Advanced Database

Assignment 1

TA: Rehame Ibrahim

NAME: 1-MENNA ELMINSHAWY 20217011
2-AHMAD WAEL ABDELAZIZ 20216016
3- AHMED KHALED AHMED 20216004
4- JANA MOHAMED NAYEF 20216129
5-RADWA BELAL ABDALLAH 20217005

Group ID: DS2.

DB Source Name: Anime dataset.

a) Questions

1. What is the anime name and the maximum episode number recorded in the "Anime_Properties" table?

```
SELECT [AnimeName], [EpisodeNumber] AS 'Max number of Episode'
FROM [dbo].[Anime_Properties]
WHERE [EpisodeNumber] = (SELECT MAX([EpisodeNumber]) FROM [dbo].[Anime_Properties]);
```

Results | Messages

AnimeName	Max number of Episode
Oyako Club	1818

2. How many distinct anime types are there in the "Anime_Properties" table?

```
SELECT COUNT(DISTINCT animeType) AS DistinctAnimeTypeCount
FROM dbo.Anime_Properties;
```

Results | Messages

DistinctAnimeTypeCount
6

3. What are the counts of each anime type, ordered by the number of occurrences?

```
SELECT animeType, COUNT(*) AS AnimeTypeCount
FROM dbo.Anime_Properties
GROUP BY animeType
ORDER BY AnimeTypeCount DESC;
```

Results | Messages

animeType	AnimeTypeCount
TV	3787
OVA	3311
Movie	2348
Special	1676
ONA	659
Music	488
NULL	25

4. Which users have given a rating of 10 in the "Rating" table, and what are their corresponding ratings?

```
13  select[UserID], ([ratingUser])
14  from[dbo].[Rating]
15  where [ratingUser]=10
```

Results | Messages

	UserID	ratingUser
1	2213	10
2	2213	10
3	2216	10
4	2216	10
5	2216	10
6	2216	10
7	2217	10
8	2217	10
9	2217	10
10	2218	10
11	2218	10
12	2221	10
13	2221	10
14	2221	10
15	2221	10
16	2221	10
17	2224	10
18	2225	10

5. How many ratings of 10 are there in the "Rating" table?

```
17
18  select count([ratingUser]) As 'Number of Rate 10'
19  from[dbo].[Rating]
20  where [ratingUser]=10
```

Results | Messages

Number of Rate 10
341496

6. What is the count of ratings given by each user in the "Rating" table, ordered by the user ID?

```
22  SELECT [ratingUser], COUNT(*) AS RatingUserCount
23  FROM [dbo].[Rating]
24  GROUP BY [ratingUser]
25  ORDER BY [ratingUser] ;
```

110 %

Results | Messages

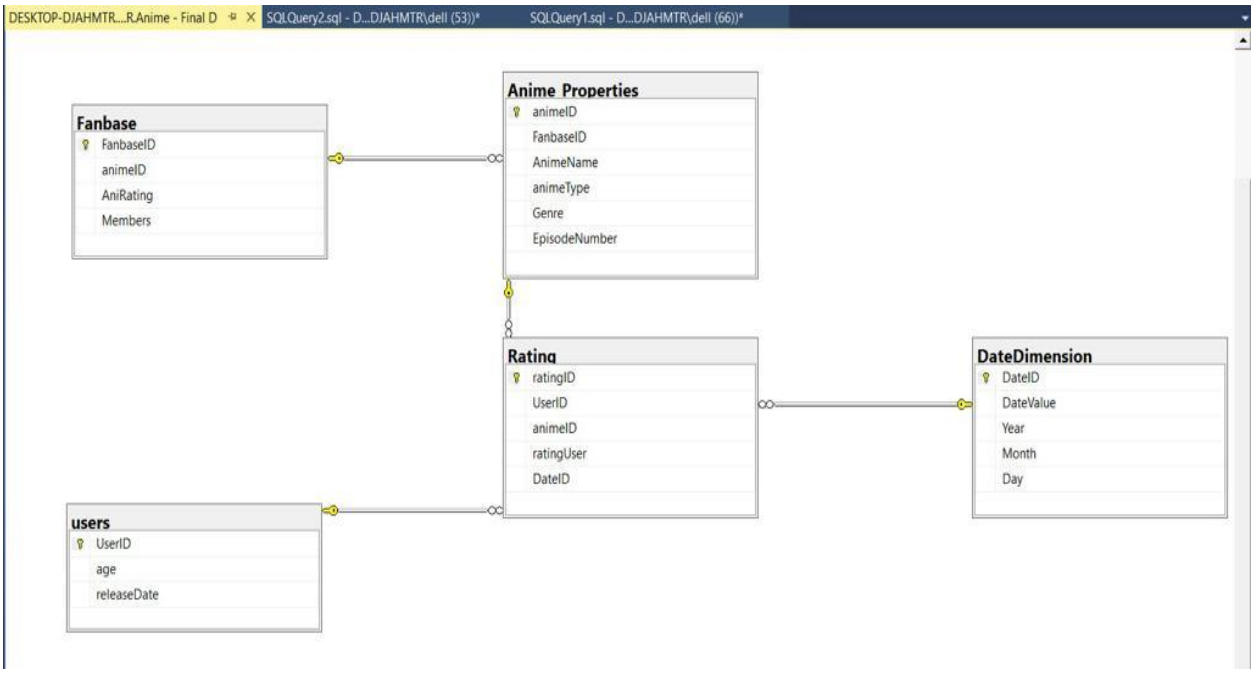
	ratingUser	RatingUserCount
1	-1	482175
2	1	5460
3	2	7588
4	3	13152
5	4	35300
6	5	96826
7	6	210430
8	7	448010
9	8	523206
10	9	414622
11	10	341496

b) we created a snowflake schema that has 3 dimensions, and one fact table as follows:

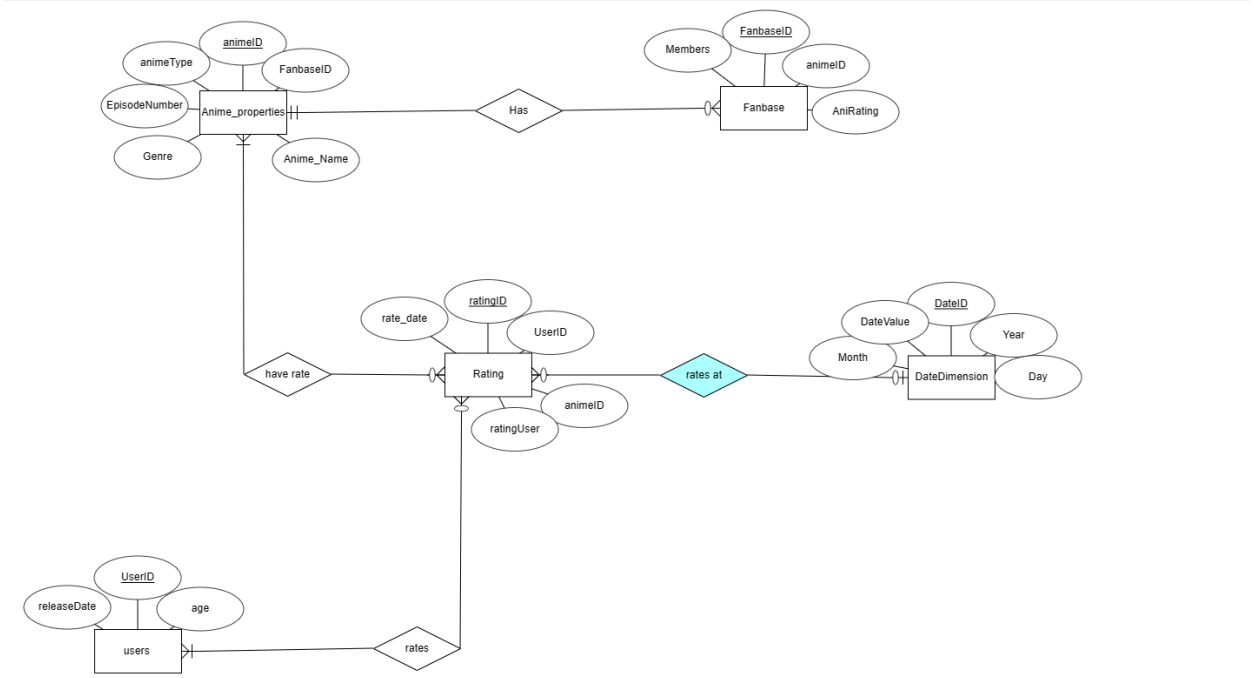
(User, Anime properties, Fanbase) those represents the tables of our dimensions, (Rating) this one represents our fact table (our measure) that we are concerned about analyzing it.

c) we are creating the rating snowflake schema (for anime) to analyze the avg rating of each anime to determine the most popular ones and the most popular genres so that we can use these results in recommending different anime to users according to its ratings which makes it easier to them to find what to watch. We are also trying to analyze the ratio between viewers who rate shows/movies after watching them and the viewers who don't.

d) the following screenshot shows what ‘s our snowflake schema looks like:



The ERD model:



e)

Dimension	Explanation
User Dimension	Contains information about users who rated an anime.
	- UserID: Primary Key
	- Age: Age of the user
	- ReleaseDate: Date when the user rated an anime
Anime Properties	Contains details about anime properties.
Dimension	- AnimeID: Primary Key
	- AnimeName: Name of the anime
	- AnimeType: Type of anime (show, movie, OVA)
	- Genre: Genre of the anime
	- EpisodeNumber: Number of episodes
	- FanbaseID: Foreign Key to Fanbase Dimension
Fanbase Dimension	Represents fanbases for each anime, forming a hierarchy.
	- FanbaseID: Primary Key
	- AnimeRating: Rating given to the anime
	- Members: Number of members in the fanbase
Rating Fact Table	Contains information about anime ratings given by users.
	- RatingID: Primary Key
	- UserID: Foreign Key to User Dimension
	- AnimeID: Foreign Key to Anime Properties Dimension
	- RatingUser: User's rating for the anime
	- RatingOverall: Overall rating (if applicable)
	- DateID: Foreign Key to Date Dimension
Date Dimension	Contains date-related information.
	- DateID: Primary Key
	- DateValue: Actual date
	- Year: Year component of the date
	- Month: Month component of the date
	- Day: Day component of the date

1)First, we create our tables:

```
CREATE TABLE Anime_Properties (
   animeID INT PRIMARY KEY, -- Retained animeID as the primary
key
   FanbaseID INT,  -- Added FanbaseID as a foreign key
   AnimeName TEXT,
   animeType CHAR(10),
   Genre TEXT,
   EpisodeNumber INT,
   FOREIGN KEY (FanbaseID) REFERENCES Fanbase(FanbaseID)
);
```

```
CREATE TABLE users (
   UserID INT PRIMARY KEY,
   age INT,
   releaseDate DATE
);
```

```
-- Create tables with corrected structure ( dimensions first)
CREATE TABLE Fanbase (
   FanbaseID INT PRIMARY KEY IDENTITY(1,1), -- Added FanbaseID as
the primary key
   animeID INT,  -- Retaining animeID for reference
   AniRating FLOAT,
   Members INT
);
```

```
-----Creating the fact table
CREATE TABLE Rating (
   ratingID INT PRIMARY KEY IDENTITY(1,1),
   UserID INT,
   animeID INT,
   ratingUser INT,
   DateID INT,
   FOREIGN KEY (DateID) REFERENCES DateDimension(DateID),
   FOREIGN KEY (UserID) REFERENCES users(UserID),
   FOREIGN KEY (animeID) REFERENCES
Anime_Properties(animeID)
);
```

```
CREATE TABLE DATEDIMENSION (
   DATEID INT PRIMARY KEY IDENTITY(1,1),
   DATEVALUE DATE,
   YEAR INT,
   MONTH INT,
   DAY INT
);
DECLARE @STARTDATE DATE = '2020-01-01';
DECLARE @ENDDATE DATE = '2025-12-31';

WHILE @STARTDATE <= @ENDDATE
BEGIN
   INSERT INTO DATEDIMENSION (DATEVALUE, YEAR, MONTH, DAY)
   VALUES (
      @STARTDATE,
      YEAR(@STARTDATE),
      MONTH(@STARTDATE),
      DAY(@STARTDATE)
   );
   SET @STARTDATE = DATEADD(DAY, 1, @STARTDATE);
END
```

i)insert the data into it using this query:

```
-- Insert into Fanbase table
INSERT INTO Fanbase (animeID, AniRating, Members)
SELECT
    CAST(anime_id AS INT),
    CAST(rating AS FLOAT),
    CAST(members AS INT)
FROM StagingAnime;
```

```
-- Insert into Anime_Properties table
INSERT INTO Anime_Properties (animeID, FanbaseID,AnimeName,
animeType, Genre, EpisodeNumber)
SELECT
    CAST(anime_id AS INT),
    f.FanbaseID,
    name,
    type,
    genre,
    CASE
        WHEN ISNUMERIC(episodes) = 1 THEN CAST(episodes AS INT)
        ELSE 0 -- Replace with your desired default value
    END
FROM StagingAnime s
JOIN Fanbase f ON s.anime_id = f.animeID;
```

```
-- Insert into users table
WITH RankedStaging AS (
  SELECT
    CAST(user_id AS INT) AS UserID,
    -- Add appropriate columns from StagingRating for age and releaseDate
    NULL AS age,
    NULL AS releaseDate,
    ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY (SELECT NULL)) AS RowNum
  FROM StagingRating
)
INSERT INTO users (UserID, age, releaseDate)
SELECT UserID, age, releaseDate
FROM RankedStaging
WHERE RowNum = 1; -- Select only the first occurrence of each UserID
```

```
CREATE TABLE DateDimension (
  DateID INT PRIMARY KEY IDENTITY(1,1),
  DateValue DATE,
  Year INT,
  Month INT,
  Day INT
);
DECLARE @StartDate DATE = '2020-01-01';
DECLARE @EndDate DATE = '2025-12-31';

WHILE @StartDate <= @EndDate
BEGIN
  INSERT INTO DateDimension (DateValue, Year, Month, Day)
  VALUES (
    @StartDate,
    YEAR(@StartDate),
    MONTH(@StartDate),
    DAY(@StartDate)
  );

  SET @StartDate = DATEADD(DAY, 1, @StartDate);
END
```

```
WITH RankedDate AS (
    SELECT
        DateID,
        ROW_NUMBER() OVER (ORDER BY NEWID()) AS RowNum
    FROM DateDimension
)
INSERT INTO Rating (UserID, animeID, ratingUser, DateID)
SELECT
    CAST(sr.user_id AS INT),
    CAST(sr.anime_id AS INT),
    CAST(sr.rating AS INT),
    rd.DateID
FROM StagingRating sr
JOIN RankedDate rd ON 1 = 1 -- Cross join without using CROSS JOIN
WHERE sr.user_id IN (SELECT UserID FROM users)
    AND sr.anime_id IN (SELECT animeID FROM Anime_Properties)
    AND sr.anime_id = rd.RowNum;


    -- Drop the table after the procedure execution
    DROP TABLE #InfiniteLoopPrevention;


END;
```

```
    -- Drop the table after the procedure execution
    DROP TABLE #InfiniteLoopPrevention;
END;

EXEC DataLoading;
```

110 %

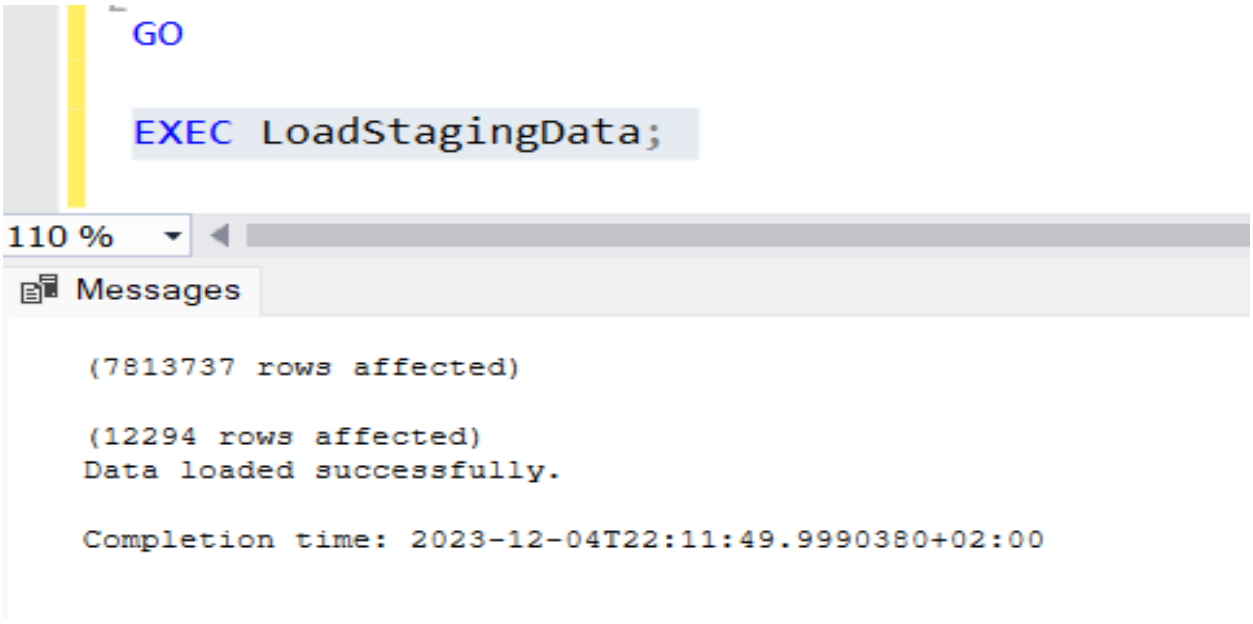
Messages

```
(1 row affected)

(12294 rows affected)

(12294 rows affected)

(73515 rows affected)

(7813727 rows affected)

Completion time: 2023-12-04T22:13:44.4292250+02:00
```

ii)This query is to load the data from the csv files with the right format:

```
CREATE PROCEDURE LoadStagingData
AS
BEGIN
    BEGIN TRY
                    IF OBJECT_ID('dbo.StagingRating', 'U') IS NOT NULL
        DROP TABLE StagingRating;
      -- Check if StagingAnime table exists before dropping
      IF OBJECT_ID('dbo.StagingAnime', 'U') IS NOT NULL
        DROP TABLE StagingAnime;
                    CREATE TABLE StagingRating (
                                user_id VARCHAR(255),
                                anime_id VARCHAR(255),
                                rating VARCHAR(255)
                    );
                    CREATE TABLE StagingAnime (
                                anime_id VARCHAR(255),
                                name VARCHAR(255),
                                genre VARCHAR(255),
                                type VARCHAR(255),
                                episodes VARCHAR(255),
                                rating VARCHAR(255),
                                members VARCHAR(255)  );
                    );
```

```
    -- Bulk insert data into StagingRating table
        BULK INSERT StagingRating
                FROM 'D:\College\Year 3\DataBase Advanced\archive\rating.csv'
    WITH (
      FIELDTERMINATOR = ',',
      ROWTERMINATOR = '\n',
      FIRSTROW = 2 -- Skip header row
    );
-------- Bulk insert data into StagingAnime table with error handling and data cleaning
        BULK INSERT StagingAnime
        FROM 'D:\College\Year 3\DataBase Advanced\archive\anime.csv'
                WITH (
                        FORMAT='CSV',
                        FIRSTROW=2
                );
--------- Additional data cleaning or validation steps can be added here
    PRINT 'Data loaded successfully.';
  END TRY
  BEGIN CATCH
    -- Handle errors
    PRINT 'Error loading data. Check error log for details.';
    PRINT ERROR_MESSAGE();
  END CATCH
END;
GO
```



2)After loading the date from the csv files and inserting it into our tables we need to create a procedure to load the data everyday at a certain time using this query:

```
-- Send email notification
    EXEC msdb.dbo.sp_send_dbmail
        @profile_name = 'Moghazy', -- Specify your mail profile
        @recipients = 'ahmedkhaledmoghazy10@gmail.com', -- System administrator's email
        @subject = @Subject,
        @body = @Body;
END;
GO
```

```
-- Create a separate stored procedure for sending email
ALTER PROCEDURE SendEmail
    @Subject NVARCHAR(255),
    @Body NVARCHAR(1000)
AS
BEGIN
    -- Check if the mail profile exists
    IF NOT EXISTS (SELECT 1 FROM msdb.dbo.sysmail_profile WHERE name = 'Moghazy')
    BEGIN
        PRINT 'Mail profile does not exist.';
        RETURN; -- Exit the stored procedure if the mail profile does not exist
    END
```

```
    -- Call the separate stored procedure to send email
    EXEC SendEmail
        @Subject = 'Data Loading Process Result',
        @Body = @ResultMessage;
END;
GO
EXEC SendEmailNotification;
```

```
-- Create the main stored procedure to send an email to indicates the success or failure of loading the   data everyday
ALTER PROCEDURE SendEmailNotification
AS
BEGIN
    DECLARE @ResultMessage NVARCHAR(1000);
    BEGIN TRY
  --- this part is to set the value of the success or failure to the variable named ResultMessage
        SET @ResultMessage = 'Data loading process succeeded.';
    END TRY
    BEGIN CATCH
        -- Log error details or take appropriate action
        SET @ResultMessage = 'Data loading process failed. Check the error logs for details.';
    END CATCH;
```

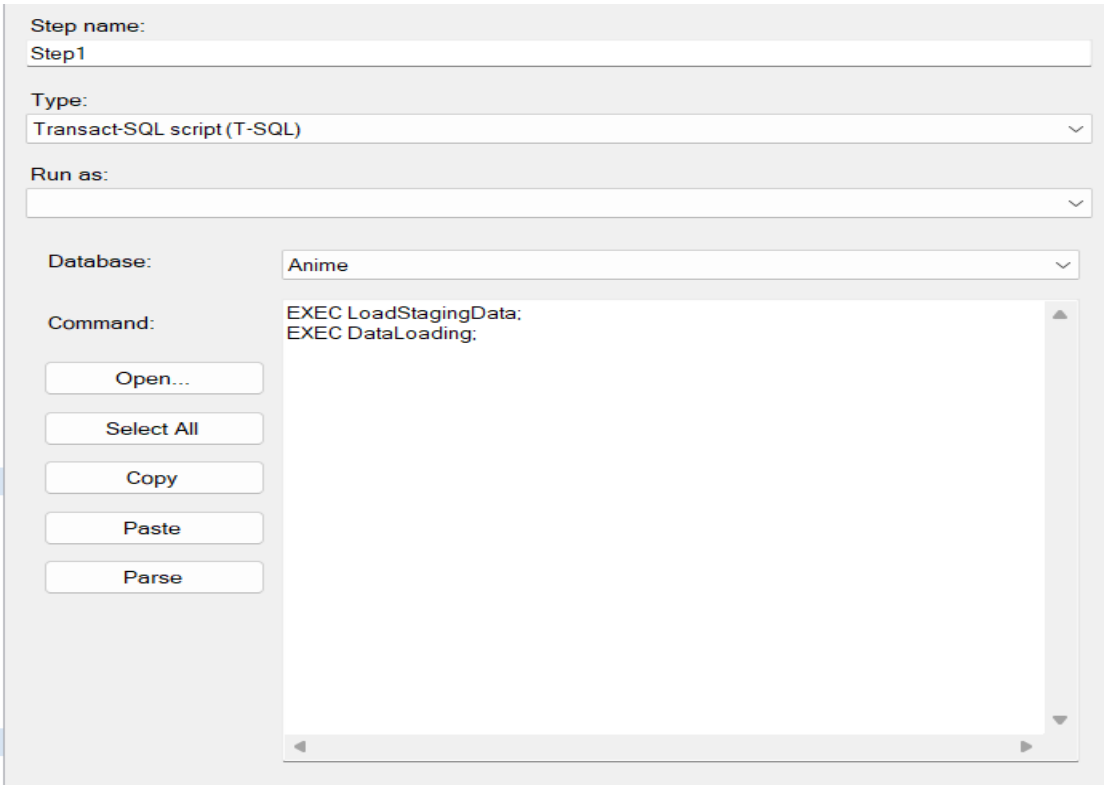
```
END;
GO
EXEC SendEmailNotification;
```

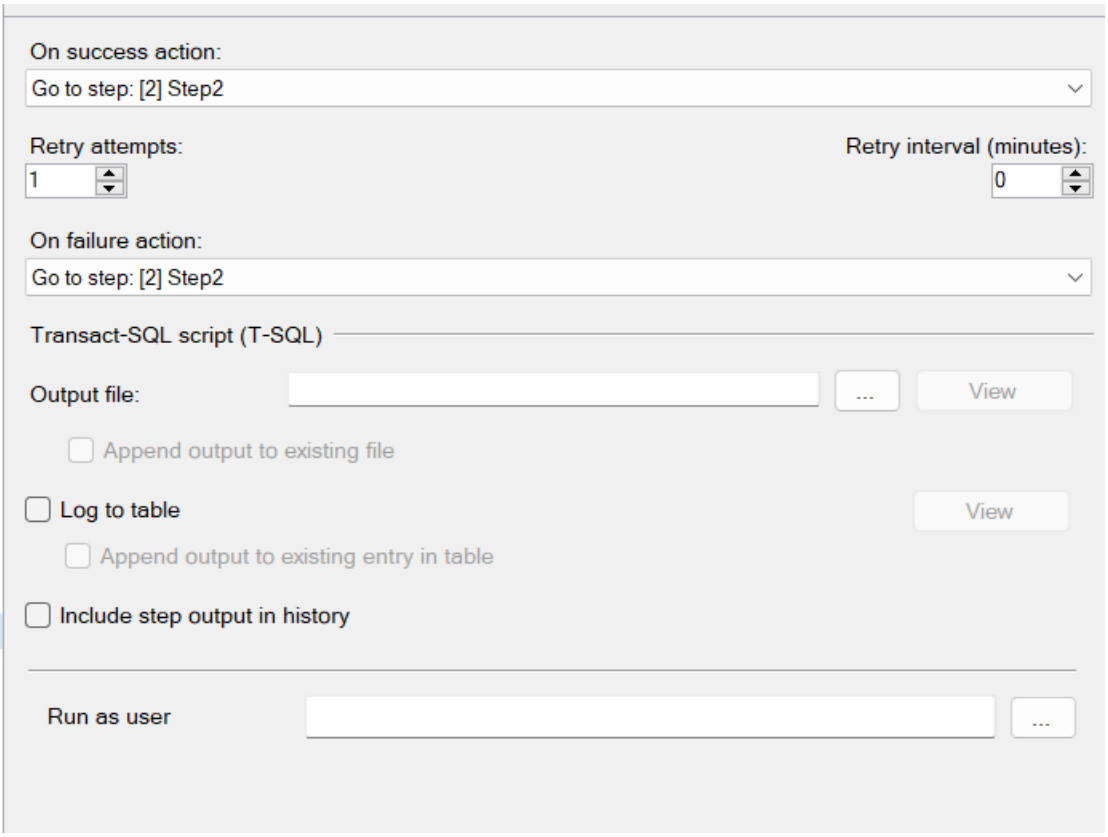
110 %

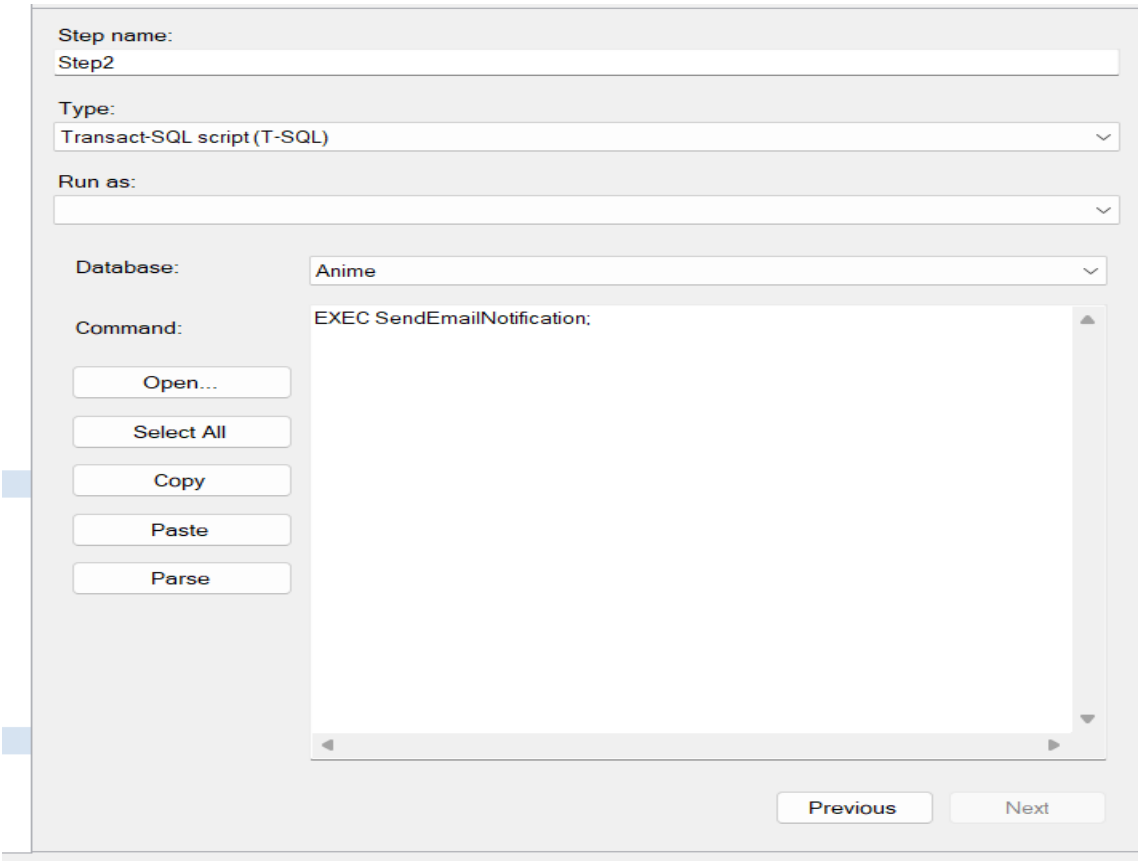
Messages

```
Mail (Id: 14) queued.

Completion time: 2023-12-04T21:59:45.0790891+02:00
```

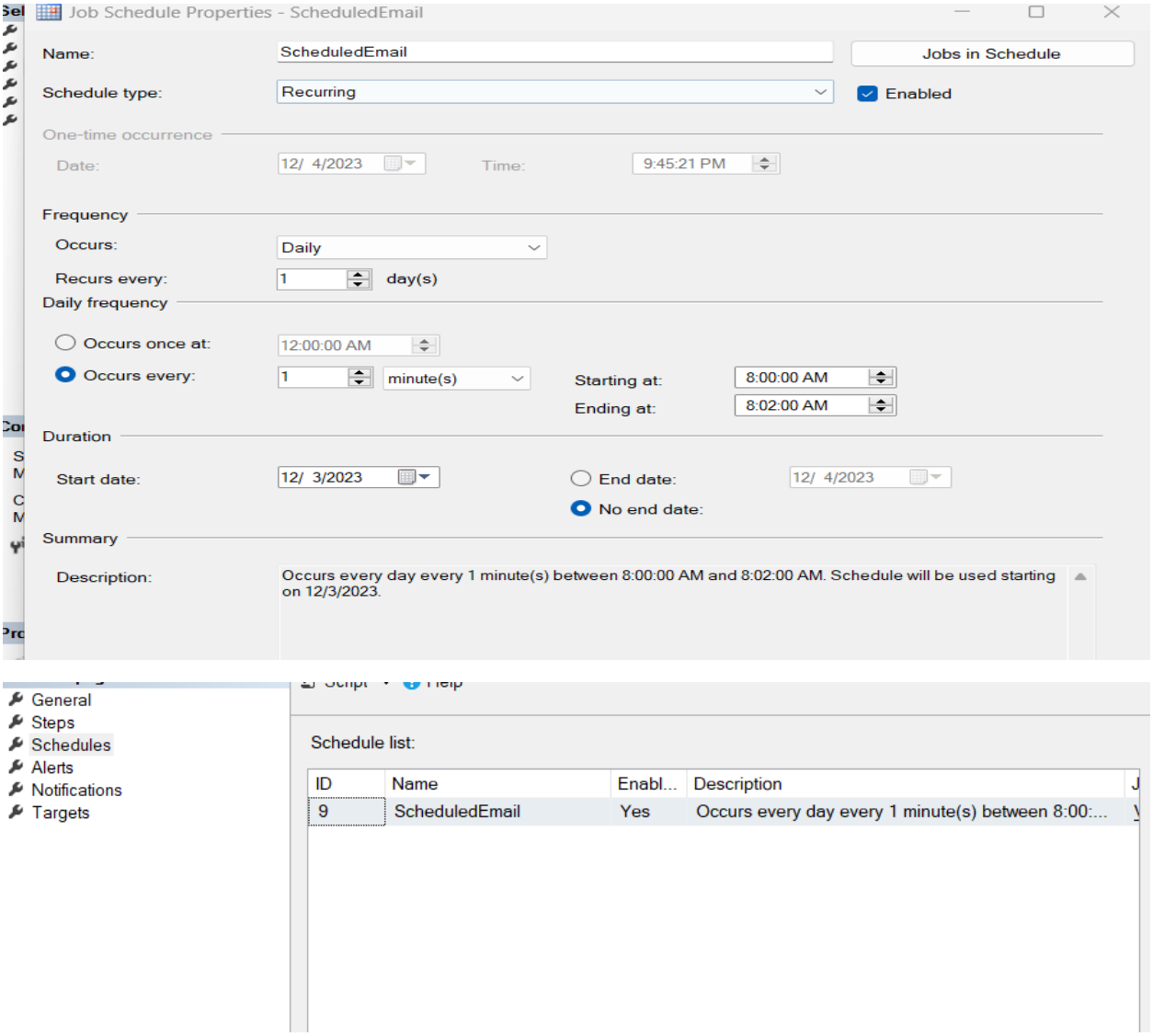
Job that runs:



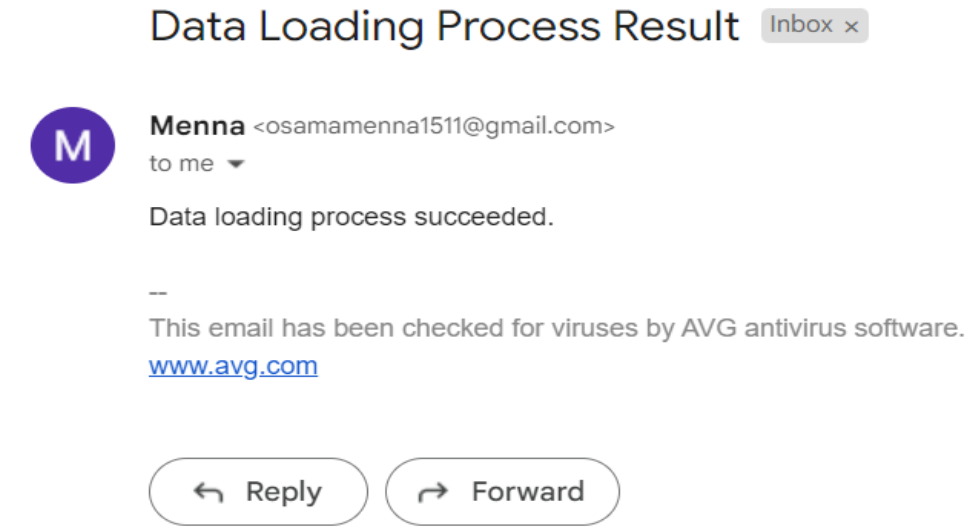

On success action:
Go to step: [2] Step2
Retry attempts:
1
Retry interval (minutes):
0
On failure action:
Go to step: [2] Step2
Transact-SQL script (T-SQL)
Output file:
...
View
Append output to existing file
Log to table
View
Append output to existing entry in table
Include step output in history
Run as user
...

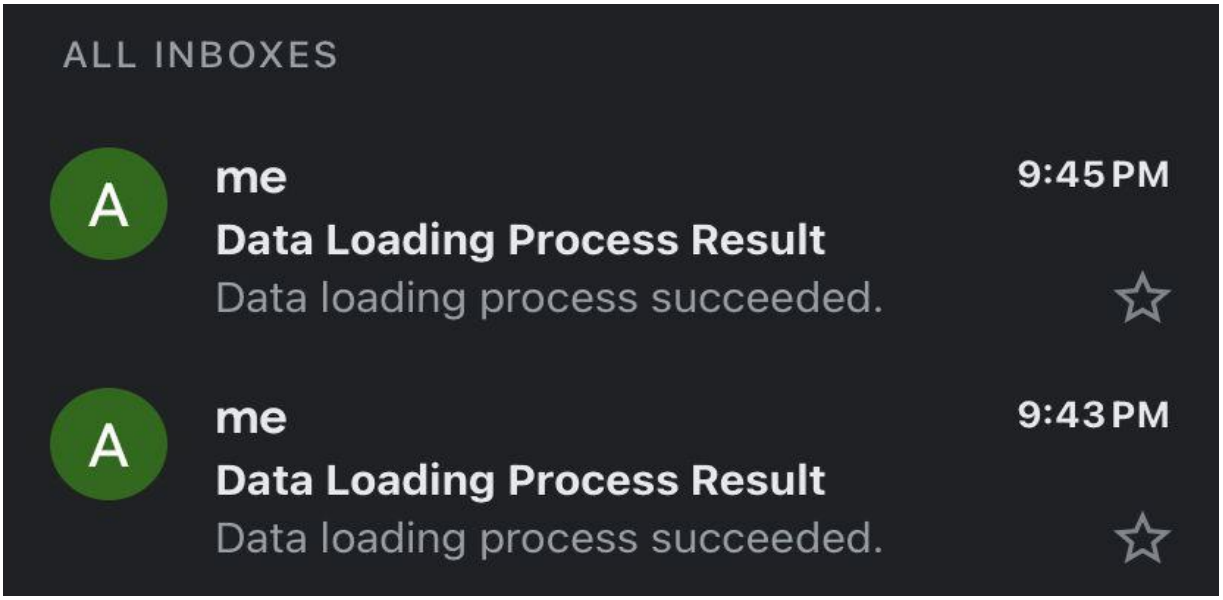

Step name:
Step2
Type:
Transact-SQL script (T-SQL)
Run as:
Database:
Anime
Command:
EXEC SendEmailNotification;
Open...
Select All
Copy
Paste
Parse
Previous
Next

Then we did Schedules:



screen of the sent email.





Emailing list

1. osamamenna1511@gmail.com
2. ahmadgadalla02@gmail.com
3. ahmedkhaledmoghazy10@gmail.com
4. radwabelal263@gmail.com
5. Janna.1118014@stemmaadi.moe.edu.eg