MET CS 669 Database Design and Implementation for Business
Term Project Iteration 6

Name: Priyanka Gnanasekaran

Table of Contents

Project Direction Overview

I plan to develop an application which helps with guidance for parents/guardians by tracking growth milestones for their children named as “Milestone”. This application will contain five major sections such as

1. Growth Milestone - This section will be the main profile page for handling all the input datas from the user such as age of the child, vaccination updates and so on.
2. Development Milestone - This section will have four main categories which will automatically update according to the age of the child given in the profile. The four main categories are cognitive, language, physical and emotional milestone updates.
3. Alerts - This section will contain doctor appointment alerts, reminders for vaccinations that are due for the child and (only in some cases) footnote notifications if the child is showing any signs of red flag(delays in growth).
4. Pediatric Consultations - This section will contain available online staff for attending immediate consultations and chat message options for the parents to directly contact the pediatric doctor.
5. Forum - Tips/Experience with the child can be shared with the public or group of audience using this app and interaction between them.

Brief example on someone using this application. The parent/guardian Liza installs this app and when she opens this app it prompts her to fill in the growth details of her 1 year old child Jay. After all the main inputs of Jay are accepted in the database, the development milestone section is ready with all the milestone details appropriate for the age of 1 year olds is filtered and displayed in all four milestone categories. As for the third section, Liza can update and edit alerts for further appointments and reminders. In the fourth section assuming the permission granted for accessing details with the pediatrician, the details of Jay’s pediatrician and pediatric groups will be updated for emergency contacts.

My interest in developing such an application is because I always enjoy keeping track of the growth details of my kids. As an early education teacher and my experiences during conferences with parents, they appreciate more when teachers educate children appropriately for their age and with this type of application I believe parents/guardians can have more exposure to how to bond and educate their child with ease.

Use Cases and Fields

Some of the use cases that could be helpful to understand and build this application. The first step is to build the profile is as follows:

1. *Account SignUp/Growth Milestone Use Case*
    - The user visits the webpage and installs the application.
    - Application asks for Growth details of the preferred child.
    - The user enters details and the profile for the child is created.
    - The growth chart with given details is automatically created and displayed.

Field	What it Stores	Why it's Needed
Child Name	This field stores the name of the child.	This is needed for recognizing the child account details and display name.
Birth Date	This is for the birth date	This is necessary for calculating age and will be the primary field that is required to be filled.
Pediatric Doctor	This is for medical consultation	This is for permission to access details from the medical group.
Sibling Details	This is for user to hold a joint account	This is needed for the parents with more than one child and easy access between accounts.

The next section will have four main categories which will automatically update according to the age of the child given in the profile. The four main categories are cognitive, language, physical and emotional milestone updates.

2. *Automatic Development Milestone Use Case*
    - The user visits the Development Milestone section in the app.
    - The application detects current details from the profile such as Date of Birth to calculate the age of the child
    - The application filters records age appropriately from the database and updates data for the four categories in development milestones such as cognitive, language, physical and emotional.
    - The app page also contains drop down box for desired age viewing, if the user clicks/selects a desired age to view details the page then follows step (c)

Field	What it Stores	Why it's Needed
Child Age	This confirms the calculated age.	This is needed for filtering all the 4 development milestone datas.
Desired Age	This is the desired age limit the user would like to view as well.	This is a drop list that the user might want to access to browse extra details.

The next section will contain doctor appointment alerts, reminders for vaccinations that are due for the child.

3. *Alerts Settings Use Case*
   - The user visits the Alert settings page in the app.
   - The application prompts the user to update the latest vaccine details for the child.
   - The application then detects whether the vaccines are up to date, if not the application prompts the user about the pending vaccine and why that particular vaccine is important for the child's immunity.
   - the user can accept and set the alert for next vaccination or they can ignore the prompt message.
   - The user can set alerts for next doctor appointments in the calendar based settings.

Field	**What it Stores**	**Why it's Needed**
Doctor Appointments in a calendar based setting.	This store any alarms in the calendar	This is an added feature that the users can set up for reminders.
Vaccine Name	This stores all the vaccination name details	This is necessary for updating parents for shots that might be needed for the child.
Date taken	This store dates when the last shot was taken.	This is necessary because some shots/vaccines need time before the next storage
Dosage	This stores the number of doses taken.	This is necessary to be mindful of the doses to prevent overdose cases.
Next booked vaccine	This stores the next vaccine update	This is necessary to prompt the user about the pending vaccine and why that particular vaccine is important for the child's immunity.

The next section will contain doctor appointment alerts, reminders for vaccinations that are due for the child.

4. *Pediatric Consultations Use Case*
    - The user visits the Pediatric consultation page in the app.
    - The application prompt for pediatric office contact details for the chat room set up with the child's pediatrician.
    - The user updates details of the pediatric office such as patient gateway.
    - The application accesses details and sets up contacts for the chat room.
    - The chat room will have a textbox for the contents and send button, ready to be used by the user. For example, text messages.
    - The Pediatrician can reply to the texts in the chat room, whenever time permits.

Field	**What it Stores**	**Why it's Needed**
Pediatric Doctor's Name	This stores Primary Doctors Name	This is needed for contacting the child's pediatrician.
Gateway UserName	This stores the username for the Gateway account	This is needed for accessing accounts for the chat room
Gateway Password	This stores the password for the Gateway account	This is needed for accessing accounts for the chat room
Text box	This takes the text contents from the user	This is necessary for the user and the pediatrician for their Q&A consultations
Send button	This button will have features to send the content in the text box.	This is necessary to send the content of the text box after the user is ready with his text message.

The next section is for sharing the Tips/Experience with the public or group of audience.

5. *Forum Use Case*
    - The user visits the forum in the app.
    - The application loads the latest/top 5 blogs discussed by the users/pediatricians.
    - The user can post questions or browse for tips from other parents/users.
    - If the user decides to create a post/blog, this page will contain title textbox for the title and the content text box for the content and signature text box for the author name and date. And when the user is done they hit the Post button for posting their blog.

- The user can edit their blog when the edit option is selected.
  - The application reloads all the contents stored for their particular blog.
  - When done the Post button is used to upload the changes.
- The user has a choice for deleting their blogs and comments. Restrictions are triggered if the user misuse any of these features.
- If the user decides to comment or discuss other users blogs, a text box with a post button appears to send their discussions/thoughts.

Field	What it Stores	Why it's Needed
Article Tile	This stores the topics details questioned	This is needed for the title of the article
Article Content	This stores the topics details discussed	This is needed for detailed understanding of the discussed topic.
Signature with Date	This stores the signature of the author.	This is necessary to identify the author of the post/blog
Post Button	This uploads the contents approved by the app	This is necessary to confirm that no data is lost and stored successfully in the database.
Edit Option	This feature helps to correct the previously stored content	This is necessary feature to update the blog with the current changes
Delete Option	This feature is used to delete the post/blog	This is necessary to reduce the space taken by the unwanted content in the database.
Comment TextBox	This feature stores less numbers of words when compared to article content	This is necessary for sending small/crisp messages or contents

Structural Database Rules

The first use case of the ‘Milestone’ app:

1. *Account SignUp/Growth Milestone Use Case(OLD)*
   a. The user visits the webpage and installs the application.
   b. Application asks for Growth details of the preferred child.
   c. The user enters details and the profile for the child is created.

d. The growth chart with given details is automatically created and displayed.

In this use case there are several components used, such as the user, application and the child's data stored in the database. To form the structural database rules, the child's profile created by the user becomes the first entity created on the database. The profile is the one entity found in this use case.

1. *Account SignUp/Growth Milestone Use Case(New)*
    a. The user visits the webpage and installs the application.
    b. Application asks for Growth details of the preferred child.
    c. The user enters details, the application asks for any sibling accounts such as siblings or none
    d. After the user selects, then the account is created.
    e. The growth chart with given details is automatically created and displayed.

In this use case, the new addition to the account is permission for joining siblings account links to this account. To derive a structural rule #6

6. An account can have None or Siblings

The scenario here brings a puzzle where an account holder may have siblings or may be none at all.

2. *Automatic Development Milestone Use Case*
    a. The user visits the Development Milestone section in the app.
    b. The application detects current details from the profile such as Date of Birth to calculate the age of the child
    c. The application filters records age appropriately from the database and updates data for the four categories in development milestones such as cognitive, language, physical and emotional.
    d. The app page also contains drop down box for desired age viewing, if the user clicks/selects a desired age to view details the page then follows step (c)

In this use case, the account entity filters age appropriate development milestones which are future divided into four categories. This creates some rules in the structural database.

3. *Alerts Settings Use Case*
    - The user visits the Alert settings page in the app.
    - The application prompts the user to update the latest vaccine details for the child
    - The application then detects whether the vaccines are up to date, if not the application prompts the user about the pending vaccine and why that particular vaccine is important for the child's immunity.
    - the user can accept and set the alert for next vaccination or they can ignore the prompt message.
    - The user can set alerts for next doctor appointments in the calendar based settings.

In this use case, the account saves alerts of vaccines and appointments to the database. This creates vaccine alerts and appointment alert entities. This adds one more rule to the structural database.

4. Pediatric Consultations Use Case(OLD)

- The user visits the Pediatric consultation page in the app.
- The application prompt for pediatric office contact details for the chat room set up with the child's pediatrician.
- The user updates details of the pediatric office such as patient gateway.
- The application accesses details and sets up contacts for the chat room.
- The chat room will have a textbox for the contents and send button, ready to be used by the user. For example, text messages.
- The Pediatrician can reply to the texts in the chat room, whenever time permits.

This use case also creates a similar rule as the third use case. An account can be associated with a pediatrician, but a pediatrician can be associated with one or more child accounts.

4. Pediatric Consultations Use Case(New)

- The user visits the Pediatric consultation page in the app.
- The application prompt for pediatric office contact details for the chat room set up with the child's pediatrician.
- The user updates details of the pediatric office such as patient gateway.
- The application accesses details and sets up contacts for the chat room.
- The chat room will have a textbox for the contents and send button, ready to be used by the user. For example, text messages.
- And a video call option where the patient could video chat with the pediatrician.
- The Pediatrician can reply to the texts in the chat room, whenever time permits Or visit them online for video consultations.

In this case, the individual chat room can possess both text chat and video call options. to derive the next structural rule #7

7. An account can text, video call, both or none of these.

Thus the scenario where the account holder can text chat and video call the pediatrician during emergencies. The term "none of these" make it particularly complete and the term "both" make it overlapping.

5. Forum Use Case

- The user visits the forum in the app.
- The application loads the latest/top 5 blogs discussed by the users/pediatricians.
- The user can post questions or browse for tips from other parents/users.
- If the user decides to create a post/blog, this page will contain title textbox for the title and the content text box for the content and signature text box for the author name and date. And when the user is done they hit the Post button for posting their blog.
- The user can edit their blog when the edit option is selected.

    - The application reloads all the contents stored for their particular blog.
    - When done the Post button is used to upload the changes.
- The user has a choice for deleting their blogs and comments. Restrictions are triggered if the user misuse any of these features.
- If the user decides to comment or discuss other users blogs, a text box with a post button appears to send their discussions/thoughts.

In this use case, an article blog entity is created even though they have other features associated with this entity it cannot be created as an entity in the structural database. This entity adds some rules to the structural database.

These are some of the associative structural database rules that came up with the 'Milestone' app:

1. Each *development milestone* is associated with the child's age in the *Age_Grown,* and each *Age_Grown* might be associated with one or more categories in the *development milestones*. (*Assuming if the child is over achiever*)

2. Each *account* can be associated with one or more *pediatric_vaccines*, and each *pediatric_vaccine* can be associated with one or more *accounts*.

3. Each *pediatric_vaccines* may be associated with one or more *pediatric consultants*, and each *pediatric consultant* must be associated with one *pediatric_vaccines*.

4. Each *account* may be associated with one or more *doctor appointments*, and each *doctor appointment* will be associated with one *account*.

5. Each blog in the *forum* can be associated with an *account*, and each *account* can have one or more blogs in the *forum*.

6. Each *milestone* is associated with one *development_milestone*, and each *developement_milestone* may be associated with one to many *milestones*.

7. An account will have No_Siblings or Siblings.

8. An account can chat with text, video call, both or none of these.

Conceptual Entity-Relationship Diagram

Some of the new associative structural database rules that came up with the 'Milestone' app:

1. Each *development milestone* is associated with the child's age in the *Age_Grown,* and each *Age_Grown* might be associated with one or more categories in the *development milestones*. (*Assuming if the child is over achiever*)

2. Each *account* can be associated with one or more *pediatric_vaccines*, and each *pediatric_vaccine* can be associated with one or more *accounts*.

3. Each *pediatric_vaccines* may be associated with one or more *pediatric consultants*, and each *pediatric consultant* must be associated with one *pediatric_vaccines*.

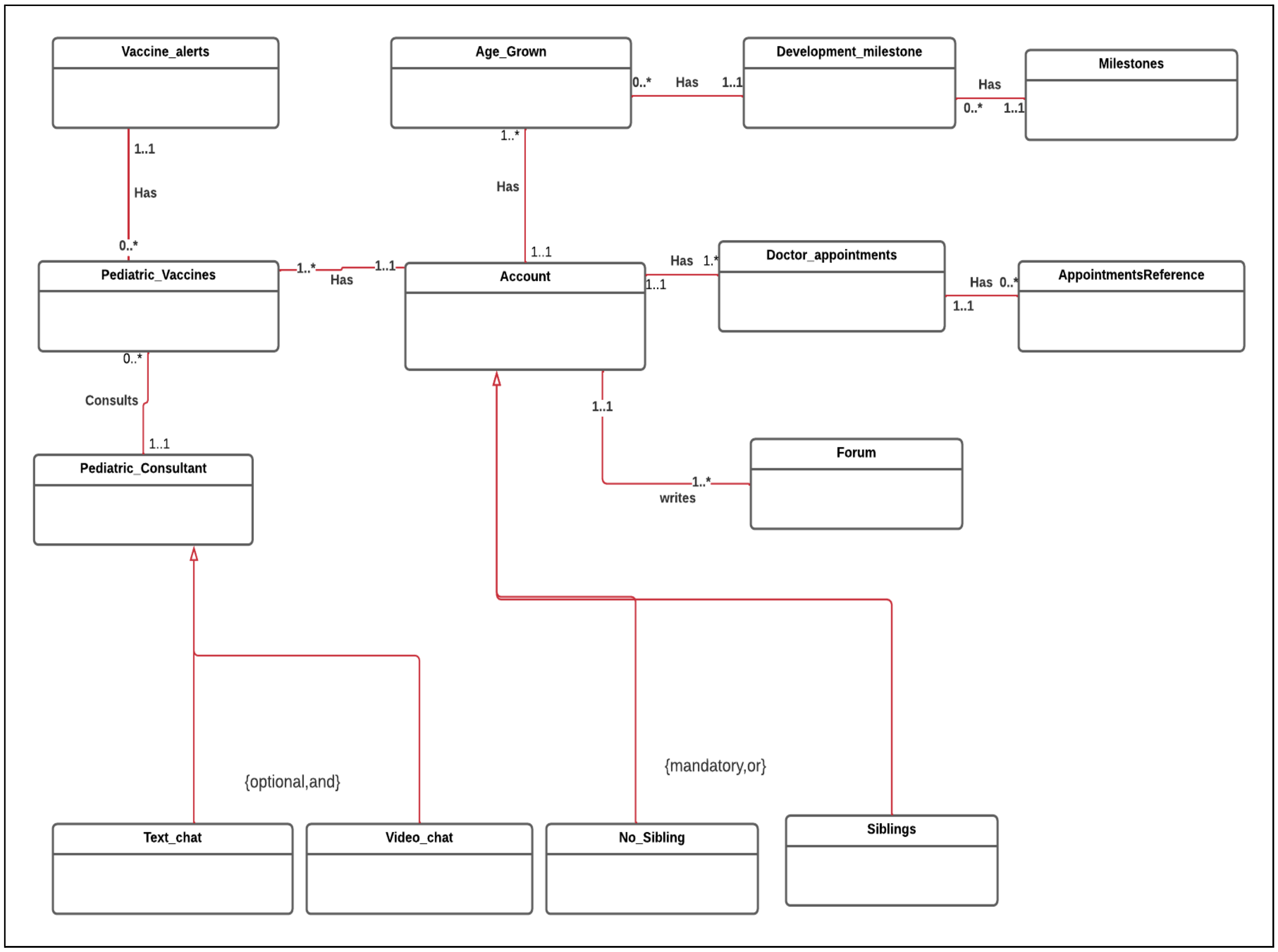
4. Each *account* may be associated with one or more *doctor appointments*, and each *doctor appointment* will be associated with one *account*.

5. Each blog in the *forum* can be associated with an *account*, and each *account* can have one or more blogs in the *forum*.

6. Each *milestone* is associated with one *development_milestone*, and each *developement_milestone* may be associated with one to many *milestones*.

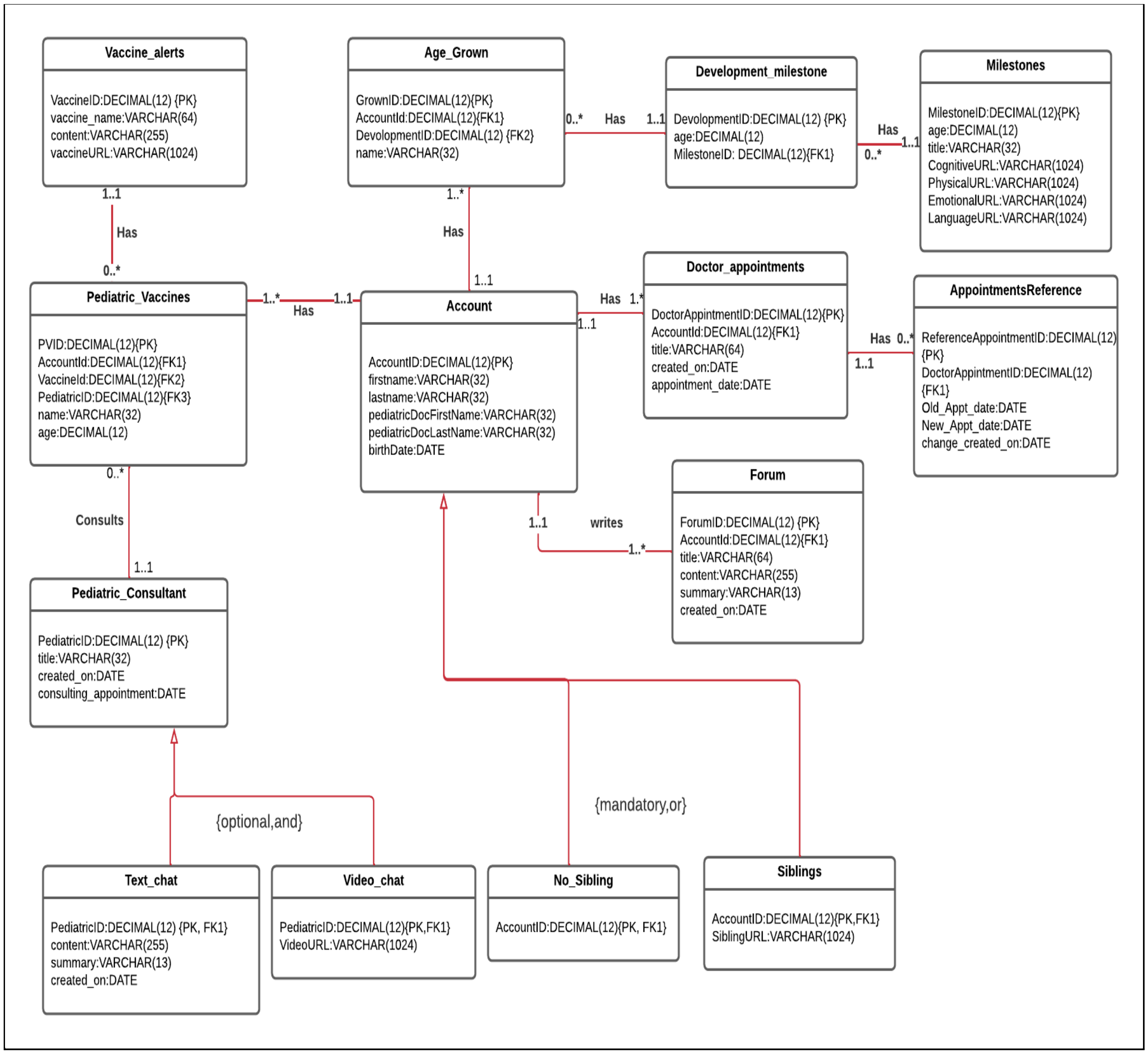
7. An account will have No_Siblings or Siblings.

8. An account can chat with text, video call, both or none of these.

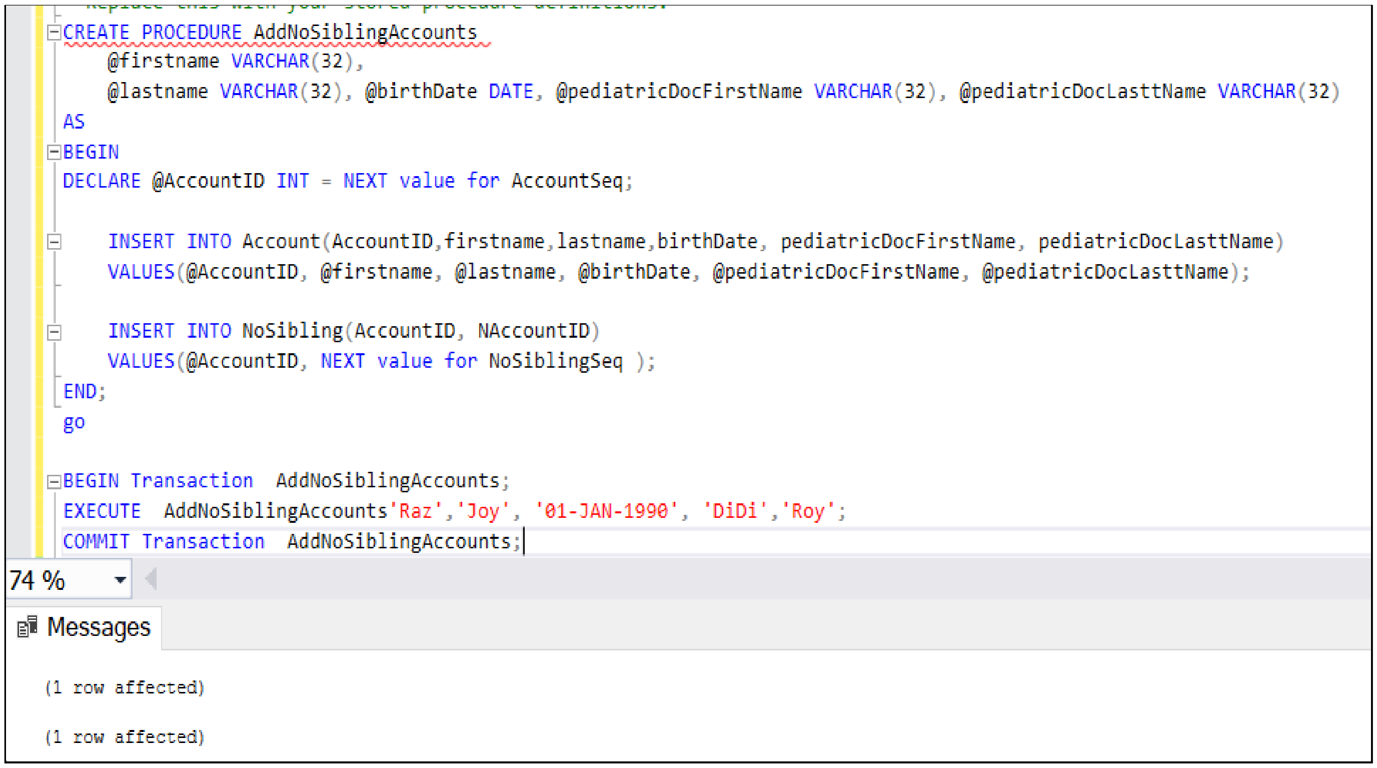


Full DBMS Physical ERD

For this iteration 6, I have introduced a new entity AppointmentsReference which helps view the history of appointment dates. This table contains old date and new date and the date when this change was created.



Stored Procedure Execution and Explanations



```
CREATE PROCEDURE AddNoSiblingAccounts
    @firstname VARCHAR(32),
    @lastname VARCHAR(32), @birthDate DATE, @pediatricDocFirstName VARCHAR(32), @pediatricDocLasttName VARCHAR(32)
AS
BEGIN
DECLARE @AccountID INT = NEXT value for AccountSeq;

    INSERT INTO Account(AccountID,firstname,lastname,birthDate, pediatricDocFirstName, pediatricDocLasttName)
    VALUES(@AccountID, @firstname, @lastname, @birthDate, @pediatricDocFirstName, @pediatricDocLasttName);

    INSERT INTO NoSibling(AccountID, NAccountID)
    VALUES(@AccountID, NEXT value for NoSiblingSeq );
END;
go

BEGIN Transaction  AddNoSiblingAccounts;
EXECUTE  AddNoSiblingAccounts'Raz','Joy', '01-JAN-1990', 'DiDi','Roy';
COMMIT Transaction  AddNoSiblingAccounts;
```

74 %

Messages

```
(1 row affected)

(1 row affected)
```

One of the procedures I created are AddNoSibblingAccounts where the Account table details from first name , last name to pediatric last name are inserted into the Account database table. And the automatically generated sequence for account ids are inserted into the NoSibling database table.

The screenshot includes the stored procedure execution where Raz Joy is the account holder's name, followed by date of birth and Didi Roy as the physician's name are stored in this statement. By executing the above executions new rows were created in the database.

Question Identification and Explanations

The Question identified and created into three queries was

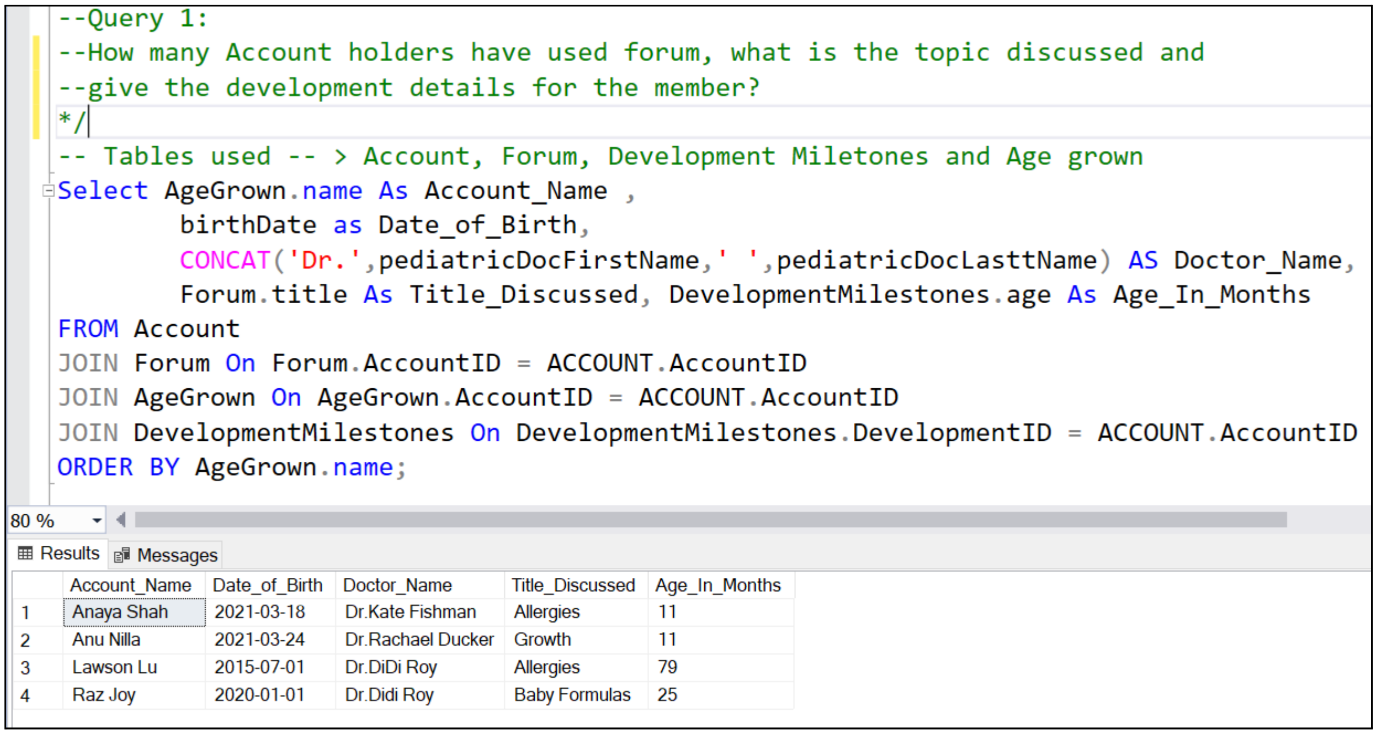
1. Query 1: How many Account holders have used the forum, what is the topic discussed and give the development details for the member?
   This question helps to analyze whether the account holders are comfortable using the app and the topics discussed are welcomed by the group or they might need topics and suggestions to be created to make the account user more interactive, informative and entertaining.

2. Query 2:Are account holders connected with their pediatrician and their sibling accounts?
   This question plays an important role because it shows the progress and usage of the app. It also holds the ability to give the overview of all the accounts with details built within the database. Answering to this question brings out all the supertypes and subtypes built in the core of the database.

3. Query 3: Filter records of children ages between 14 and 17 month old? What are their development milestones?
   This question brings out the main purpose of this application. It filters out the required age and shows them all their development milestone URL. The result from this question may give the depth of all the general milestone datas stored in the database.

Query Executions and Explanations

The Three Queries with their successful display of filtered records from the database.



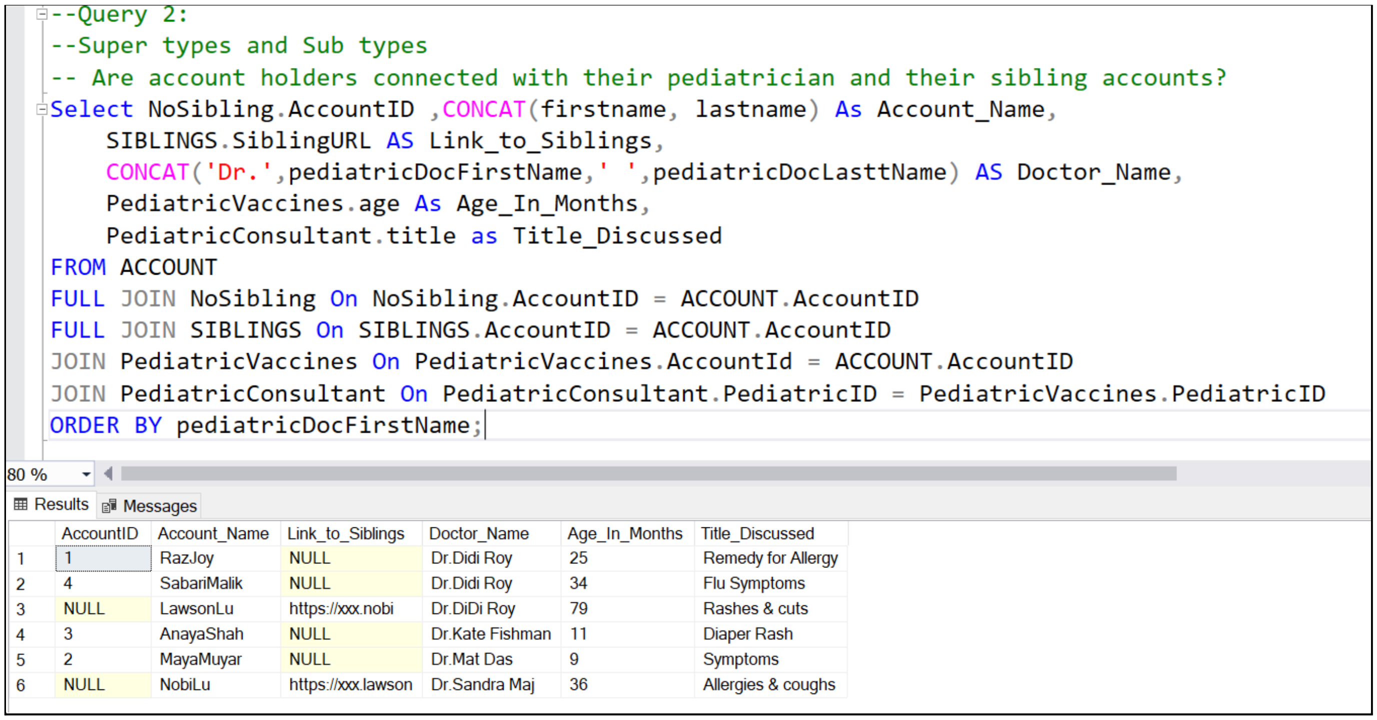
```
--Query 1:
--How many Account holders have used forum, what is the topic discussed and
--give the development details for the member?
*/
-- Tables used -- > Account, Forum, Development Miletones and Age grown
Select AgeGrown.name As Account_Name ,
        birthDate as Date_of_Birth,
        CONCAT('Dr.',pediatricDocFirstName,' ',pediatricDocLasttName) AS Doctor_Name,
        Forum.title As Title_Discussed, DevelopmentMilestones.age As Age_In_Months
FROM Account
JOIN Forum On Forum.AccountID = ACCOUNT.AccountID
JOIN AgeGrown On AgeGrown.AccountID = ACCOUNT.AccountID
JOIN DevelopmentMilestones On DevelopmentMilestones.DevelopmentID = ACCOUNT.AccountID
ORDER BY AgeGrown.name;
```

80 %

Results | Messages

	Account_Name	Date_of_Birth	Doctor_Name	Title_Discussed	Age_In_Months
1	Anaya Shah	2021-03-18	Dr.Kate Fishman	Allergies	11
2	Anu Nilla	2021-03-24	Dr.Rachael Ducker	Growth	11
3	Lawson Lu	2015-07-01	Dr.DiDi Roy	Allergies	79
4	Raz Joy	2020-01-01	Dr.Didi Roy	Baby Formulas	25

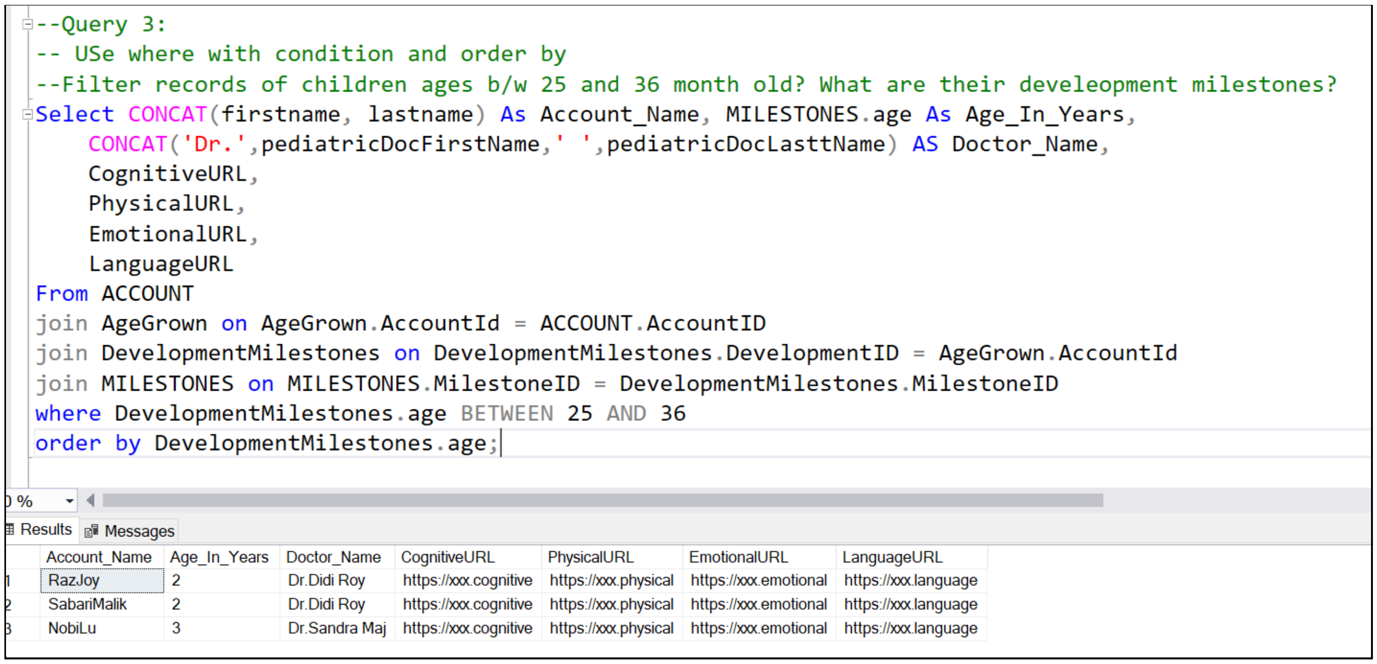
Query 1: Tables used to create this query are Account, Forum, Development Milestones and Age grown. The records were joined, filtered and ordered to display only specific records in the database.



```sql
--Query 2:
--Super types and Sub types
-- Are account holders connected with their pediatrician and their sibling accounts?
Select NoSibling.AccountID ,CONCAT(firstname, lastname) As Account_Name,
    SIBLINGS.SiblingURL AS Link_to_Siblings,
    CONCAT('Dr.',pediatricDocFirstName,' ',pediatricDocLasttName) AS Doctor_Name,
    PediatricVaccines.age As Age_In_Months,
    PediatricConsultant.title as Title_Discussed
FROM ACCOUNT
FULL JOIN NoSibling On NoSibling.AccountID = ACCOUNT.AccountID
FULL JOIN SIBLINGS On SIBLINGS.AccountID = ACCOUNT.AccountID
JOIN PediatricVaccines On PediatricVaccines.AccountId = ACCOUNT.AccountID
JOIN PediatricConsultant On PediatricConsultant.PediatricID = PediatricVaccines.PediatricID
ORDER BY pediatricDocFirstName;
```

	AccountID	Account_Name	Link_to_Siblings	Doctor_Name	Age_In_Months	Title_Discussed
1	1	RazJoy	NULL	Dr.Didi Roy	25	Remedy for Allergy
2	4	SabariMalik	NULL	Dr.Didi Roy	34	Flu Symptoms
3	NULL	LawsonLu	https://xxx.nobi	Dr.DiDi Roy	79	Rashes & cuts
4	3	AnayaShah	NULL	Dr.Kate Fishman	11	Diaper Rash
5	2	MayaMuyar	NULL	Dr.Mat Das	9	Symptoms
6	NULL	NobiLu	https://xxx.lawson	Dr.Sandra Maj	36	Allergies & coughs

Query 2: Tables used are Super types and subtypes such as Account as Super type with Siblings and NoSibling as subtypes. Since not all the data is stored in the database, I have FULL join to bring out all the records and also their progress of data stored in all the tables. To connect Pediatric Consult I have used the Pediatric vaccines table which helps act as the foreign key link between the Account and Pediatric Consultant super types.



```sql
--Query 3:
-- USe where with condition and order by
--Filter records of children ages b/w 25 and 36 month old? What are their develeopment milestones?
Select CONCAT(firstname, lastname) As Account_Name, MILESTONES.age As Age_In_Years,
    CONCAT('Dr.',pediatricDocFirstName,' ',pediatricDocLasttName) AS Doctor_Name,
    CognitiveURL,
    PhysicalURL,
    EmotionalURL,
    LanguageURL
From ACCOUNT
join AgeGrown on AgeGrown.AccountId = ACCOUNT.AccountID
join DevelopmentMilestones on DevelopmentMilestones.DevelopmentID = AgeGrown.AccountId
join MILESTONES on MILESTONES.MilestoneID = DevelopmentMilestones.MilestoneID
where DevelopmentMilestones.age BETWEEN 25 AND 36
order by DevelopmentMilestones.age;
```

	Account_Name	Age_In_Years	Doctor_Name	CognitiveURL	PhysicalURL	EmotionalURL	LanguageURL
1	RazJoy	2	Dr.Didi Roy	https://xxx.cognitive	https://xxx.physical	https://xxx.emotional	https://xxx.language
2	SabariMalik	2	Dr.Didi Roy	https://xxx.cognitive	https://xxx.physical	https://xxx.emotional	https://xxx.language
3	NobiLu	3	Dr.Sandra Maj	https://xxx.cognitive	https://xxx.physical	https://xxx.emotional	https://xxx.language

Query 3: In this query I have joined four tables and filtered a required age group with their three specific development milestones. I have used the where clause to maintain the condition of the age required. The order by used was not much used because of the less input given to the database.

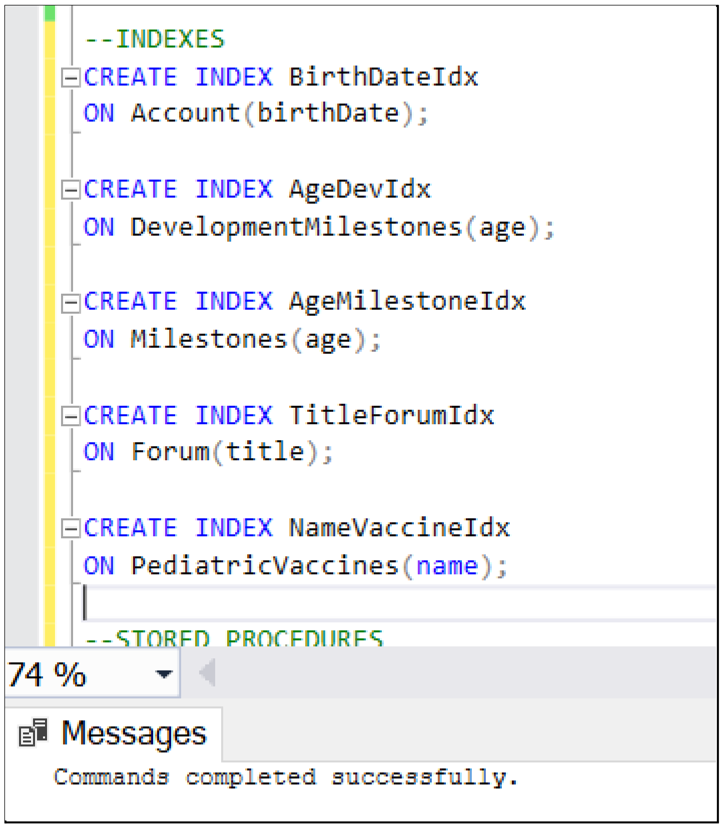
Index Identification and Creations

As for the primary keys which are already indexed:

1. Account.AccountID
2. NoSibling.NAccountID
3. Siblings.SaccountId
4. Forum.ForumID = accountID FK
5. DoctorAppointments.SoctorAppointmentID
6. DevelopmentMilestones.DevelopmentID
7. Milestones.MilestoneID = DevelopmentID FK
8. AgeGrown.GrownID
9. PediatricConsultant.PediatricID
10. TextChat.TextID = pediatricID FK
11. VideoChat.VideoID
12. PediatricVaccines.PVID
13. VaccineAlerts.VaccineID

As for foriegn Keys

Column	**Unique**	**Description**
AccountID	Not Unique	They are used for four tables as a way for linking all the base details like forum
VaccineID	Not Unique	They are used for linking pediatric consultant with vaccine alerts
DevelopmentID	Not unique	They are used for linking age with four category milestones
PediatricID	Not unique	They are used for linking the pediatric consultant with text and video chats



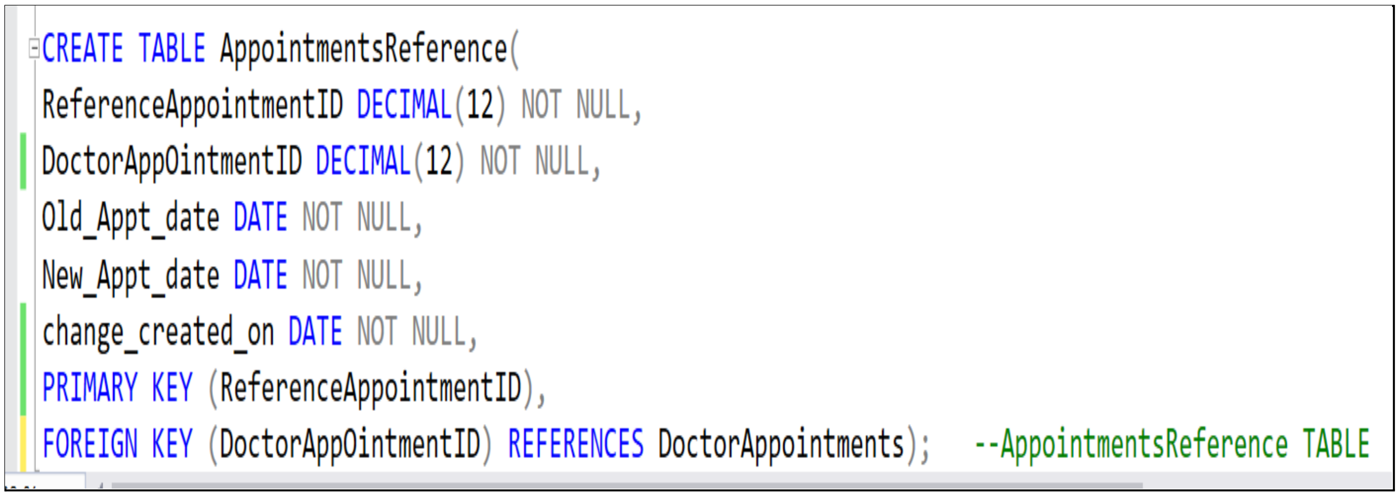
```
--INDEXES
CREATE INDEX BirthDateIdx
ON Account(birthDate);

CREATE INDEX AgeDevIdx
ON DevelopmentMilestones(age);

CREATE INDEX AgeMilestoneIdx
ON Milestones(age);

CREATE INDEX TitleForumIdx
ON Forum(title);

CREATE INDEX NameVaccineIdx
ON PediatricVaccines(name);

--STORED PROCEDURES
```

```
Commands completed successfully.
```

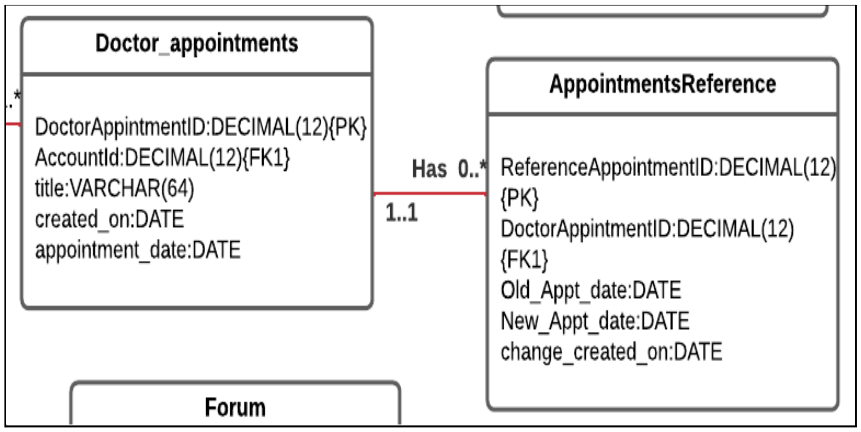
I have created five new indexes such as birth date index for Account, age for development milestones and milestone tables , title for the topic discussed in forum and name of vaccines from the pediatric vaccines could give way for new queries in the future.

History Table Demonstration

The history table I created is called AppointmentsReference. Here is the screenshot of the table.



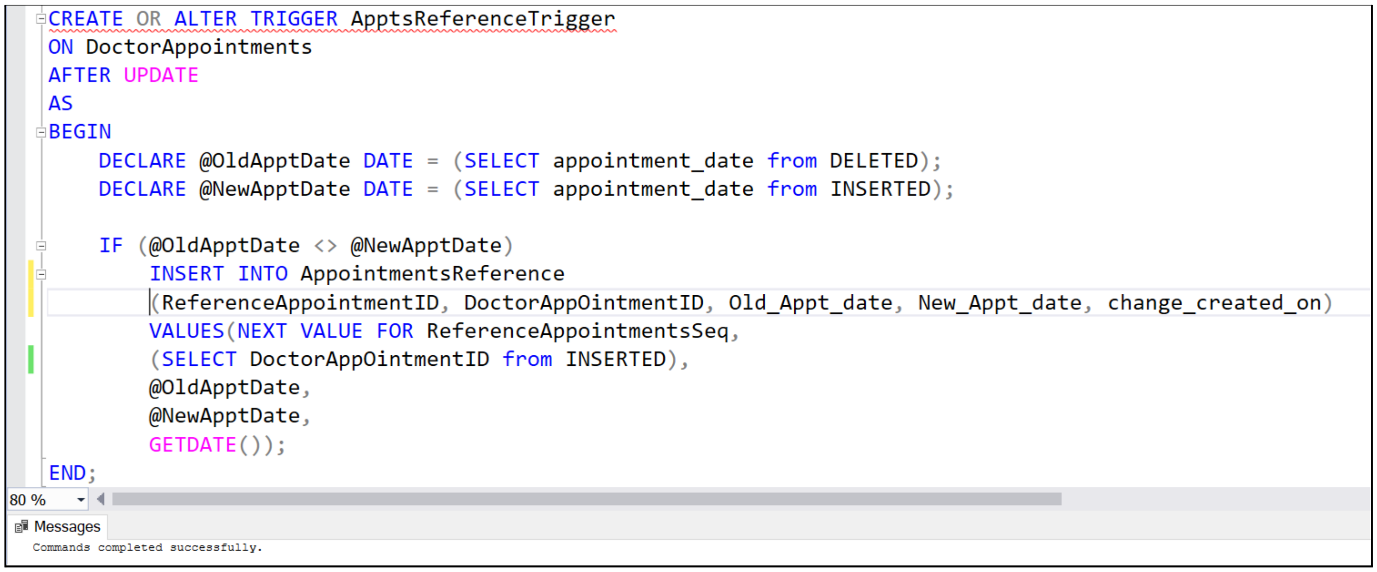
```
CREATE TABLE AppointmentsReference(
ReferenceAppointmentID DECIMAL(12) NOT NULL,
DoctorAppOintmentID DECIMAL(12) NOT NULL,
Old_Appt_date DATE NOT NULL,
New_Appt_date DATE NOT NULL,
change_created_on DATE NOT NULL,
PRIMARY KEY (ReferenceAppointmentID),
FOREIGN KEY (DoctorAppOintmentID) REFERENCES DoctorAppointments);   --AppointmentsReference TABLE
```



Below are the attributes I added

Attributes	**Description**
ReferenceAppointmentID	This is the primary key for this Reference table.
DoctorAppointmentID	This acts as the foreign key and helps keep the records connected.
Old_Apt_Date	This contains the old date that was previously stored in the DoctorAppointments table before the update.
New_Apt_Date	This contains the new updated date of the doctors appointments.
change_created_on	This contains the updated date. it helps keep track of the day when the update was done.

Here is the screenshot of the trigger



```
CREATE OR ALTER TRIGGER ApptsReferenceTrigger
ON DoctorAppointments
AFTER UPDATE
AS
BEGIN
    DECLARE @OldApptDate DATE = (SELECT appointment_date from DELETED);
    DECLARE @NewApptDate DATE = (SELECT appointment_date from INSERTED);

    IF (@OldApptDate <> @NewApptDate)
        INSERT INTO AppointmentsReference
        (ReferenceAppointmentID, DoctorAppOintmentID, Old_Appt_date, New_Appt_date, change_created_on)
        VALUES(NEXT VALUE FOR ReferenceAppointmentsSeq,
        (SELECT DoctorAppOintmentID from INSERTED),
        @OldApptDate,
        @NewApptDate,
        GETDATE());
END;
```

```
Commands completed successfully.
```

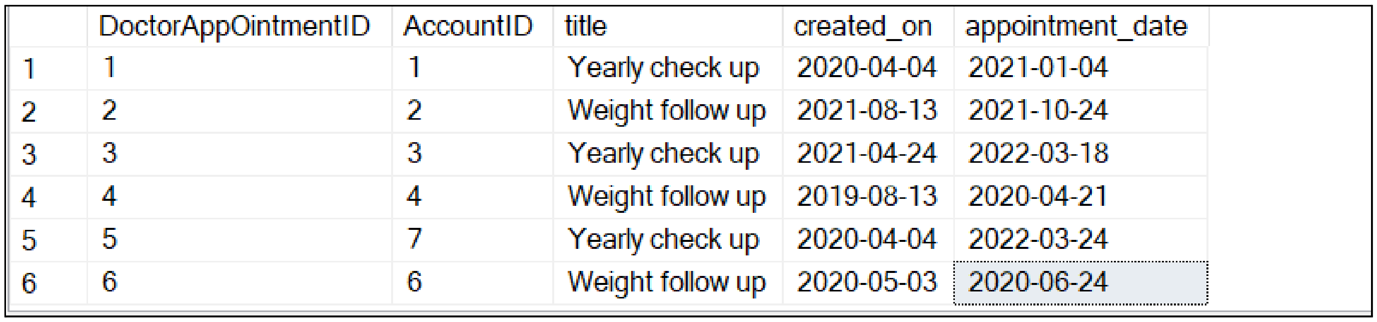
Here is how it works

Code	**Description**
CREATE OR ALTER TRIGGER ApptsReferenceTrigger ON DoctorAppointments	The trigger ApptsReferenceTrigger has been created on the DoctorAppointments table.
AFTER UPDATE AS BEGIN	After the update statement is executed, this trigger begins to take action.

DECLARE @OldApptDate DATE = (SELECT appointment_date from DELETED);
DECLARE @NewApptDate DATE = (SELECT appointment_date from INSERTED);
	This saves the record of dates from the DELETED and INSERTED tables.

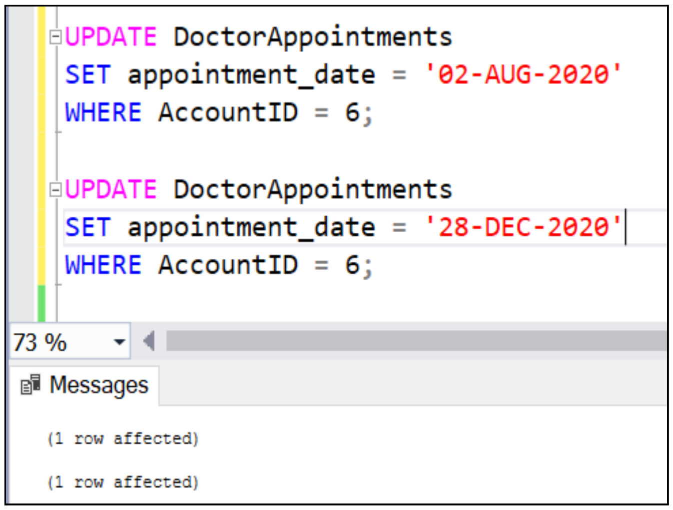
IF (@OldApptDate <> @NewApptDate)	the IF statements have a condition where the old date and the new date have to be different.
INSERT INTO AppointmentsReference VALUES (NEXT VALUE FOR ReferenceAppointmentsSeq, (SELECT DoctorAppOintmentID from INSERTED), @OldApptDate, @NewApptDate, GETDATE());	This insert statement inserted the next value of the Reference sequence, links the DoctorAppointementID for the INSERTED pseudo table with the declared old date, new dates. Lastly the created dates been inserted.
END;	This brings the trigger block to an end.

Before updating the Doctor Appointment table. Please note the 4th and 6th record appointment date.



	DoctorAppOintmentID	AccountID	title	created_on	appointment_date
1	1	1	Yearly check up	2020-04-04	2021-01-04
2	2	2	Weight follow up	2021-08-13	2021-10-24
3	3	3	Yearly check up	2021-04-24	2022-03-18
4	4	4	Weight follow up	2019-08-13	2020-04-21
5	5	7	Yearly check up	2020-04-04	2022-03-24
6	6	6	Weight follow up	2020-05-03	2020-06-24

Here are the update statements with the resulted AppointmentsReference table

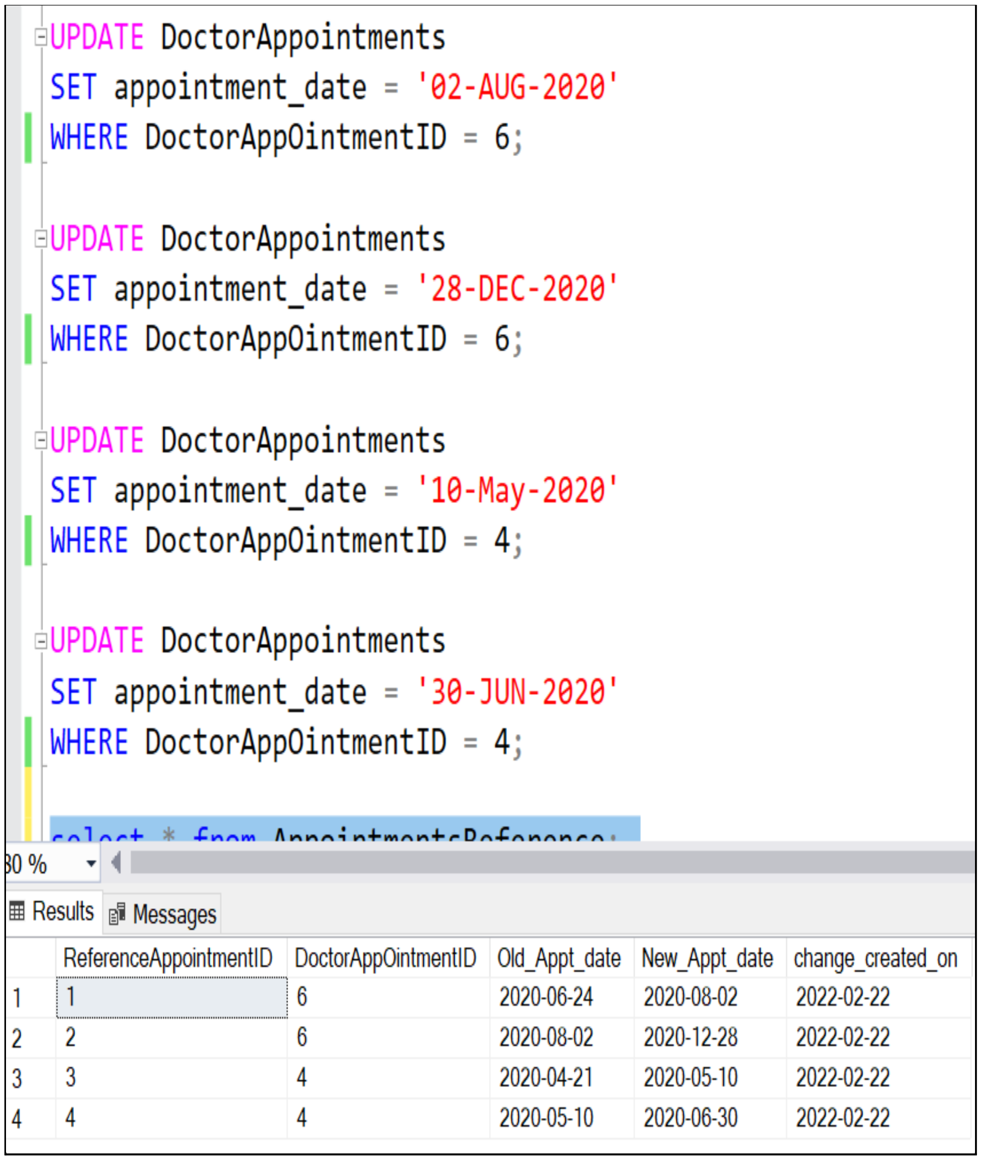


```
UPDATE DoctorAppointments
SET appointment_date = '02-AUG-2020'
WHERE AccountID = 6;

UPDATE DoctorAppointments
SET appointment_date = '28-DEC-2020'
WHERE AccountID = 6;
```

73 %

Messages

```
(1 row affected)

(1 row affected)
```



```
UPDATE DoctorAppointments
SET appointment_date = '02-AUG-2020'
WHERE DoctorAppOintmentID = 6;

UPDATE DoctorAppointments
SET appointment_date = '28-DEC-2020'
WHERE DoctorAppOintmentID = 6;

UPDATE DoctorAppointments
SET appointment_date = '10-May-2020'
WHERE DoctorAppOintmentID = 4;

UPDATE DoctorAppointments
SET appointment_date = '30-JUN-2020'
WHERE DoctorAppOintmentID = 4;

select * from AppointmentsReference;
```

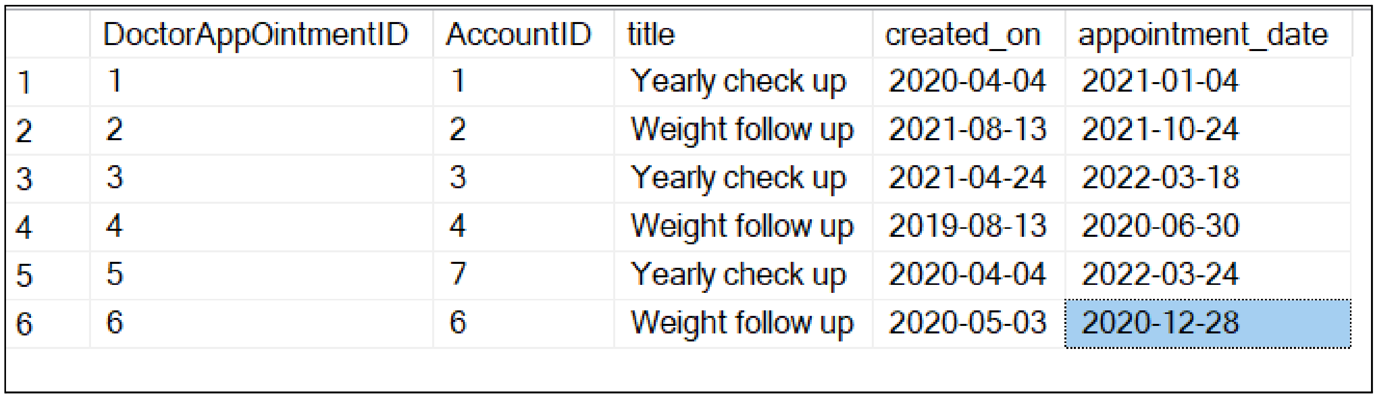
80 %

Results Messages

	ReferenceAppointmentID	DoctorAppOintmentID	Old_Appt_date	New_Appt_date	change_created_on
1	1	6	2020-06-24	2020-08-02	2022-02-22
2	2	6	2020-08-02	2020-12-28	2022-02-22
3	3	4	2020-04-21	2020-05-10	2022-02-22
4	4	4	2020-05-10	2020-06-30	2022-02-22

As you can notice, the first row shows the record of the 6th DoctorAppointmentId, the old date with the update new date and the date it's created on.

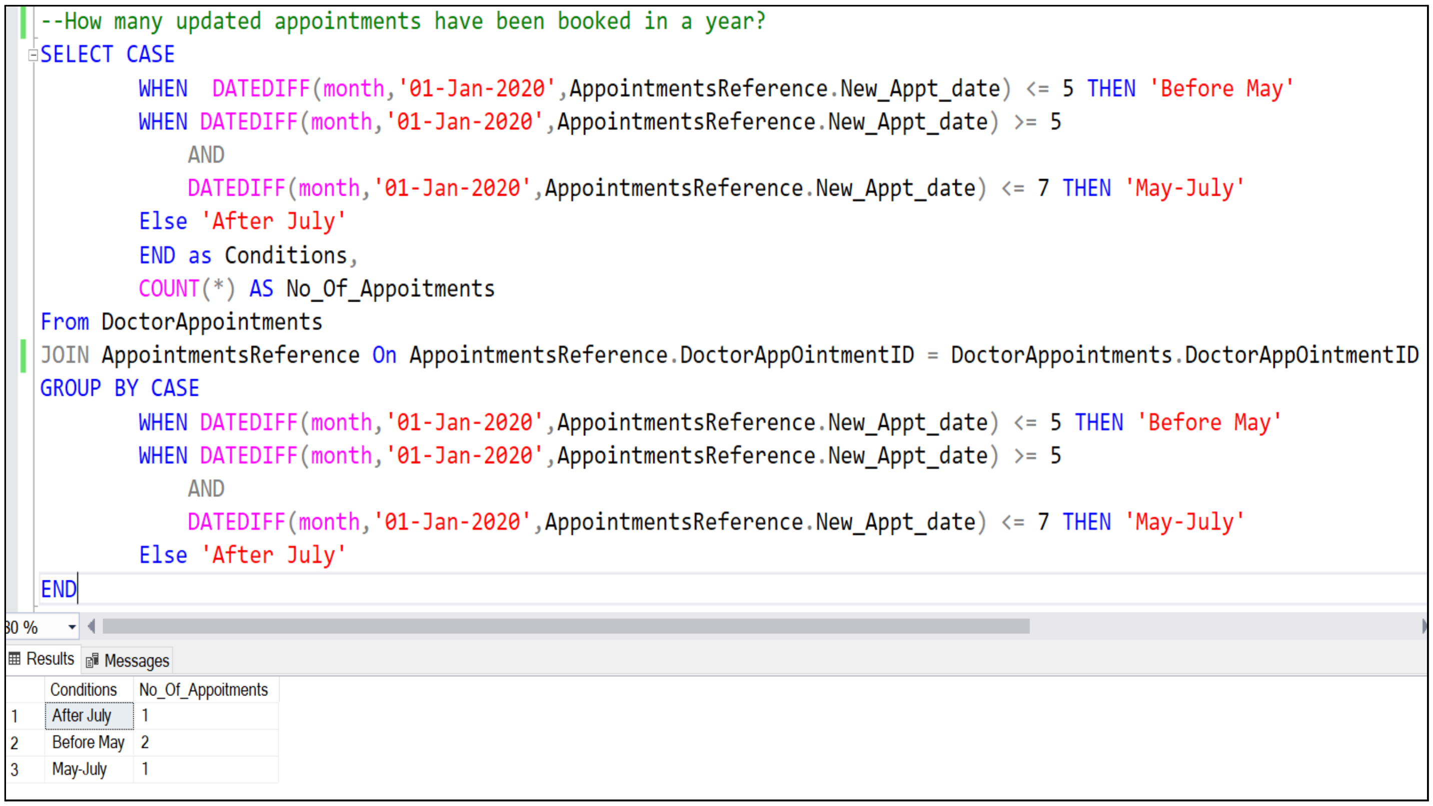
After updating here is the Doctor Appointment table. Note the 4th and 6th record appointment date has been updated successfully.



	DoctorAppOintmentID	AccountID	title	created_on	appointment_date
1	1	1	Yearly check up	2020-04-04	2021-01-04
2	2	2	Weight follow up	2021-08-13	2021-10-24
3	3	3	Yearly check up	2021-04-24	2022-03-18
4	4	4	Weight follow up	2019-08-13	2020-06-30
5	5	7	Yearly check up	2020-04-04	2022-03-24
6	6	6	Weight follow up	2020-05-03	2020-12-28

Some of the useful queries that can demonstrate how the history table can be used.

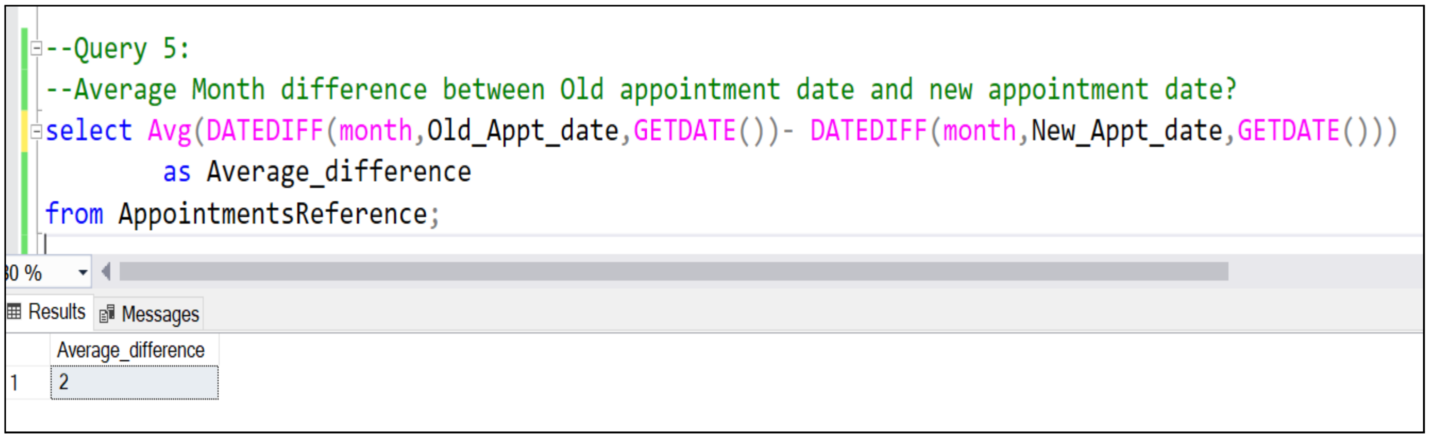
Query 4: In a year, How many appointments have been updated. In this query, I have placed three conditions in a year such as Before May, from May till July and After May. This will help determine the most popular months and helps notice how often the appointments are booked for the child.



```
--How many updated appointments have been booked in a year?
SELECT CASE
        WHEN  DATEDIFF(month,'01-Jan-2020',AppointmentsReference.New_Appt_date) <= 5 THEN 'Before May'
        WHEN DATEDIFF(month,'01-Jan-2020',AppointmentsReference.New_Appt_date) >= 5
            AND
            DATEDIFF(month,'01-Jan-2020',AppointmentsReference.New_Appt_date) <= 7 THEN 'May-July'
        Else 'After July'
        END as Conditions,
        COUNT(*) AS No_Of_Appoitments
From DoctorAppointments
JOIN AppointmentsReference On AppointmentsReference.DoctorAppOintmentID = DoctorAppointments.DoctorAppOintmentID
GROUP BY CASE
        WHEN DATEDIFF(month,'01-Jan-2020',AppointmentsReference.New_Appt_date) <= 5 THEN 'Before May'
        WHEN DATEDIFF(month,'01-Jan-2020',AppointmentsReference.New_Appt_date) >= 5
            AND
            DATEDIFF(month,'01-Jan-2020',AppointmentsReference.New_Appt_date) <= 7 THEN 'May-July'
        Else 'After July'
END
```

	Conditions	No_Of_Appoitments
1	After July	1
2	Before May	2
3	May-July	1

Query 5: In this query, the average difference between the two appointments has been noted. This helps keep track of the account holders' frequent visits and helps set up reminders for them.



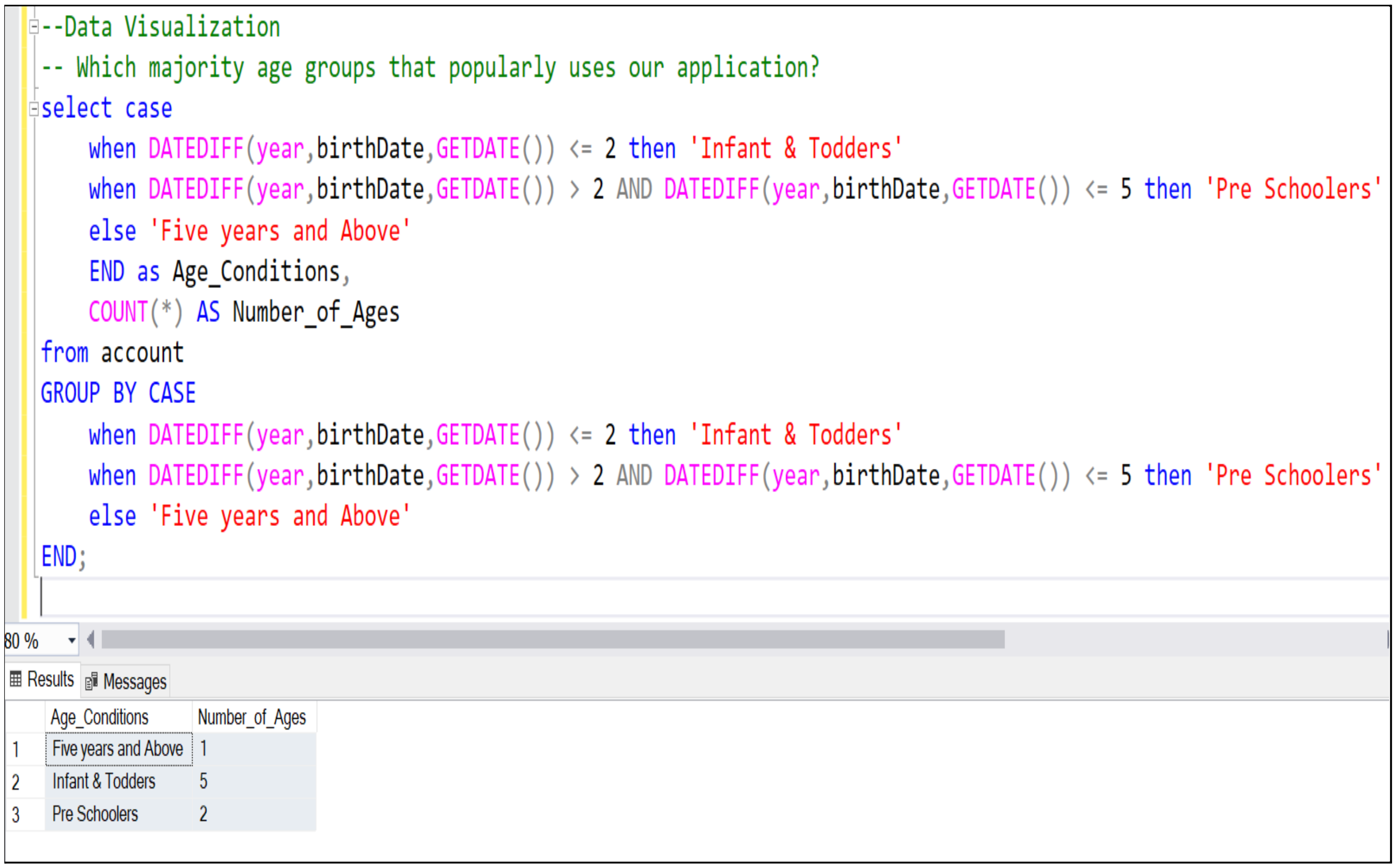
```
--Query 5:
--Average Month difference between Old appointment date and new appointment date?
select Avg(DATEDIFF(month,Old_Appt_date,GETDATE())- DATEDIFF(month,New_Appt_date,GETDATE()))
        as Average_difference
from AppointmentsReference;
```

	Average_difference
1	2

Data Visualizations

One of the major goals when a product is released is to visualize, analyze and improve the application to keep the product most popular and efficient with lots of new features to explore. To achieve this we need to know more about our customers' interests that could help us strive towards this goal. And it also gives us a clear view of what kind of engaging activities that we could introduce to them for their attention. While analyzing this fact, one of the basic queries was to find which age group has been interested to learn more about growth.

To answer this query, I developed this select case query that counts all the records with their age ranges that are actively and currently using our application.



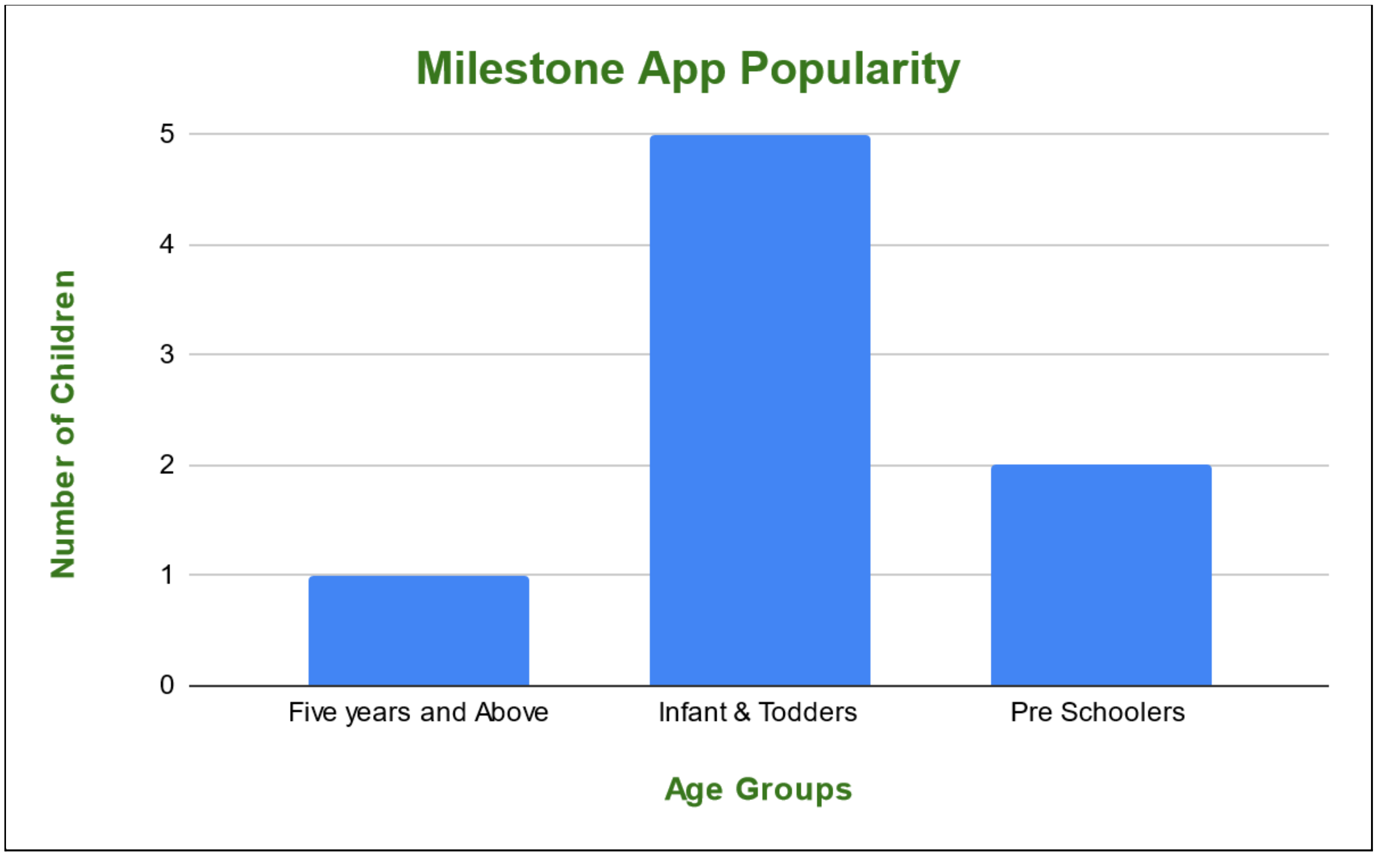
```
--Data Visualization
-- Which majority age groups that popularly uses our application?
select case
    when DATEDIFF(year,birthDate,GETDATE()) <= 2 then 'Infant & Todders'
    when DATEDIFF(year,birthDate,GETDATE()) > 2 AND DATEDIFF(year,birthDate,GETDATE()) <= 5 then 'Pre Schoolers'
    else 'Five years and Above'
    END as Age_Conditions,
    COUNT(*) AS Number_of_Ages
from account
GROUP BY CASE
    when DATEDIFF(year,birthDate,GETDATE()) <= 2 then 'Infant & Todders'
    when DATEDIFF(year,birthDate,GETDATE()) > 2 AND DATEDIFF(year,birthDate,GETDATE()) <= 5 then 'Pre Schoolers'
    else 'Five years and Above'
END;
```

80 %

Results | Messages

	Age_Conditions	Number_of_Ages
1	Five years and Above	1
2	Infant & Todders	5
3	Pre Schoolers	2

Since the filtered data is very brief, I planned to use the bar chart which would help visualize our results. After filtering the data from the query, I created the bar chart by exporting the content into the CSV file.



The First content we learn from this chart is that there are three targeted age groups. The highest bar is the Infant and Toddler age group with the highest number of children enrolled for activities and updates in our application. The next popular age group that are using our app is the preschoolers, this also shows that just by introducing some more activities we could increase more enrollments. Finally the lowest age group is the Five years and above, I would like to introduce more features to welcome this age group as well.

Summary and Reflection

My database project is the Milestone app which tracks growth of the child, filters development milestones for appropriate age for the parents to browse, gives notification for doctor appointments and due vaccines for the child. And also a section where parents/ guardians can have online consultation/chat messages with pediatricians and last but not least it has forums to read and write articles/blogs and interactions between users.

For this iteration, I rebuilt the database and introduced a total of six stored procedures with inserters. After reviewing and analyzing the queries, I have a total of five queries with one trigger and updates for the history table. Lastly, the filtered query for data visualization I have kept in the end of the sql file.

In this document, I have reviewed and updated the physical ERD and completed the contents that are needed for this week. Thank you for your support. I learnt a lot from this project.