintain it. |  |
| Trainer\_ Jane\_Smith | grant select on [dbo].[View\_Trainer\_Jane\_Smith\_Schedule] to Trainer\_Jane\_Smit; | Trainers have the ability to view in there schedule all courses they teach information and location and maintain a clear view of it, In this command we granted him the abilty to view his schedule. |  |
| grant select on [dbo].[Course]to Trainer\_Jane\_Smith; | The trainer has the ability only to select the course table and view all courses information which was granted to the user. |  |
|  | grant execute on [dbo].[Trainer\_UpdateCourseInfo] to Trainer\_Jane\_Smith; | In this command we granted the trainer the ability to update only the courses he is only responsible of through the procedure [dbo].[Trainer\_UpdateCourseInfo] |  |
| Beneficary\_ Khaled\_Mohammed | grant select on [dbo].[View\_Beneficary\_Khaled\_Mohammed\_Schedule] to Beneficary\_Khaled\_Mohammed; | The beneficiary Khaled has the ability to view his schedule which has all the registered courses and its locations which was granted in this command. |  |
|  | grant select on [dbo].[View\_Courses\_To\_Beneficary] to Beneficary\_Khaled\_Mohammed; | This command grants the beneficiary the ability to view current info of all the courses from the course table |  |

## Security

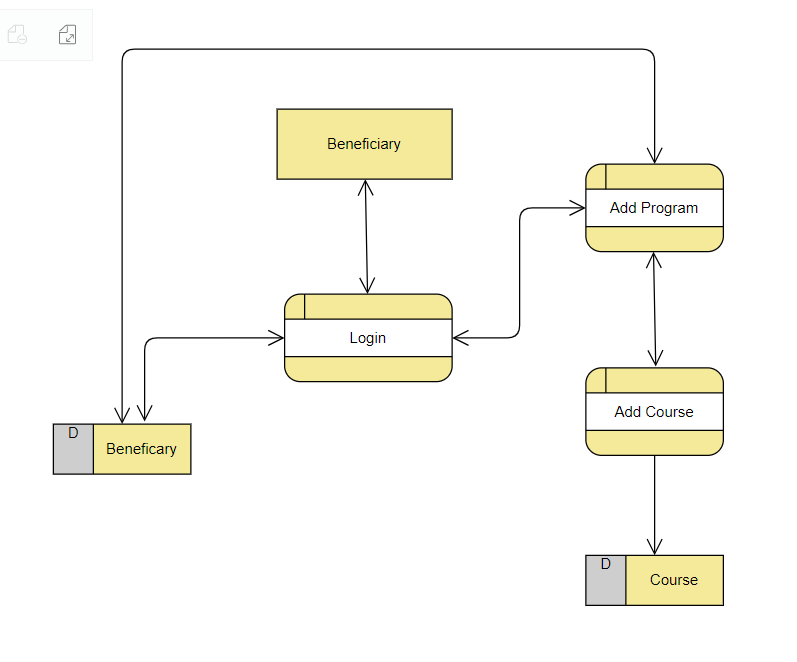
## User Interface

### Flowchart and Data Movement Diagrams

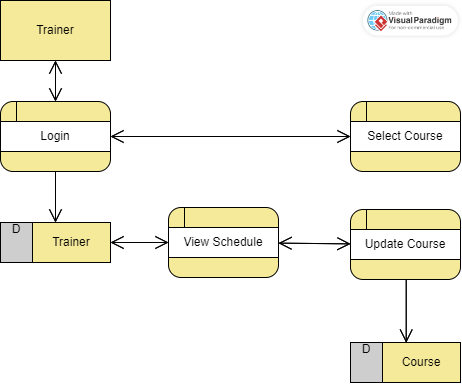
**Flowchart:**



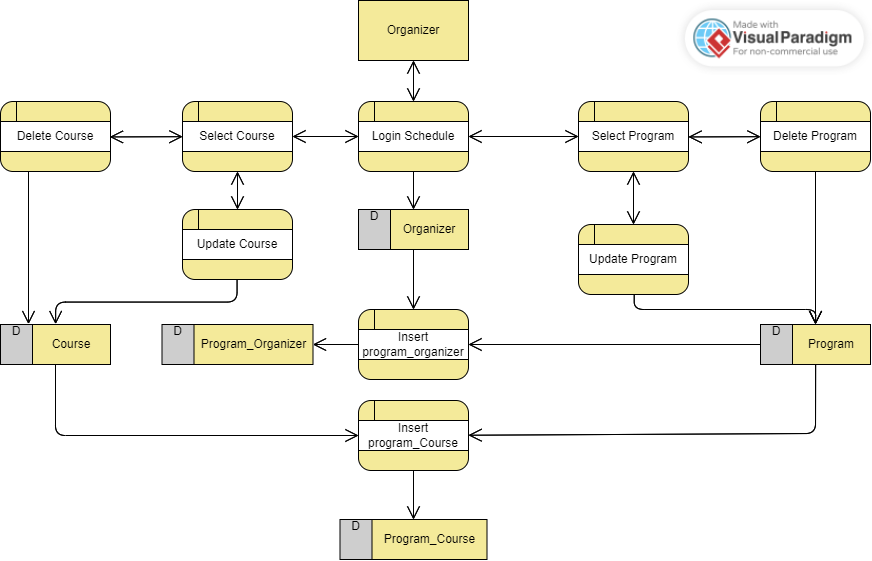
**Data Movement Diagrams**

User 1: Beneficiary

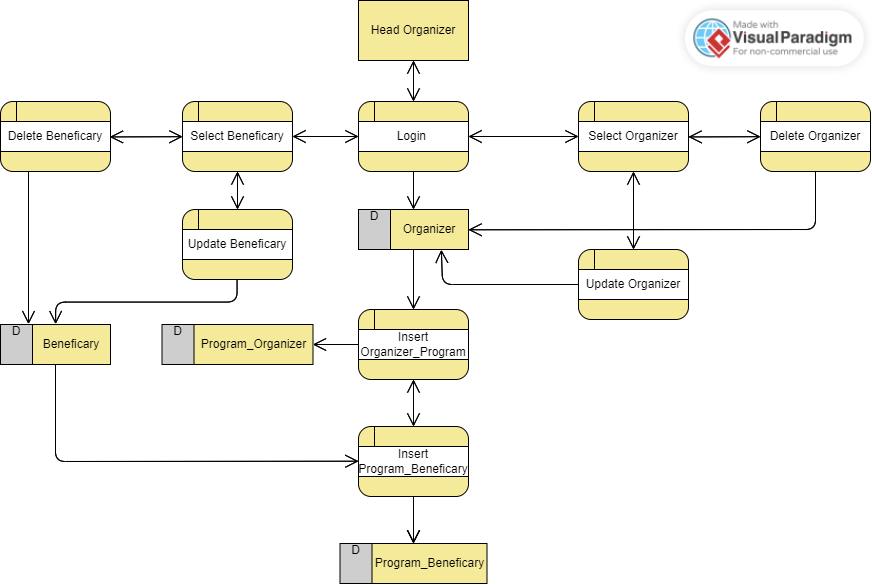
User 2: Trainer



User 3: Organizer



User 4: Head Organizer



### Interfaces Development

|  |  |  |  |
| --- | --- | --- | --- |
| **Page ID** | **Title** | **Description** | **Screenshot** |
|  | Home Page | The head organizer here is able to view all the tables he is permissioned to view the beneficiary and the Organizer tables. He is able to choose which table he wants to edit by navigating to the options along side the table. |  |
|  | Organizer View Table | After choosing the organizer table he views all records of the table and all information associated with the table he can navigate and choose which record he wants to make a current action on from editing, deleting and on the end of the records he can add a new record. |  |
|  | Editing Record Page | After choosing one of the records edit he is views a edit page that allows him to choose the attribute he wants to edit and either save his edit that will be accordingly updated or close the page. |  |
|  | Deleting Record | The head organizer decided to delete a record from the organizer view table he is able to delete a record and when he navigated to delete a record it gave him a caution on whether he’s sure on deleting the record or not. |  |
|  | Adding Record | On the end of the organizer table there is an option on adding a record when the head organizer chose to add a record a page with empty attributes to the table appeared in order to fill the record with the values the head organizer wants to add giving him the option to either save the added record or close it, as shown we added a record that wasn’t there and it was added successfully. |  |
|  | Error Page | This page accured to the attempt of the head organizer to enter a record with the Organizer\_ID = 46 that was already added by him and is inserted in the table, so a page showing an error message accured stating that the Save failed: ('23000', "[23000] [Microsoft][ODBC SQL Server Driver][SQL Server]Violation of PRIMARY KEY constraint 'PK\_\_Organize\_\_79B53581BB18E941'. Cannot insert duplicate key in object 'dbo.Organizer'. The duplicate key value is (46). Which means the save was failed due to a duplicate entry of the primary key 46 across the table. |  |

# Maintenance

## Database recovery & backups

**Database Backup**

The process of creating a copy of the structured data of a organization by a current database, it’s main purpose is to create a copy of the original data to reconstruct in the need of the data and database in different cases such as accidental deletion or attacks. A database backup is a protection data solution for structured data in database workloads. Where companies require backup and recovery their data whether on-premises or in cloud or both environments. Structures data housed databases is critical for businesses who rely on them to store data as a backend for many business. As cost of the downtime increasing companies must ensure that they have a secure backup of their data to ensure recovery.

While the database is running the database backup solutions now perform the backup operations while the database is running such a backup operation resents in a more work behind the scenes and that’s because the data is ongoing workload while it is trying to copy it out in the backup for the protection. The database workload protection uses an integrated method that performs a backup of all files in the database without the need of shutting down the database engine, this ensures the database are not corrupted during the backup process. To prevent continuous changing structured data the best bractice is to perform a scheduled database backups frequently.

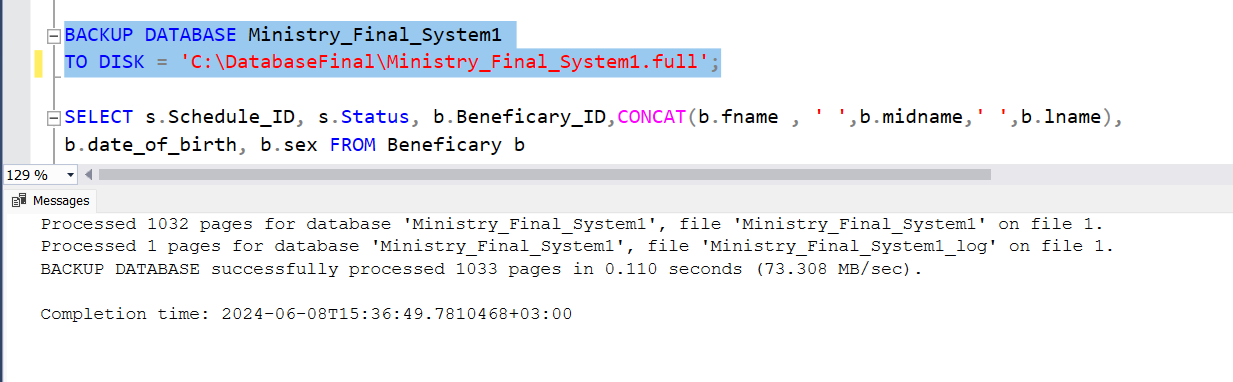
**It's Importance**

Databases are in continuous growth from it’s size, speed and complexity than before and businesses are relying on these bigger and faster databases for storing their info where if it was lost their entire data will be not replaceable, where the importance of database backup arises in reducing downtime and database recoverability from attacks. Data security is a main reason as well for having a database backup, if a company suffers from a attack the recovery here may offer a backbone on such a case where it’s possible if the backup recovery is implemented to go back before the attack even happened and restore your data. Therefore, without an implemented backup organizations can’t recover there data if lost in any condition where the entire business can’t keep running. In conclusion, it enhances the business continuity, it provides a data protection workload providing a usable copies of lost data and provides a disaster recovery where when facing an unexpected events that cause data loss the database backup ensures readiness and reduces risks.

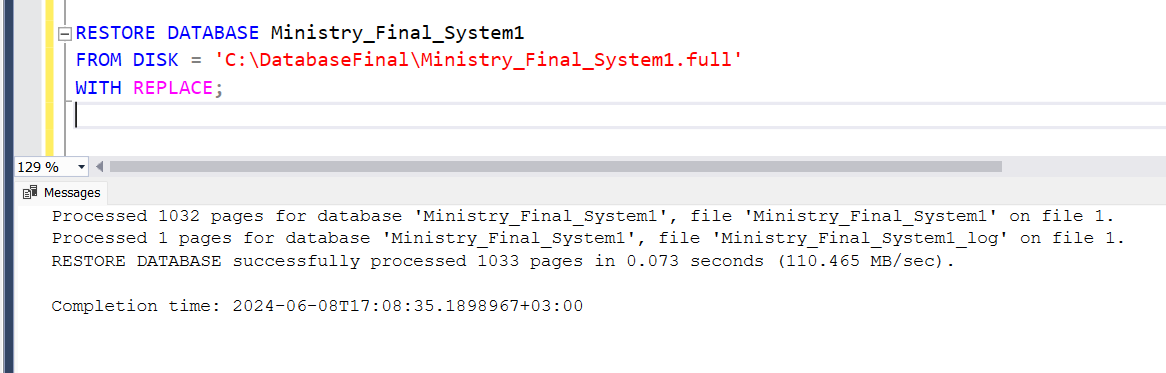
**Backup Types and Process**

**Full Backup:** It’s the backup that contains a copy of the entire database which includes the data pages and the log files written in the process of the backup. In order to restore the database a full backup is what you need where you can’t restore a differential backup nor a transaction log backup without a full backup.

Let’s illustrate performing a full backup on to our database system:



To recover the database after storing the backup we use:



**Differential Backup:** This backup contains only the data that has been modified since the last full backup where it is considered identical to the full backup

To perform a differential backup we use the BACKUP DATABASE statement with DIFFERENTIAL

BACKUP DATABASE database\_name

TO DISK = path\_to\_backup\_file

WITH DIFFERENTIAL;

**Transaction Log Backup:** It contains all changes made to the database. A transaction log backup requires using the full or bulk-logged recovery because of the need of the truncate the log. The two subsequent log backups don’t contain redundant data. Plus, the transaction log backup has no impact on performance which gives you the ability to perform it when the workload is high.

To perform a transaction log backup we use the BACKUP LOG

BACKUP LOG database\_name

TO DISK = path\_to\_backup\_file;

## Database maintenance in general

**Database Maintenance**

A set of procedures that ensures your database is operating properly is known as database maintenance. Different database management solutions such as oracle and SQL make regular updates to align with the new technology and security risks, compromising different duties to contribute the safety on your data is formed in many tasks one of these tasks are the export/import operations, it’s assigned as a critical aspect since it allows a recovery from issues that have affected your system,, such as data corruption or hardware failures, this enables you to back up or migrate pieces of the database to another system for backup needs.

Database maintenance verses the characteristics and features of the system and handles it with different many components such as the backup of current data, any issues may occur handles it by retrieving the data from the backup. Generally, backups are insufficient for database maintenance that’s why there are different other additional strategies that maintains the system operations.

**Significance**

There are many advantages of having a database maintenance routine that’s why many leading organizations don’t hesitate to implement a good maintenance plan.

* **It Keeps Companies Up To Date:** Organizations are already having a hard time in keeping it up with the speed of technology where failing to adapt to the latest trends may lead them to a slow in there growth, that’s why regular database maintenance is a must in these leading organizations that ensures your data is as current as it gets, catering to your audience and ensuring a an up to date status on your database system.
* **Protects From Malicious Threats**: Any system may be attacked by cyber threats, any small threat may be the main cause to a shutdown to your own system. Database maintenance ensures a regular assist where it reduces the downtime and guards against and malicious attacks, heling you to discover the threats, but also gives you a sight on how you can fix them.
* **Saving Time:** Rather than spending time on database administration activities, rather the database maintenance techniques are automated, once you insatiate them it will work to clean you your system for you saving your time and hands it to you for other productive tasks.

**Different Methods of Database Maintenance**

* Over time, as data is added, updated and deleted the database files can become filled with unused space, that happens because the database management system most often does not reclaim the space of deleted records. **Compacting the database** happens by first before performing any maintenance ensuring that the complete backup of the database is saved preventing any accidental data loss, then open each table in the database and delete all records then save the changes to the table. That is done to improve the performance where reclaiming unused space helps improve the database performance and reduces the file size and enhances data integrity where it prevents potential corruption
* **Resetting Auto values** ,when a primary is set to auto-increment, the values can rise to large numbers even after deleting the records, as the auto-increment does not reset automatically, This method involves using the SQL command ‘ALTER TABLE “table\_name” AUTO\_INCREEMNT = 1;’ to reset the primary key starting value, replacing the table\_name with the actual table name and after = with the desired starting value for new records. This method mainly performs a consistent primary key value where you ensure that the primary key is kept in a reasonable range making it easier to manage and understand the data.

# Testing

## Data Validation

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Type** | **Description** | **screenshot** |
|  | All cases of PK | **Uniqueness validation** was accrued due to attempting to insert into the organizer table a record with the same primary key already inserted, that happened due to the primary key constraint on the organizer\_ID that is uniquely identified and doesn’t allow multiple records with the same primary key which in another hand identifies each record uniquely and enhances dependency between the data |  |
| **Nullability validation** occurs due to attempting to insert a record with the primary key as a null value which happened in attempting to insert a record into the organizer table with the organizer\_ID set not set, what explains that is as mentioned the primary key is a main attribute in each table that identifies each row setting it to null violets this concept entirely. |  |
|  | All cases of FK | **Referential Integrity** occurs when a statement is conflicted with the foreign key constraint when I inserted a value in the Course\_ID (Foreign key) that isn’t there in the other side of the connected table course as a primary key the error occurred due to no connection will be conducted in the entered value which violets the foreign key constraint. |  |
| **ON UPDATE CASCAE** is a constraint that ensures any updated status on the primary key all alternative foreign keys for the primary key will be updated as well in our case we updated the primary key in the course table to 14 where it was 10 so alternatively the course ID in the trainer table (the foregin key)all records where the foreign key was 10 became 14, here patricia became reasponsible on the course 14 not 10 . |  |
| **ON DELETE CASCADE** is a constraint that ensures any deleted status on the primary key all alternative foreign keys for the primary key will be deleted as well in our case we deleted the primary key 9 in the course table alternatively the course ID in the trainer table (the foregin key)all records where the foreign key was 9 were deleted |  |
| **ON DELETE NO ACTION** ensures that no action will be done when attempting to delete any primary key to the alternative foreign keys therefor when I attempted to delete the record where the primary key was 14 in the course table it didn’t delete since no delete no action was the constraint on the foreign key in the trainer table. |  |
|  | Unique | The fname, midname, lnyame attributes has a unique constraint UNIQUE(fname,midname,lname) that doesn't allow a any other record to have the same set of the three attributes, when attembpting to insert a record with the same set of the three attributes all toghteher the violation occurred stating that it’s not allowed to have the same set of name. |  |
|  | Default | The attitude of the default constraint is simply if a attribute is assigned a default constraint any non entered value will be filled with the default assigned value in our case since we didn’t insert the level attribute and it’s assigned a default value ‘intermediate’ the default value was inserted. |  |
|  | Not null | NOT NULL constrsaint doesn’t allow for the value to be null where when we entered the sex as a null value and it’s assigned as NOT NULL it violated the constraint and didn’t allow to insert the statement into the beneficiary table. |  |
|  | Check | The check constraint is a constraint that allows a current condition to be entered as a value in the attribute therefore in our case its allowed to either be M or F entered in the sex and since we entered X we violated the check constraint and it was denied to be entered. |  |
|  | ……… |  |  |

## Output Validation/

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **Query Description** | **Screenshot (query + result)** | **Result validation** |
|  | The head organizer wants to retrieve specific info of the first organizer employed within a each program such as the organizer full name and salary so he used a query that does two select statements the first retrieves the info according to the date and for the date he selected the minimum date within the program, let’s assume the first program as shown in the result validation all three organizers 21, 26 and 31 the first employed organizer is yara with the date 15-05-2016 and that’s what was executed in his query. |  |  |
|  | The head organizer decided to check what is the max and min salary of organizers within each program so he implemented a query where he grouped the condition where within the program name, let’s focus on the salary’s of the organizers within the first program we have three organizers 21,26 and 31 and if you notice from the organizer table the max salary of the three organizers is 800.35 and the min salary is 750.10 and that was retrieved successfully in his query. |  |  |
|  | The beneficiary with the ID has a new phone number so the head organizer should insert his phone new phone number here we used an insert statement in order to enter the new phone number and as shown the row was successfully effected due to the fact that the beneficiary is refrenced to the main table and the entered phone number is unique. |  |  |
|  | The head organizer wants to view all organizers from the first employed to the last employed organizer and as shown in the organizer table the first employed organizer is mona and the last employed is amal but each are in a row in the ORDER BY we specify what we want to order and how in our case the head organizer wants to order them asceding ASC so he orders all employes within all programs from the first employed to the last employed organizer. |  |  |

## Security Validation

**Note**: you need to test the given and not given privileges.

|  |  |  |  |
| --- | --- | --- | --- |
| **Number** | **User Name** | **Description of privilege/no privilege** | **Screenshot (query + result)** |
|  | Head\_Organizer (Privilege) | The head organizer main role is to maintain the structure of the system but has limited actions on the programs that are organized by the organizers, yet he has the access to select the program table in order to check on the program table. |  |
|  | Head\_Organizer (Non-Privilege) | In our case the head organizer isn’t allowed to insert, update or even delete any record of the program table where when he inserted the program it was denied due to the fact that it wasn’t granted him the access to do so. |  |
|  | Head\_Organizer(privilege) | The head organizer main role is to maintain the organizers where he can insert, update, delete and select to the organizer table, as shown the head organizer first inserted than updated then deleted and selected the organizer table, this grant statement mainly helped the head organizer to do all different actions on the organizer table. |  |
|  | Organizer\_ Hassan\_farsi (Privilege) | The organizer is granted the access select, update and insert into the program table since he’s responsible on the maintenance of the program and since we granted him the access to do the operations selecting the table was executed successfully. |  |
|  | Organizer\_ Hassan\_farsi (Privilege) | You might have noticed that he wasn’t allowed to delete from the program table since such a privilege could cause many effects on other tables so it wasn’t granted for the organizer, when executed the action of deleting a program it denied the operation. |  |
|  | Organizer\_ Hassan\_farsi (Privilege) | Organizers main role is the ability to insert courses and connect it to its respective program where we granted the organizer Hassan the ability to use the PROCEDURE Organizer\_AssignCourseToProgram associated with this task, this procedure helped him insert a course and connect it to its perspective program without the need to write the entire command of insertion using both the course and course\_program table and when we granted the access to a current user hassan we enhanced the security and enshured who can use this procedure. |  |
|  | Organizer\_ Hassan\_farsi (Privilege) | Organizers should be able to make different actions on the course table so in this privilege command we granted the organizer the access to select, update, delete and insert on the course table, this mainly helped the organizer to make all actions on the course table and shows our ability to grant current access to the tables for users |  |
|  | Organizer\_ Hassan\_farsi (Non-Privilege) | The organizer is only permissioned in this procedure to assign courses to programs he is reasponsible for so when he attempted to assign the inserted course to a program he isn’t permissoned to it diddn’t allow him to do so and that’s our main aim to enhance security and maintain the privileges given to the users to what each users need and allowed to do. |  |
|  | Trainer\_Jane\_ Smith | The trainer jane smith is permissioned to select the course table only in case the trainer wants to view all the courses in the system and since its granted when the trainer tried to select it the execution was done successfully |  |
|  | Trainer\_Jane\_ Smith | Yet the trainer isn’t allowed to do other than select on the course table , so attempting to update the credit\_hours in the record of the course\_ID 3 was denied since we didn’t grant the trainer the access to update the table. |  |

### 

## GUI Validation

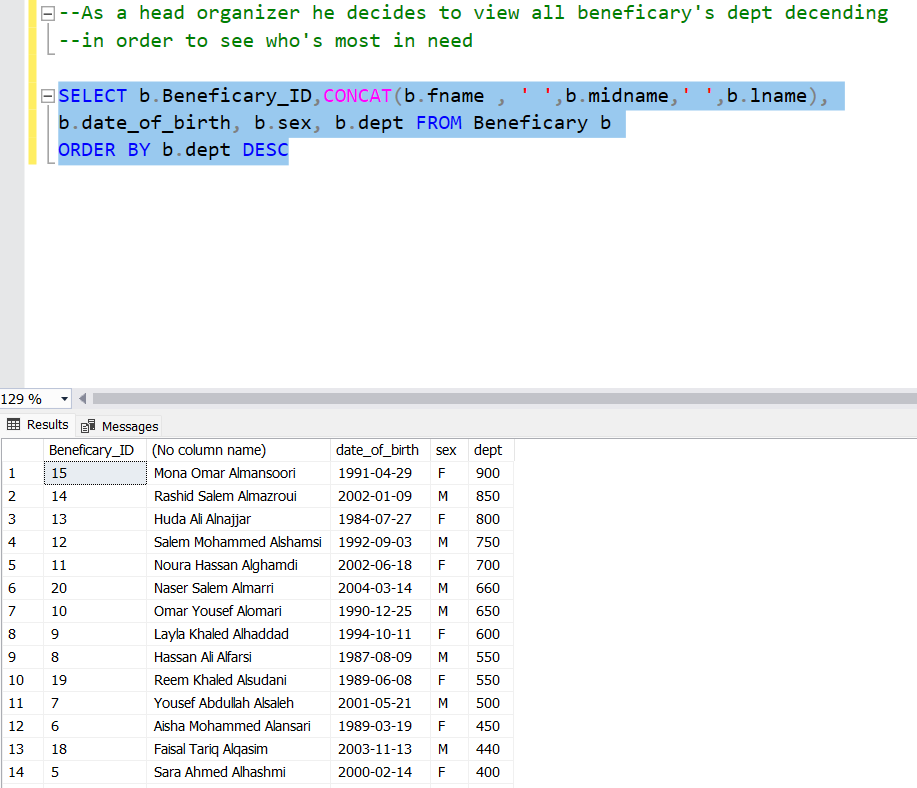
|  |  |  |
| --- | --- | --- |
| **Number** | **Description** | **screenshot** |
|  | The head organizer has access to four tables the beneficiary, organizer, program and course table he is able to select them and view all the tables records |  |
|  | The head organizer was granted the access to select all four tables and view all its records and since its granted to the head organizer to view the beneficiary table he can navigate and view all the beneficiary records. |  |
|  | He is accessed as well to view, update, delete and add records to the table where when he attempted to update a record it was successfully executed to the permissions granted to the user, we granted him such a privilege to ensure he can maintain all beneficiary’s records. |  |
|  | The head organizer is granted the access to select both the program and course table only since updating, deleting or inserting into these tables are the organizers privilege and role to do, granting him the access to view the program table was successful where when he navigated to the table he was able to view all records of the program table. |  |
|  | As mentioned the head organizer is granted the access to only select the program table where he was able to view it but when he attempted to update one of the records it gave him an error [Save failed: ('42000', "[42000] [Microsoft][ODBC SQL Server Driver][SQL Server]The UPDATE permission was denied on the object 'Program', database 'Ministry\_Final\_System1', schema 'dbo'.] which indicates that not giving him the access to do any other operation on the table was successful and indicates that outputs from the GUI are accurate. |  |

## Assess whether meaningful data has been extracted

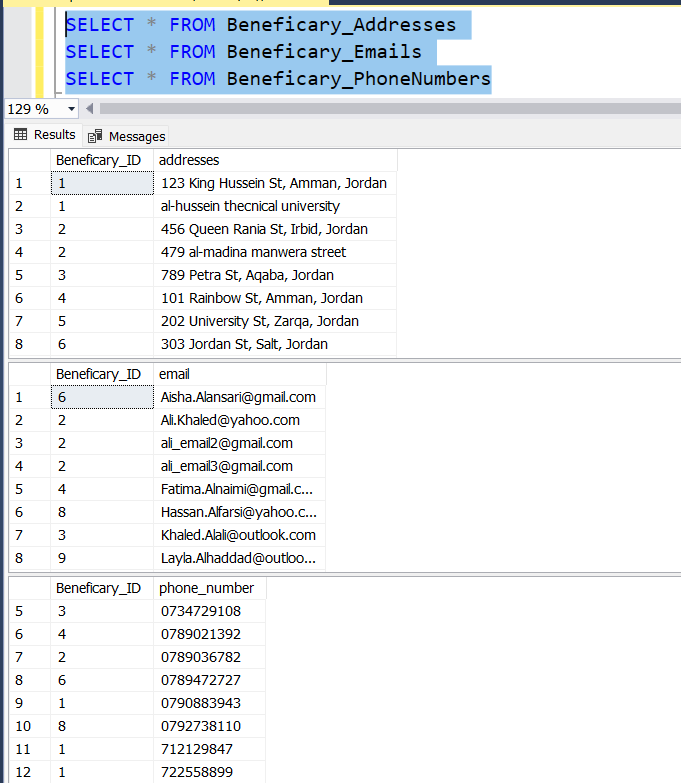
We are going to evaluate each entity tables and extract how meaningful was extracted from each table and it’s relations attitude is it accurate or not, plus demonstrate its extracted data how is it useful to our system. Then we will demonstrate our main users and there functionalities from views and procedures and all the grants for each one demonstrating a bigger understand to our system and evaluate whether our database system modifies our scenario or not. Let’s demonstrate our database system tables and all it’s relations and evaluate the general structure of our system for each entity.

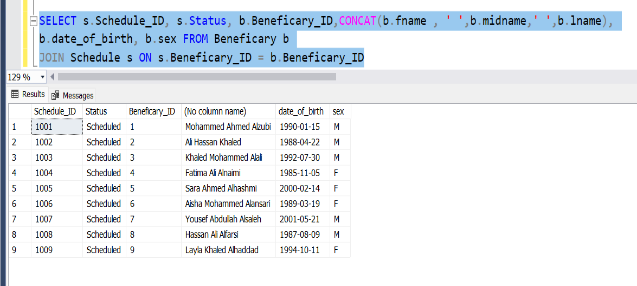
**Beneficiary**

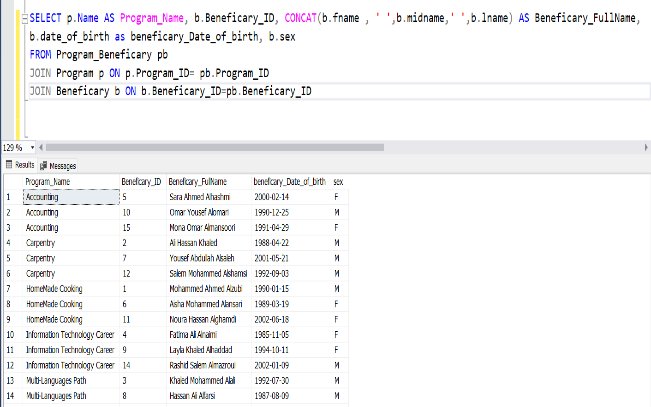
The beneficiary is a strong entity where it is the wheel of the program and courses, he registers in a program and it’s set of courses and has a schedule that demonstrates for him all his registered courses important information and its program. The beneficiary table contains important information where it’s functionally dependent on it self all attributes rely on the primary key beneficiary\_ID, this table is necessary and extracts meaningful data where you can navigate through all necessary info of the beneficiary such as the dept attribute where from it the head organizer decides the most needed beneficiary to enrol him in a program.

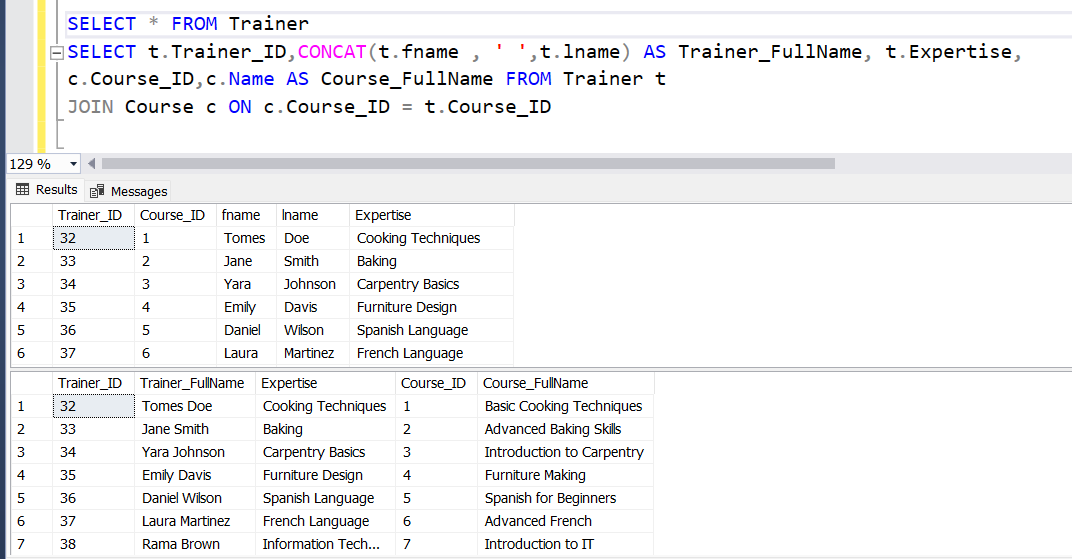
In this select statement all attribute of this table is extracted where in our case the head organizer wants to view all dept beneficiary’s ordered in descending to see the most in need to register him/her into a program, which the data was meaningful and helped the head organizer decide that mona Omar is the most in need to register in a program.

**Beneficiary’s Relations**

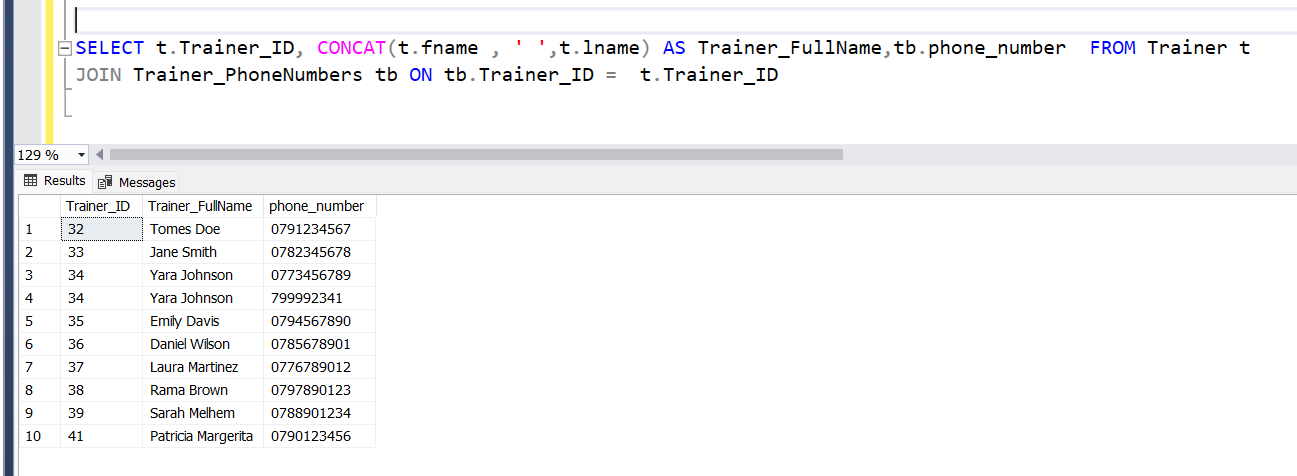
Beneficary\_Emails, Beneficary\_Addresses, Beneficary\_PhoneNumbers are all tables extracted due to multi value attribute and since it’s a multi value we conducted a different table connected with main table with a foreign key to its primary key in the main table. Emails are uniquely identified to ensure one email is registered across all the records, Addresses could have a possible case where two beneficiary’s have the same address yet not identifying it unique doesn’t state that each record in the Beneficary\_Addresses table isn’t unique where both email and the primary key are a composite primary key, and the phone numbers is uniquely identified where we don’t allow duplicate entry of phone numbers across all records within the table.

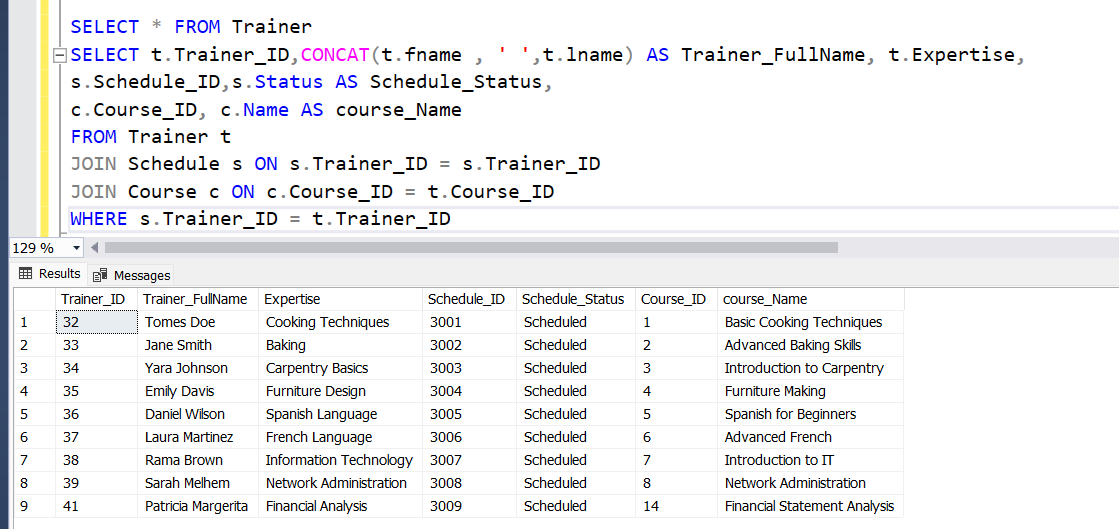
The beneficiary is connected to the schedule table with a 1:1 relation where each beneficiary has schedule this relation extracts meaningful data where when attempting to extract data that depends on the relation between them it shows that all beneficiary’s now have a schedule with it’s details, assuming we want extract all beneficiary’s schedules

The program\_Beneficary is a table that shares with beneficiary a M:M relation which is essential to ensure each beneficiary is connected to one program each program can have many beneficiary’s and each beneficiary’s should be able to register in only one program, this table is important to be able to connect both the beneficiary and the program table in different extractions of data moreover the relationship between both the beneficiary and the program maintains that many beneficiaries can register in many programs. Demonstrating this relation there’s noting better than showing all programs and there registered beneficiary.

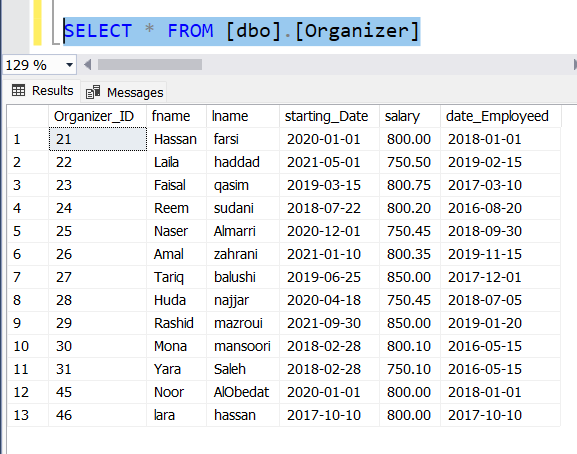
**Trainer**

This strong entity is simplified as a main user In our system where each trainer is responsible for a course which both share a 1:1 relation. Meaningful data is extracted out of this table where it’s functionally dependent on it self all attributes depend on the primary key Trainer\_ID as the course\_id is a foreign key that depends on the referential attitude of a foreign key to its origin primary key in the course table and is set in the trainer table with a null constraint which opens the ability for trainers to be registered without connecting it to the course table. Other attributes that depend it’s extraction on the primary key such as the expertise, fname and lname are set to not null since it’s necessary to register these data for each trainer.

**Trainer’s Relations**

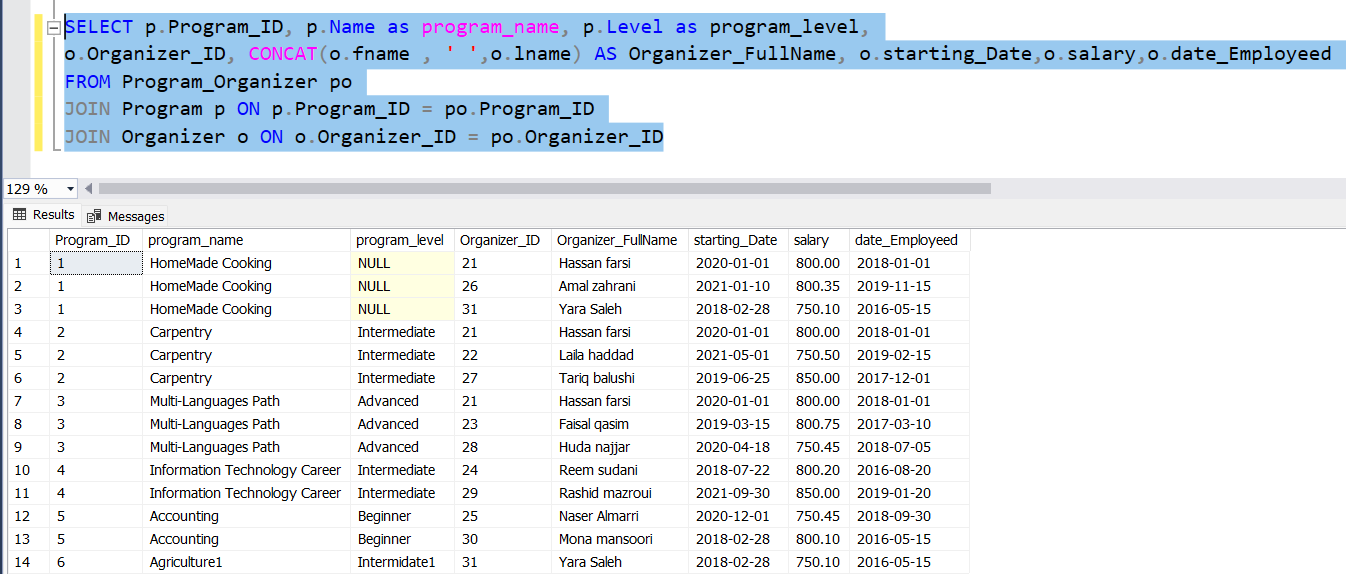
 The relations this entity is connected to are all dependent on a current way we need to maintain , for example the attribute phone number in the trainer table is a multi-attribute where each trainer can have multiple phone numbers which in order to maintain a unique record in the trainer table it was extracted to another table called trainer\_PhoneNumber and contains a composite primary key where the trainer\_ID is a foreign key as well connected to the trainer table and the phone number the other part of the primary key as unique constraint as well which indicates having one phone number across all records. Meaningful data is extracted from this relation where as shown we extracted all phone numbers of the trainers maintaining the aspect of no duplicate records attained in the trainer table. As shown in the table ID there are duplicate entries where the trainer with the ID 34 has two records where if the attribute wasn’t extracted to another table many issues will occur.

The relationship between the trainer and the schedule is 1:1 where each trainer should have a schedule that helps him to view all his registered courses where he can update its info. As shown meaningful data is extracted from the relationship between the trainer and the schedule helped us view all trainer’s schedules.

**Organizer**

This table demonstrates all the necessary info needed for the organizers and is functionally dependent on it self all attributes rely on only one primary key and this primary key is the main attribute that ensures the relations between it and other table, different constraints where implemented on non-primary attributes such as the not null constraint on fname, lname, starting Date and date employed which ensures data is inserted into these attributes, when we extracted all the data from the table meaningful data was extracted where we are able to view all records of organizers.

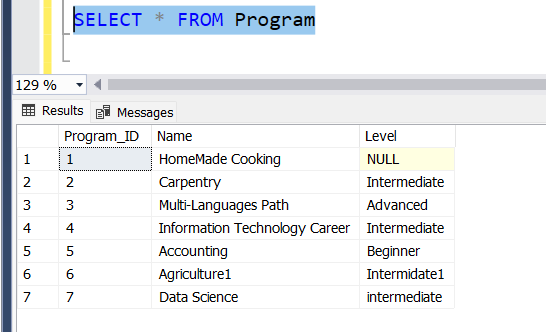
**Organizer Relations**

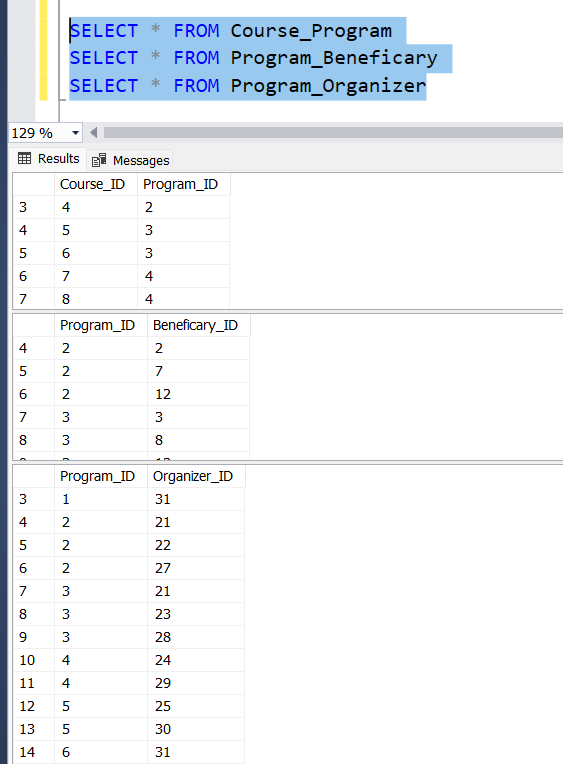
Organizer\_PhoneNumbers is a table that was extracted from the organizer table due to having a multi attribute in it, we needed to specify for the phone number a different table to ensure no duplicate records in the organizer table where there.

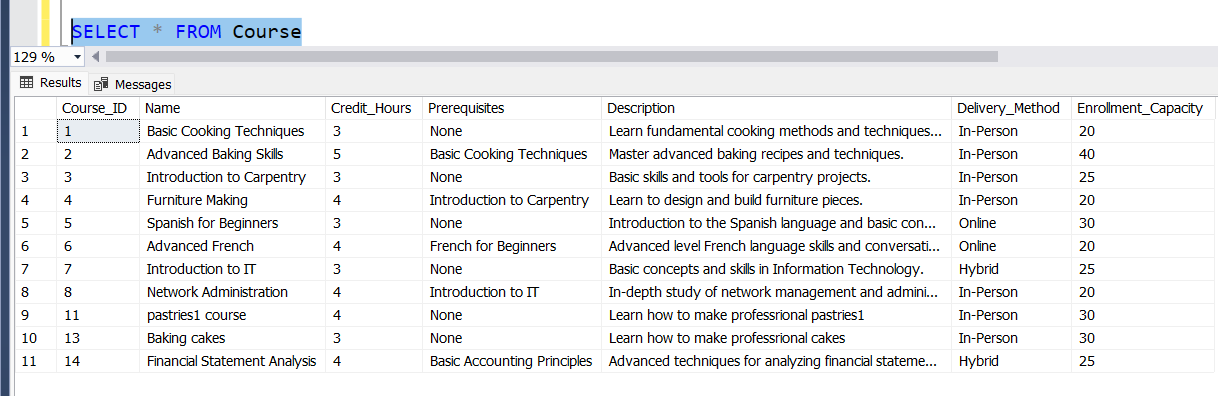
As for the program\_Organizer table it demonstrates a M:M relationship between the organizer and the program which was a main connector between the two table, from this table meaningful data was extracted where we were able to view all organizers within all programs, as shown all organizers within each table is connected to program and this table was used in many cases where we were able to extract all programs and it’s organizers using this table as the connector between the two tables using the composite primary key that each part of this key is referenced to a table.

The relationship between the organizer and the schedule is 1:1 where each organizer should have a schedule that helps each organizer view all the programs and its alternative courses, he is responsible of where from it the organizer checks all details of these programs and its courses and helps him have more clear picture of his duties. As shown in our example using the schedule table we were able to extract meaningful data that all organizers schedule in each program.

**Program**

****This table maintains the data of all programs where each record is dependent and unique with the set of data in each row. All attributes depend on the primary key which make it a totally functionally dependent. When we extracted all the data info meaningful data was extracted all records of the table and understand the overall info of each program.

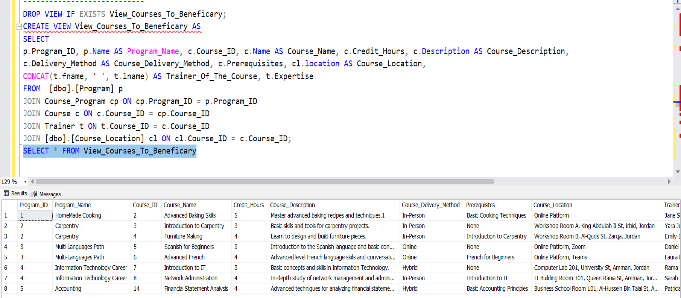
All relations connected to the program table are M:M where each reflect the connection and the normal relationship between both entities. Mentioning that each table contains only a composite primary key each part of the primary key connects the table as a foreign key as well which demonstrates a uniqueness in all records where you can’t two records with the same sat of the two primary key(The composite PK). When observing all the tables you can extract data that demonstrates each table purpose where the Program\_Course contains the program id and the course id where from it you are able to view all programs and there ID’s. The Program\_Beneficary which has the program id and the beneficiary id that are together a composite primary key from it we were able to extract all programs and there beneficiary. The Program\_Organizer that has the program id and the organizer id both a composite primary key where we were able to extract all programs and there alternative courses, being more clear each a program id could be duplicated in its row but when we say a primary composite key that clarify together attributes identify a unique record across all other records.

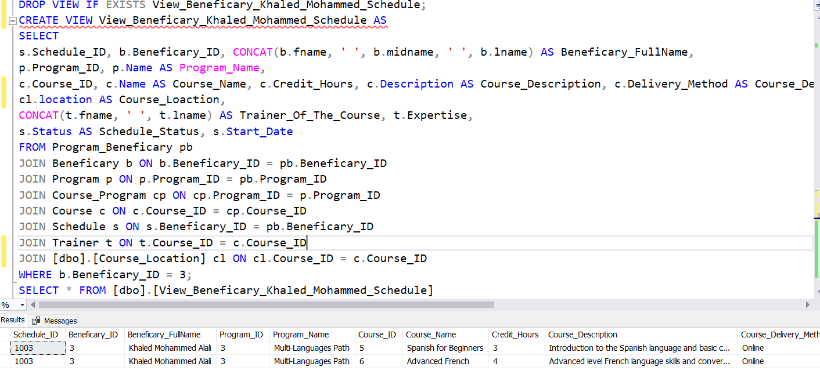
**Course**This table maintains the data of all Courses where each record is dependent and unique with the set of data in each row. All attributes depend on the primary key which make it totally functionally dependent. When we extracted all the data info meaningful data was extracted all records of the table and understand the overall info of each course.

Both these relations connected to the course table are M:M where each reflect the connection and the normal relationship between both entities. Mentioning that each table contains only a composite primary key each part of the primary key connects the table as a foreign key as well as it demonstrates a uniqueness in all records where you can’t find two records with the same sat of the two primary key(The composite PK). When observing all the tables you can extract data that demonstrates each table purpose where the Program\_Course contains the program id and the course id where from it you are able to view all programs and there respective course id. The Course\_Program demonstrates all courses and it’s respective schedule it is registered in. Clearly each course id could be duplicated in its row but when we say a primary composite key that clarify together attributes identify a unique record across all other records.  
**Schedule**

This table is the main connector of all the info of the status of the beneficiary, trainer and organizer it contains from them each its primary key and connected to its alternative foreign key in the schedule table as a null constraint where we don’t want to connect in this table the organizer , trainer and the beneficiary together our main aim is to maintain for each user a schedule info and ensure they are all registered in the system..

Our social ministry system provides beneficiary’s with the ability to register in a program with it’s associated courses where the organizer in the other hand modifies and manages the entire structure of the program and its courses and the head organizer manages the beneficiary and organizers actions, his main role is to maintain there info and ensure accurate info is stored. Lastly, the trainer that is assigned for the course can modify and update the course info he teaches .Different views and procedures where created to benefit the users main role in the system,

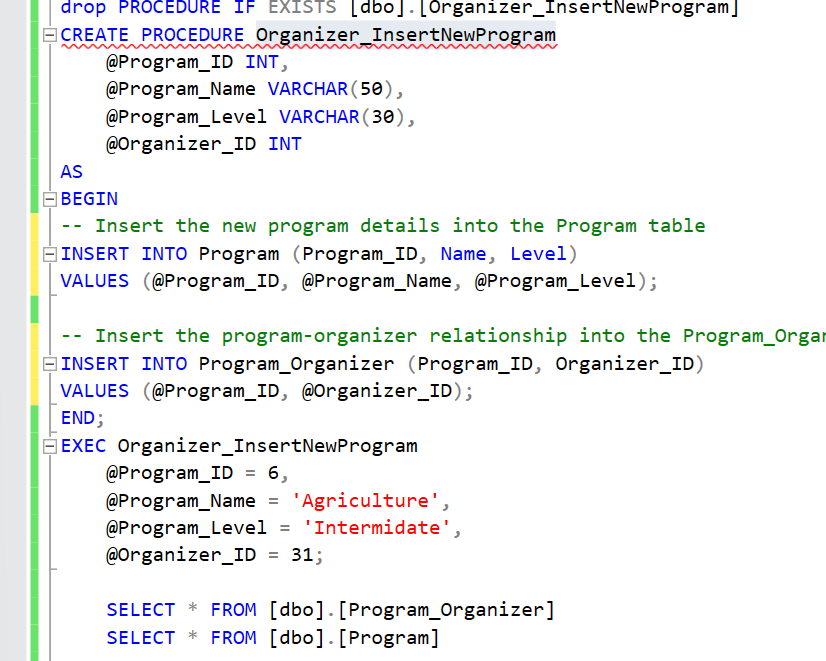
**Beneficiary**

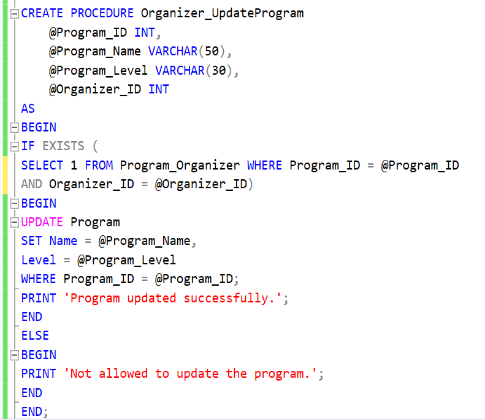
Each beneficiary has an access on two view the first one is the View\_Courses\_To\_ Beneficary that shows the user all available courses to register from in order to have a more clear vision on the overall implemented courses and see which he has a bound to register to, plus current info such as the enrolment capacity isn’t viewed to the beneficiary that’ why views are created it gives the access to show current things for the targeted view maintaining the original tables to be viewed by outsiders.

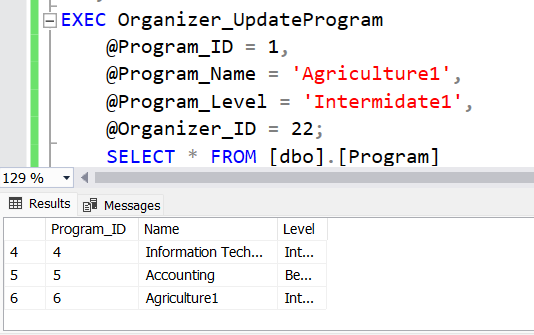
The second view we created is the View\_Beneficary\_Khaled\_Mohammed\_Schedule that was created to the specific user Khaled Mohammad where he is able to view all his registered courses such as the course location and the trainer who teaches the course in order to provide the user the ability to view them and be in contact with all his courses.

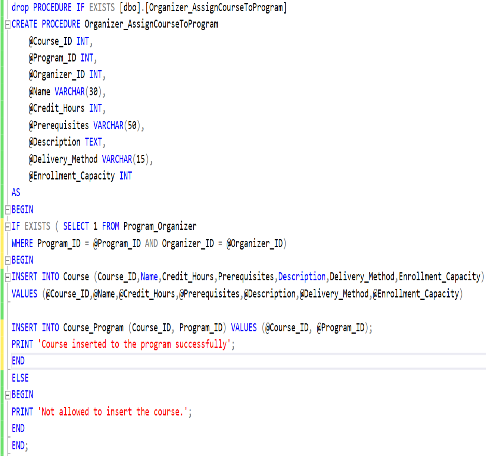
**Organizer**

The organizer main role is to be able to maintain all programs and courses info , connections and updates, plus he assigns the beneficiary’s to there respective program, where the beneficiary contacts the organizer to register him if accepted by the head organizer. Different procedures and views where created to the organizer to ease there work on them where they don’t need to keep creating query statements in order to do a current operation on to the tables, and the views to access current info by calling it just.

**Procedures**

The procedure Organizer\_InsertNewProgram works on inserting a new program with its data and ensures the inserted new program is maintained as well with the organizer id the program id the organizer has inserted in the table program\_organizer where we ensure that the program isn’t organized by no one rather it’s organized with the entered organizer and that’s by the parameters of the procedure, program\_ID and the organizer\_ID to assign them to there tables program and program\_organizer where successfully he inserts a program and assign it to the organizer to maintain it.

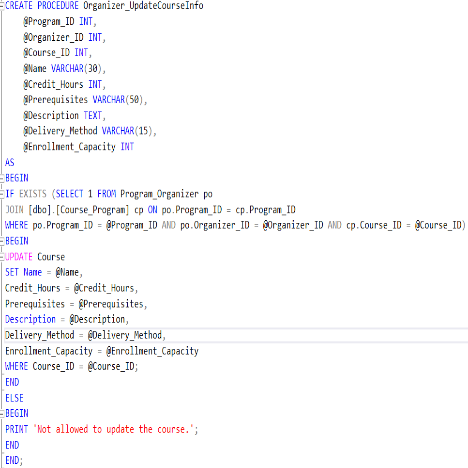
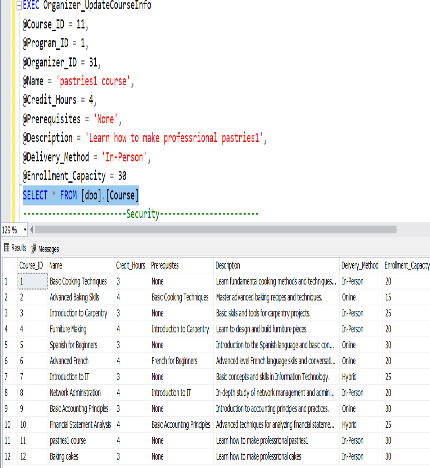
Organizer\_UpdateProgram helps the organizer to update the program by first checking whether the organizer is assigned to the program from the program\_organizer table if yes the update is executed according to the entered update values if not a print statement with not allowed to update the program is executed this helps the organizer to update the program without the need to write an entire query statement plus we ensure with the if statement that only the authorized organizers to there programs can update the program rather the procedure statement wont be executed.

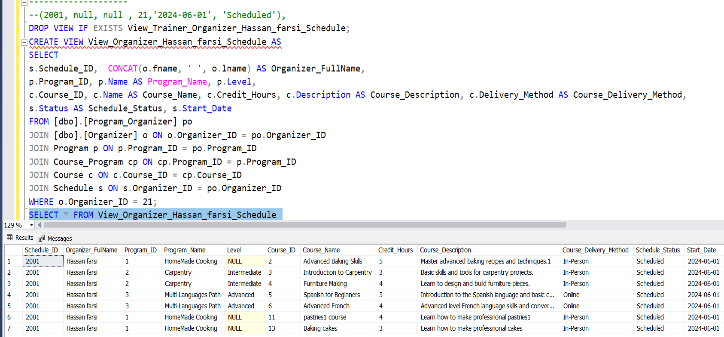


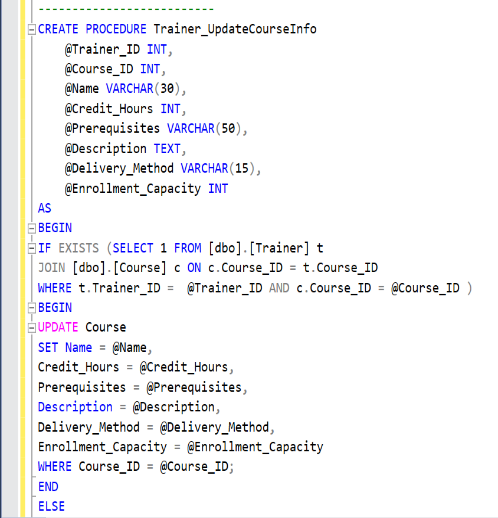


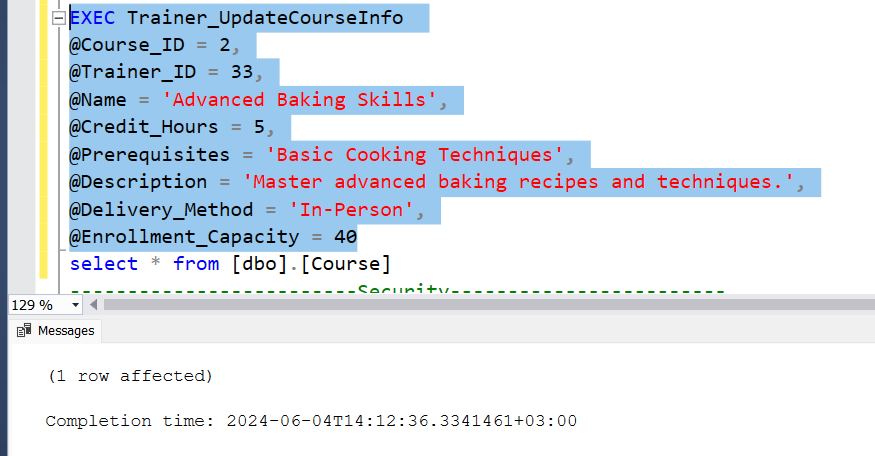
Organizer\_AssignCourseToProgram procedure helps the organizer to inserts a course and assigns it to its respective program where t first checks if the organizer is allowed to insert the course that will be connected to the program and that’s by checking wether he is authorized to maintain and modify the program of the course he s willing to insert and assign it to the program from the

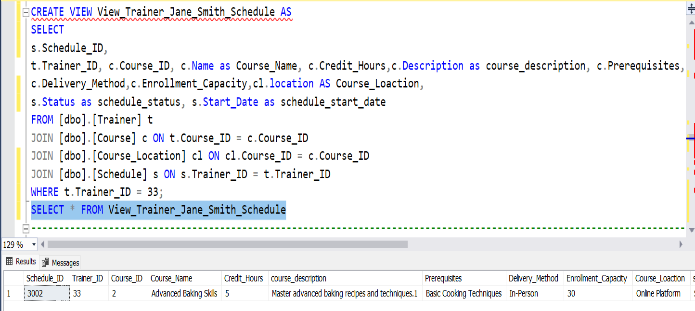
program\_organizer table then he inserts the course info into the course table after that the course is inserted to the program into the table course\_program. If the organizer isn’t connected to the program he gets a message of denying the insert into the course. This procedure mainly focus on completing the assign course to program steps in one execution helps the organizer do the entire process by just executing the command.

Organizer\_UpdateCourseInfo procedure helps the organizer update the course that he is assigned to by checking first it the course he is willing to update is connected to the program he has a privilege on from the table program\_organizer if yes he can update the course if not he will be denied to update the course, this procedure as well helps the organizer to update the course by just executing it rather then ritingg it’s entire query plus ensures he is allowed to update the course or not.

View\_Organizer\_Hassan\_ farsi\_Schedule this view helps the organizer view all the programs and its respective courses he has permission to modify in order to help him know and maintain only his respective programs and have a clear vision on his maintenance range.

**Trainer**

The trainer has the access to update the courses he is only assigned to in this procedure where it checks from the table course\_trainer if he has the course id and the trainer id are identical in the course\_trainer table he updates the course info rather a message that indicates denying the permission on updating the course info, this saves time and doesn’t need the trainer to have knowledge in query’s and is quite a logical aspect since that’s not essential.



This view helps the trainer view all his courses necessary info in order to have a clear vision on them and be able to view which courses he has access to update, this view helps in clarifying a clear path for the trainer jane smith .

## Assess the effectiveness of testing

Our testing process proved to be effective where it ensured integrity, security and the reliability of the system. By validating data, verifying outputs , enforcing security measures and testing the GUI , we identified different issues and enhanced the overall quality of our software.

**Significance of testing**

The testing was crucial for our software mostly since we are implementing a database system for a ministry validating the system operations correctly validates and ensures that system is reliable and maintains data integrity. The importance of testing arises in our system where we have persistent storage of data, and a centralized control over data , data redundancy, consistency and integrity, to ensure all these aspects we need a reliable testing process that covers validating data, verifying outputs , enforcing security measures and testing the GUI.

**Steps Followed in Testing**

* **In Data Validation** We tested the primary key in there uniqueness and nullability by attempting to insert duplicate primary keys and for nullability by trying to inert null values into primary key fields. We confirmed by these tests that the system enforces unique identifiers and does not allow null values, ensuring that each record is unique and identifiable. Foreign key validation was implemented in both the referential integrity and ON CASCADE, ON DELETE CASCADE where in the referential integrity we attempted to insert values into the foreign key that don’t exist in the related primary key table ensuring the refrencial integrity maintenance, in cascades we tested both of them by updating and deleting the primary key values verifying that related foreign keys were appropriately updated or deleted maintaining data consistency.

Unique Constraint Validation was attempted by inserting records with duplicate sets of fname, midname and lname confirming that our system enforces unique constraints for combined attributes preventing duplicate entries, Default constraint was tested by inserting records without specifying default- assigned fields ensuring default values were correctly applied, while not null attempted to insert null values into not null fields, confirming that they cannot be left blank where here we maintain data completeness. Lastly, the check constraint testing was verified by inserting invalid data into fields with check constraints ensuring only the valid and specified data to be entered. In this validation we ensured structural integrity by enforcing constraints, preventing data anomalies and maintaining consistency and accurancy.

* **In Output Validation** we validated the query through the data retrieval and data update, In data retrieval we tested the queries for retrieving data such as retrieveing the first employed organizer and calculating the max and min salries with program. These tests confirmed accurate data processing and retrieval while data update was verified by inserting new data and updating existing data reflected correctly in queries, ensuring the correctness of data operations. We verified that the system processes and retrieves data correctly, providing accurate and reliable information.
* **Security Validation** was tested in both privilege and non-privilege testing were tested the actions of different user roles, ensuring that our system enforces user-specific permissions. For instance, the head organizer could view not modify certain tables, confirming proper enforcement of access controls. Wile in the stored procedures we verified that only authorized users could execute specific stored procedures enhancing security by restricting access to critical operations. We protected the system from unauthorized access and we ensured that users perform actions with their granted permissions enhancing the overall security
* **GUI Validation** we validated the access and permissions confirming first the view and navigate for example we confirmed that the head organizer could access and view records in specified tables and the ones we allowed him to ensuring data representation. We also verified that the head organizer for example could update records in allowed tables ensuing authorized actions are permitted. Restricting actions was also tested where we attempted unauthorized updates in the GUI confirming that the system correctly blocks such actions and through that we ensured security and accuracy. Here we ensured that the user interface accurately reflect the data permissions and provide seamless and secure user experience and prevent unauthorized actions.

# Evaluation of database solution

## Effectiveness of the database solution based on user and system requirement

## Suggested improvements

## Evaluation based on improvements needed

7) References

Odogwu, C. (2022). *What Is Database Maintenance and Why Is It Important?* [online] MUO. Available at: <https://www.makeuseof.com/what-is-database-maintenance/#:~:text=Database%20maintenance%20comprises%20a%20variety>.

‌ Chapter 10 Database Maintenance. (n.d.). Available at: https://wiki.documentfoundation.org/images/3/3b/BG7210-DatabaseMaintenance.pdf [Accessed 4 Jun. 2024].

Metallic. (n.d.). *Database Backup*. [online] Available at: <https://metallic.io/knowledge-center/glossary/database-backup#:~:text=A%20database%20backup%20protects%20all>.

‌ LaRock, T. (2022). *The Importance of Database Backups: A Guide to Costs, Benefits, and Risks*. [online] Orange Matter. Available at: <https://orangematter.solarwinds.com/2022/01/11/the-cost-benefit-and-risk-of-database-backups/>.

‌ SQL Server Tutorial. (n.d.). *SQL Server Backup Types*. [online] Available at: <https://www.sqlservertutorial.net/sql-server-administration/sql-server-backup-types/>.

admin (2015). *Significance of Database Testing | The PSI*. [online] PSI. Available at: <https://www.thepsi.com/significance-of-database-testing/#:~:text=Database%20is%20a%20valuable%20asset>

[Accessed 6 Jun. 2024].

Astera. (2023). *What is Data Validation?* [online] Available at: <https://www.astera.com/knowledge-center/what-is-data-validation/#:~:text=Improves%20the%20Efficiency%20of%20Data>

[Accessed 6 Jun. 2024].

‌

‌

‌

‌