Assignment 1: Data Warehouse

Group ID: DS2

DB Source Name: Anime

Group names:

- Salma Abdelhalim
- Farida Hamid
- Mariam Mohsen
- Mohab Yasser
- AbdelRaouf Essam

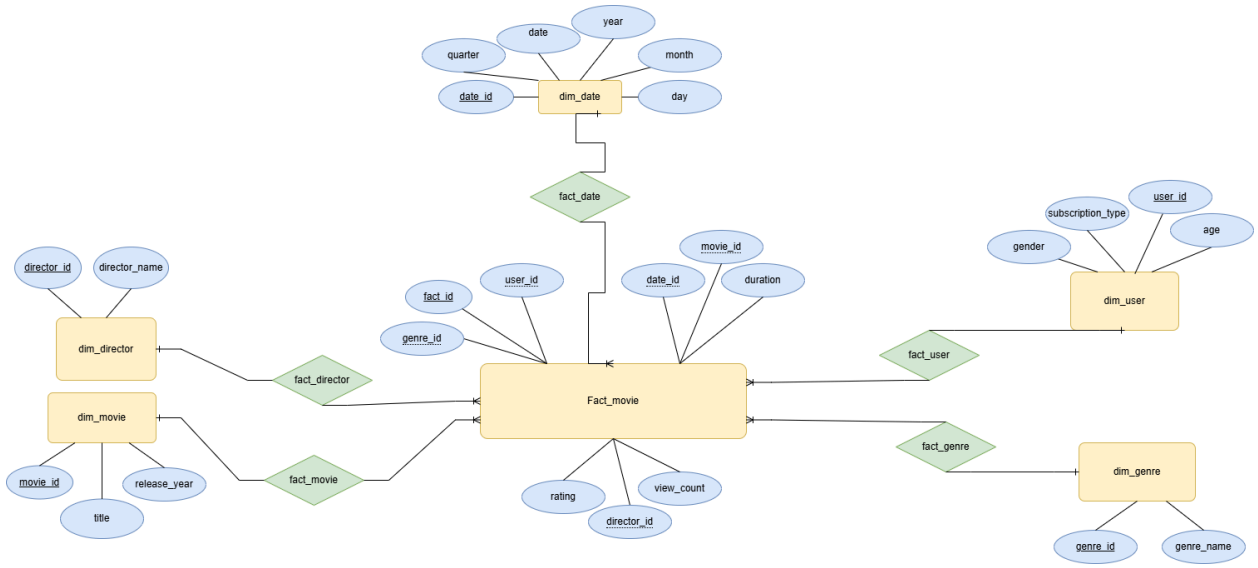
IDS:

- 20227014
- 20227020
- 20227023
- 20227026
- 20227038

Emails:

- salmaabdelhalim2024@gmail.com
- faridahamid2004@gmail.com
- mariammohsen888@gmail.com
- mohabyasser960@gmail.com
- raoufessam63@gmail.com

ERD:



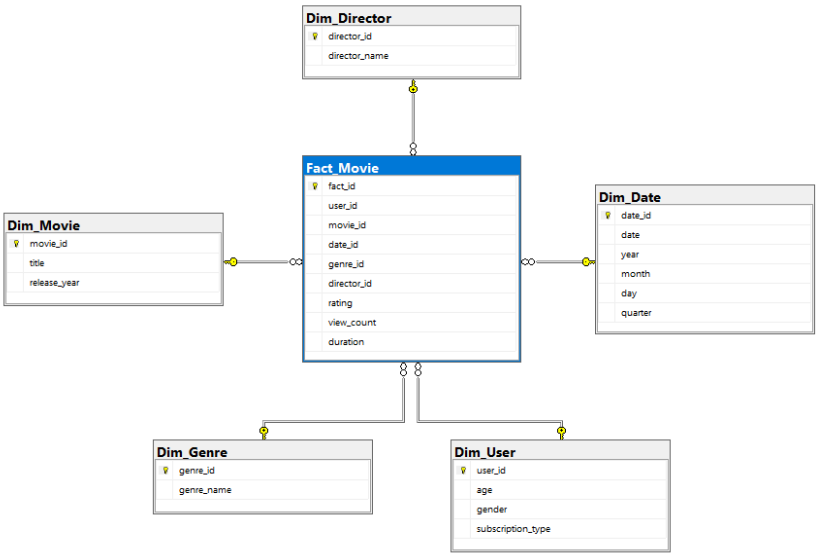
Motivation:

We are creating a Movie Analytics Star Schema to analyse various aspects of user engagement, movie ratings, and viewing trends across different dimensions. The star schema facilitates the analysis of key metrics such as view counts, user ratings, and movie popularity, providing valuable insights for decision-making in the entertainment industry.

Questions:

1. Get the average rating for a movie with it's genre
2. total number of movies per genre
3. top 10 movies by rating
4. Get the most popular director by total movie views (view_count)
5. Count the number of unique users who rated movies by genre
6. top 10 most-watched movies (highest view_count)
7. Analyse user engagement by subscription type (total views per subscription type)

Star Schema Model:



Schema Description:

1. Dimension Tables:

- Dim_Movie: Contains movie-related data (movie_id, title, release_year).
- Dim_User: Represents user attributes (user_id, age, gender, subscription_type).
- Dim_Genre: Stores genres (genre_id, genre_name).
- Dim_Director: Captures director information (director_id, director_name).
- Dim_Date: Provides a time hierarchy (year, quarter, month, day).

2. Dimension Levels:

- Dim_Date: Organized hierarchically (Year → Quarter → Month → Day).
- Other dimensions (e.g., Dim_User, Dim_Movie) are flat (non-hierarchical).

3. Measures:

- Rating: Numeric value representing a user's rating of a movie.
- view_count: Count of ratings per movie.
- Duration: Length of the movie.

Fact table Query:

Create Query:

```
Create TABLE Fact_Movie
(
   fact_id INT PRIMARY KEY,
   user_id INT,
   movie_id INT,
   date_id INT,
   genre_id INT,
   director_id INT,
   rating INT,
   view_count INT,
   duration INT,
   FOREIGN KEY (user_id) REFERENCES Dim_User(user_id),
   FOREIGN KEY (movie_id) REFERENCES Dim_Movie(movie_id),
   FOREIGN KEY (date_id) REFERENCES Dim_Date(date_id),
   FOREIGN KEY (genre_id) REFERENCES Dim_Genre(genre_id),
   FOREIGN KEY (director_id) REFERENCES Dim_Director(director_id) );
```

Insert Query:

```
WITH FactData AS (
   SELECT
      CAST(SR.user_id AS INT) AS user_id,
      CAST(SA.anime_id AS INT) AS movie_id,
      (SELECT date_id FROM Dim_Date WHERE date = CAST(GETDATE() AS DATE)) AS
date_id,
      FLOOR(RAND(CHECKSUM(NEWID())) * 15) + 1 AS director_id,
      CAST(SR.rating AS INT) AS rating,
      FLOOR(RAND(CHECKSUM(NEWID())) * 180) + 60 AS duration,
      FLOOR(RAND(CHECKSUM(NEWID())) * 3265) + 1 AS genre_id,
      ROW_NUMBER() OVER (ORDER BY (SELECT NULL)) AS fact_id
   FROM dbo.StagingAnime SA
   JOIN dbo.StagingRating SR ON SA.anime_id = SR.anime_id
)

INSERT INTO Fact_Movie (fact_id, user_id, movie_id, date_id, director_id, rating,
view_count, duration, genre_id)
SELECT
   FD.fact_id,
   FD.user_id,
   FD.movie_id,
   FD.date_id,
   FD.director_id,
   FD.rating,
   VC.view_count,
```

```
    FD.duration,
    FD.genre_id
FROM FactData FD
JOIN ViewCount VC ON FD.movie_id = VC.movie_id
-- Ensure the user_id exists in Dim_User (validate against Dim_User table)
JOIN dbo.Dim_User DU ON FD.user_id = DU.user_id;
```

Dimension Tables Queries:

1. Dim_Date

Create Query:

```
CREATE TABLE Dim_Date
 (date_id INT PRIMARY KEY,
    date DATE,
    year INT,
    month INT,
    day INT,
    quarter INT);
```

Insert Query:

```
INSERT INTO Dim_Date (date_id, date, year, month, day, quarter)
SELECT
    CAST(CONVERT(VARCHAR(8), date_value, 112) AS INT) AS date_id,
    date_value AS date,
    YEAR(date_value) AS year,
    MONTH(date_value) AS month,
    DAY(date_value) AS day,
    CASE
        WHEN MONTH(date_value) BETWEEN 1 AND 3 THEN 1
        WHEN MONTH(date_value) BETWEEN 4 AND 6 THEN 2
        WHEN MONTH(date_value) BETWEEN 7 AND 9 THEN 3
        WHEN MONTH(date_value) BETWEEN 10 AND 12 THEN 4
    END AS quarter
FROM DateRange
OPTION (MAXRECURSION 0);
```

2. Dim_Genre

Create Query:

```
CREATE TABLE Dim_Genre
 (
    genre_id INT PRIMARY KEY,
    genre_name VARCHAR(255)
);
```

Insert Query:

```
INSERT INTO Dim_Genre (genre_id, genre_name)
SELECT
   ROW_NUMBER() OVER (ORDER BY genre) AS genre_id,
   genre
FROM StagingAnime
GROUP BY genre;
```

3. Dim_Movie

Create Query:

```
CREATE TABLE Dim_Movie
 ( movie_id INT PRIMARY KEY,
   title VARCHAR(255),
   release_year INT);
```

Insert Query:

```
INSERT INTO Dim_Movie (movie_id, title, release_year)
SELECT  DISTINCT
   anime_id AS movie_id,
   name AS title,
   (   CASE
       WHEN RAND(CHECKSUM(NEWID())) < 0.14 THEN 2011
       WHEN RAND(CHECKSUM(NEWID())) < 0.28 THEN 2015
       WHEN RAND(CHECKSUM(NEWID())) < 0.42 THEN 2017
       WHEN RAND(CHECKSUM(NEWID())) < 0.57 THEN 2019
       WHEN RAND(CHECKSUM(NEWID())) < 0.71 THEN 2020
       WHEN RAND(CHECKSUM(NEWID())) < 0.85 THEN 2021
       ELSE 2022
       END) AS release_year
FROM StagingAnime;
```

4. Dim_Director

Create Query:

```
CREATE TABLE Dim_Director
 ( director_id INT PRIMARY KEY,
   director_name VARCHAR(100));
```

Insert Query:

```
INSERT INTO Dim_Director (director_id, director_name)
VALUES
(1, 'John Smith'),
(2, 'Sarah Johnson'),
(3, 'Michael Brown'),
(4, 'Emily Davis'),
```

```
(5, 'James Wilson'),
(6, 'Olivia Taylor'),
(7, 'William Martinez'),
(8, 'Sophia Garcia'),
(9, 'Liam Anderson'),
(10, 'Charlotte Thompson'),
(11, 'Noah White'),
(12, 'Isabella Harris'),
(13, 'Mason Clark'),
(14, 'Mia Robinson'),
(15, 'Elijah Walker');
```

5. Dim_User

Create Query:

```
CREATE TABLE Dim_User
 ( user_id INT PRIMARY KEY,
   age INT,
   gender VARCHAR(10),
   subscription_type VARCHAR(20));
```

Insert Query:

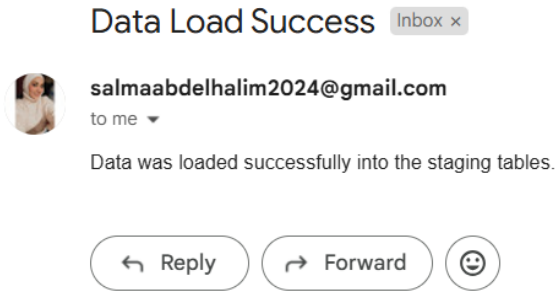
```
INSERT INTO dbo.Dim_User (user_id, age, gender, subscription_type)
      SELECT DISTINCT
             CAST(CLEANED.user_id AS INT) AS user_id,
             FLOOR(RAND(CHECKSUM(NEWID())) * (60 - 18 + 1)) + 18 AS age,
             CASE
                 WHEN RAND(CHECKSUM(NEWID())) > 0.5 THEN 'Male'
                 ELSE 'Female'
                 END AS gender,
            CASE
                 WHEN RAND(CHECKSUM(NEWID())) < 0.33 THEN 'Premium'
                 WHEN RAND(CHECKSUM(NEWID())) < 0.66 THEN 'Standard'
                 ELSE 'Free'
                 END AS subscription_type
          FROM (SELECT DISTINCT REPLACE(REPLACE(SR.user_id, CHAR(13), ''),
CHAR(10), '') AS user_id FROM dbo.StagingRating SR) CLEANED
          WHERE NOT EXISTS (SELECT 1 FROM dbo.Dim_User DU WHERE
DU.user_id = CAST(CLEANED.user_id AS INT));
```

Stored procedure to load data from CSV file:

```
Alter PROCEDURE LoadAndNotifyData
AS BEGIN
   DECLARE @email_subject NVARCHAR(255);
   DECLARE @email_body NVARCHAR(MAX);
   BEGIN TRY
      DROP TABLE IF EXISTS StagingRating;
```

```
        DROP TABLE IF EXISTS StagingAnime;
        CREATE TABLE StagingRating(
            user_id VARCHAR(255),
            anime_id VARCHAR(255),
            rating INT  );
        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 StagingRating
        FROM 'C:\Users\Salma Abdelhalim\Desktop\rating.csv'
        WITH (
            FIELDTERMINATOR = ',',
            ROWTERMINATOR = '0x0A',
            FIRSTROW = 2 -- Skip header row );
         BULK INSERT StagingAnime
        FROM 'C:\Users\Salma Abdelhalim\Downloads\anime_cleaned.csv'
        WITH (
            FORMAT = 'CSV',
            FIRSTROW = 2,
            FIELDTERMINATOR = ',',
            ROWTERMINATOR = '0x0A',
            TABLOCK);
        SET @email_subject = 'Data Load Success';
        SET @email_body = 'Data was loaded successfully into the staging tables.';
        PRINT 'Data loaded successfully.';
    END TRY
    BEGIN CATCH
        SET @email_subject = 'Data Load Failure';
        SET @email_body = 'Error occurred while loading data. Error: ' + ERROR_MESSAGE();
        PRINT 'Error loading data. Check error log for details.';
        PRINT ERROR_MESSAGE();
    END CATCH
    BEGIN TRY
        EXEC msdb.dbo.sp_send_dbmail
            @profile_name = 'Salma',
            @recipients = 'salmaabdelhalim2024@gmail.com',
            @subject = @email_subject,
            @body = @email_body;
        PRINT 'Email sent successfully.';
    END TRY
    BEGIN CATCH
        PRINT 'Error sending email.';
        PRINT ERROR_MESSAGE();
    END CATCH
END; EXEC LoadAndNotifyData;
```

Print screen of the sent email:



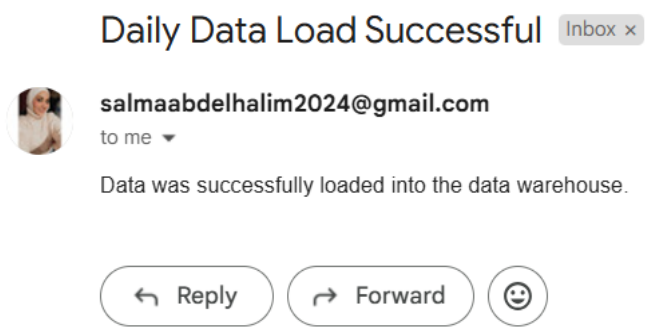
Stored procedure to load data from CSV file and add DWH:

```
ALTER TABLE dbo.StagingRating ADD last_updated DATETIME DEFAULT GETDATE();
ALTER TABLE dbo.StagingAnime ADD last_updated DATETIME DEFAULT GETDATE();
ALTER PROCEDURE LoadAndNotifyData
AS
BEGIN
   DECLARE @email_subject NVARCHAR(255);
   DECLARE @email_body NVARCHAR(MAX);
   DECLARE @last_run_time DATETIME;
   SET @last_run_time = (SELECT MAX(last_updated) FROM dbo.Fact_Movie);
   BEGIN TRY
      IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME = 'StagingRating' AND COLUMN_NAME = 'last_updated')
      BEGIN
         ALTER TABLE dbo.StagingRating ADD last_updated DATETIME DEFAULT
GETDATE();
         PRINT 'Added last_updated column to StagingRating';
      END
      IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME = 'StagingAnime' AND COLUMN_NAME = 'last_updated')
      BEGIN
         ALTER TABLE dbo.StagingAnime ADD last_updated DATETIME DEFAULT
GETDATE();
         PRINT 'Added last_updated column to StagingAnime';
      END
      IF NOT EXISTS (SELECT 1 FROM INFORMATION_SCHEMA.COLUMNS WHERE
TABLE_NAME = 'Fact_Movie' AND COLUMN_NAME = 'last_updated')
      BEGIN
         ALTER TABLE dbo.Fact_Movie ADD last_updated DATETIME DEFAULT GETDATE();
         PRINT 'Added last_updated column to Fact_Movie';
      END
        CREATE TABLE #TempStagingRating (
```

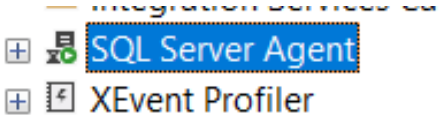
```
        user_id VARCHAR(255),
        anime_id VARCHAR(255),
        rating INT);
    BULK INSERT #TempStagingRating
    FROM 'E:\new_rating_records.csv'
    WITH (FIELDTERMINATOR = ',',
        ROWTERMINATOR = '0x0A',
        FIRSTROW = 2 -- Skip header row );
    MERGE INTO StagingRating AS Target
    USING #TempStagingRating AS Source
    ON Target.user_id = Source.user_id AND Target.anime_id = Source.anime_id
    WHEN MATCHED THEN
        UPDATE SET Target.rating = Source.rating, Target.last_updated = GETDATE()
    WHEN NOT MATCHED THEN
        INSERT (user_id, anime_id, rating, last_updated)
        VALUES (Source.user_id, Source.anime_id, Source.rating, GETDATE());
    DROP TABLE #TempStagingRating;
    CREATE TABLE #TempStagingAnime (
        anime_id VARCHAR(255),
        name VARCHAR(255),
        genre VARCHAR(255),
        type VARCHAR(255),
        episodes VARCHAR(255),
        rating VARCHAR(255),
        members VARCHAR(255) );
    BULK INSERT #TempStagingAnime
    FROM 'E:\anime_cleaned.csv'
    WITH (FIELDTERMINATOR = ',',
        ROWTERMINATOR = '0x0A',
        FIRSTROW = 2 -- Skip header row);
    MERGE INTO StagingAnime AS Target
    USING #TempStagingAnime AS Source
    ON Target.anime_id = Source.anime_id
    WHEN MATCHED THEN
        UPDATE SET
            Target.name = Source.name,
            Target.genre = Source.genre,
            Target.type = Source.type,
            Target.episodes = Source.episodes,
            Target.rating = Source.rating,
            Target.members = Source.members,
            Target.last_updated = GETDATE()
    WHEN NOT MATCHED THEN
        INSERT (anime_id, name, genre, type, episodes, rating, members, last_updated)
        VALUES (Source.anime_id, Source.name, Source.genre, Source.type,
Source.episodes, Source.rating, Source.members, GETDATE());
    DROP TABLE #TempStagingAnime;
    INSERT INTO dbo.Dim_Movie (movie_id, title, release_year)
    SELECT DISTINCT
    CAST(anime_id AS INT) AS movie_id,
```

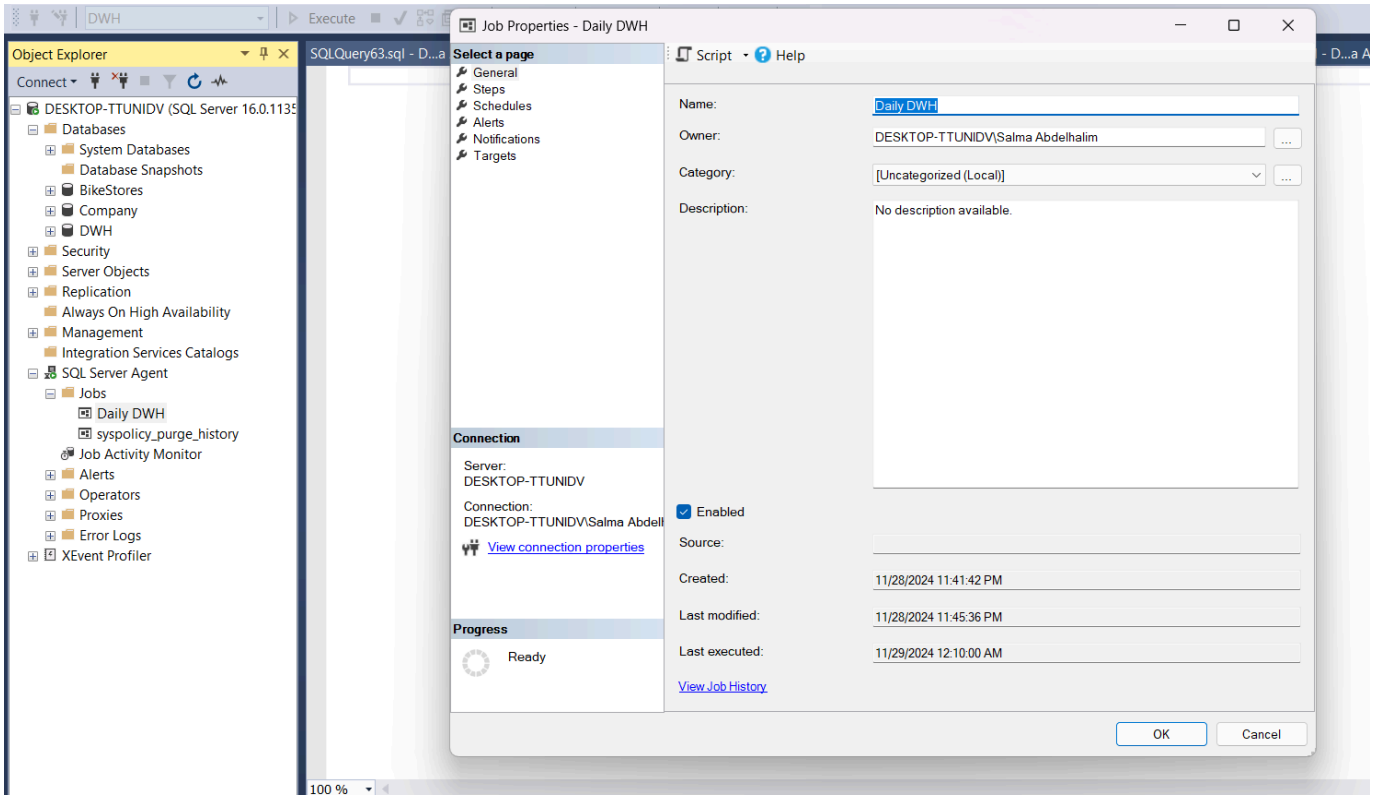
```
        name AS title,
        CASE
            WHEN RAND(CHECKSUM(NEWID())) < 0.14 THEN 2011
            WHEN RAND(CHECKSUM(NEWID())) < 0.28 THEN 2015
            WHEN RAND(CHECKSUM(NEWID())) < 0.42 THEN 2017
            WHEN RAND(CHECKSUM(NEWID())) < 0.57 THEN 2019
            WHEN RAND(CHECKSUM(NEWID())) < 0.71 THEN 2020
            WHEN RAND(CHECKSUM(NEWID())) < 0.85 THEN 2021
            ELSE 2022
        END AS release_year
       FROM dbo.StagingAnime SA
       WHERE NOT EXISTS (SELECT 1 FROM dbo.Dim_Movie DM WHERE DM.movie_id =
CAST(SA.anime_id AS INT));
       INSERT INTO dbo.Dim_User (user_id, age, gender, subscription_type)
       SELECT DISTINCT
              CAST(CLEANED.user_id AS INT) AS user_id,
              FLOOR(RAND(CHECKSUM(NEWID())) * (60 - 18 + 1)) + 18 AS age, -- Random
age between 18 and 60
              CASE
                  WHEN RAND(CHECKSUM(NEWID())) > 0.5 THEN 'Male'
                  ELSE 'Female'
                  END AS gender,
             CASE
                  WHEN RAND(CHECKSUM(NEWID())) < 0.33 THEN 'Premium'
                  WHEN RAND(CHECKSUM(NEWID())) < 0.66 THEN 'Standard'
                  ELSE 'Free'
                  END AS subscription_type
           FROM (SELECT DISTINCT REPLACE(REPLACE(SR.user_id, CHAR(13), ''),
CHAR(10), '') AS user_id FROM dbo.StagingRating SR) CLEANED
           WHERE NOT EXISTS (SELECT 1 FROM dbo.Dim_User DU WHERE DU.user_id =
CAST(CLEANED.user_id AS INT));
        INSERT INTO dbo.Dim_Genre (genre_id, genre_name)
        SELECT DISTINCT RN.genre_id, RN.genre_name
        FROM (SELECT
        ROW_NUMBER() OVER (ORDER BY genre) + (SELECT ISNULL(MAX(genre_id), 0)
           FROM dbo.Dim_Genre) AS genre_id,
        genre AS genre_name
        FROM dbo.StagingAnime
        GROUP BY genre) RN
        WHERE NOT EXISTS (SELECT 1 FROM dbo.Dim_Genre DG WHERE DG.genre_name
= RN.genre_name);
        WITH FactData AS (SELECT
              CAST(SR.user_id AS INT) AS user_id,
              CAST(SA.anime_id AS INT) AS movie_id,
              (SELECT date_id FROM dbo.Dim_Date WHERE date = CAST(GETDATE() AS
DATE)) AS date_id,
              FLOOR(RAND(CHECKSUM(NEWID())) * 15) + 1 AS director_id,
              CAST(SR.rating AS INT) AS rating,
              FLOOR(RAND(CHECKSUM(NEWID())) * 180) + 60 AS duration,
              FLOOR(RAND(CHECKSUM(NEWID())) * 3265) + 1 AS genre_id,
```

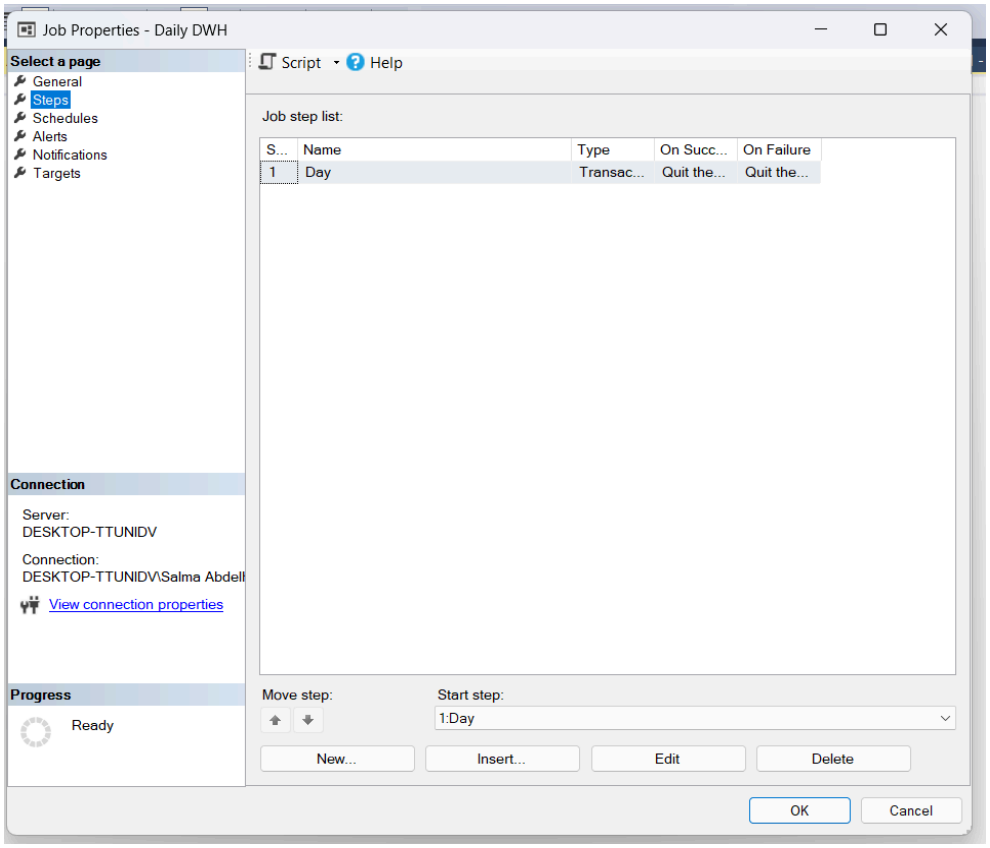
```
            ROW_NUMBER() OVER (ORDER BY SR.user_id, SA.anime_id) +
            (SELECT ISNULL(MAX(fact_id), 0) FROM dbo.Fact_Movie) AS fact_id
FROM dbo.StagingAnime SA
        JOIN dbo.StagingRating SR ON SA.anime_id = SR.anime_id)
    INSERT INTO dbo.Fact_Movie (fact_id, user_id, movie_id, date_id, director_id, rating,
view_count, duration, genre_id, last_updated)
    SELECT
        FD.fact_id,  FD.user_id, FD.movie_id, FD.date_id, FD.director_id, FD.rating,
        VC.view_count, FD.duration, FD.genre_id,
        GETDATE() AS last_updated -- Timestamp for insertion
    FROM FactData FD
    JOIN (SELECT anime_id AS movie_id, COUNT(DISTINCT user_id) AS view_count
        FROM dbo.StagingRating
        GROUP BY anime_id) VC ON FD.movie_id = VC.movie_id
    WHERE NOT EXISTS ( SELECT 1 FROM dbo.Fact_Movie FM
        WHERE FM.user_id = FD.user_id AND FM.movie_id = FD.movie_id
          AND FM.date_id = FD.date_id );
    SET @email_subject = 'Daily Data Load Successful';
    SET @email_body = 'Data was successfully loaded into the data warehouse.';
    PRINT 'Data load completed successfully.';
  END TRY
  BEGIN CATCH
    SET @email_subject = 'Daily Data Load Failed';
    SET @email_body = 'Error during data load: ' + ERROR_MESSAGE();
    PRINT ERROR_MESSAGE();
  END CATCH
  EXEC msdb.dbo.sp_send_dbmail
    @profile_name = 'Salma',
    @recipients = 'salmaabdelhalim2024@gmail.com',
    @subject = @email_subject,
    @body = @email_body;
END;EXEC LoadAndNotifyData;
```

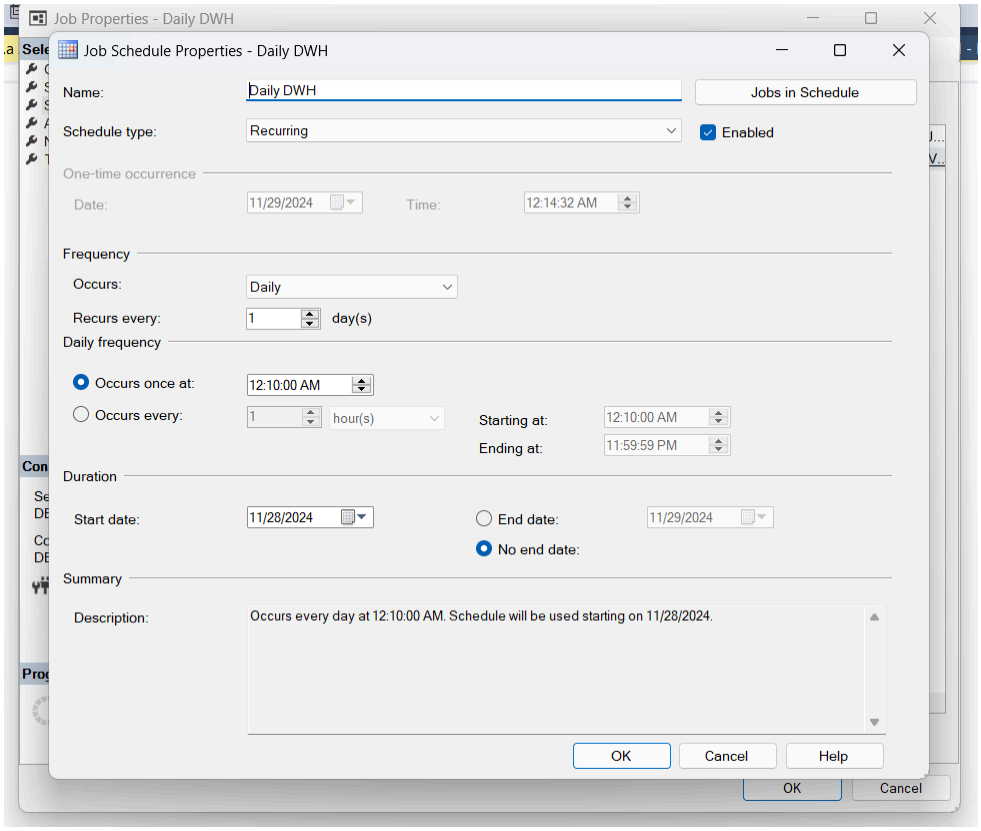


SQL Job:


SQL Server Agent
XEvent Profiler


Object Explorer
Connect
DESKTOP-TTUNIDV (SQL Server 16.0.1135
Databases
System Databases
Database Snapshots
BikeStores
Company
DWH
Security
Server Objects
Replication
Always On High Availability
Management
Integration Services Catalogs
SQL Server Agent
Jobs
Daily DWH
syspolicy_purge_history
Job Activity Monitor
Alerts
Operators
Proxies
Error Logs
XEvent Profiler
Job Properties - Daily DWH
Select a page
General
Steps
Schedules
Alerts
Notifications
Targets
Script
Help
Name:
Daily DWH
Owner:
DESKTOP-TTUNIDV\Salma Abdelhalim
Category:
[Uncategorized (Local)]
Description:
No description available.
Enabled
Source:
Created:
11/28/2024 11:41:42 PM
Last modified:
11/28/2024 11:45:36 PM
Last executed:
11/29/2024 12:10:00 AM
View Job History
Connection
Server:
DESKTOP-TTUNIDV
Connection:
View connection properties
Progress
Ready
OK
Cancel
100 %


Job Properties - Daily DWH
Select a page
General
Steps
Schedules
Alerts
Notifications
Targets
Script
Help
Job step list:
S...
Name
Type
On Succ...
On Failure
1
Day
Transac...
Quit the...
Quit the...
Connection
Server:
DESKTOP-TTUNIDV
Connection:
DESKTOP-TTUNIDV\Salma Abdell
View connection properties
Progress
Ready
Move step:
Start step:
1:Day
New...
Insert...
Edit
Delete
OK
Cancel


Job Properties - Daily DWH
Job Schedule Properties - Daily DWH
Name:
Daily DWH
Jobs in Schedule
Schedule type:
Recurring
Enabled
One-time occurrence
Date:
11/29/2024
Time:
12:14:32 AM
Frequency
Occurs:
Daily
Recurs every:
1
day(s)
Daily frequency
Occurs once at:
12:10:00 AM
Occurs every:
1
hour(s)
Starting at:
12:10:00 AM
Ending at:
11:59:59 PM
Duration
Start date:
11/28/2024
End date:
11/29/2024
No end date:
Summary
Description:
Occurs every day at 12:10:00 AM. Schedule will be used starting on 11/28/2024.
OK
Cancel
Help
OK
Cancel

Questions Queries:

1. Get the average rating for a movie with its genre

Query:

```
SELECT
   g.genre_name,
   m.title, -- Assuming there is a movie_name column in the Fact_Movie table
   AVG(f.rating) AS avg_rating
FROM
   Fact_Movie f
JOIN
   Dim_Genre g ON f.movie_id = g.genre_id
JOIN
   Dim_Movie m ON f.movie_id = m.movie_id
GROUP BY
   g.genre_name, m.title
ORDER BY
   avg_rating DESC;
```

Result:

100 %

Results | Messages

	genre_name	title	avg_rating
1	Action, Drama, Samurai	Ginga Eiyuu Densetsu	8
2	Horror, Sci-Fi, Space	Ginga Eiyuu Densetsu: Waga Yuku wa Hoshi no Taikai	7
3	Adventure, Comedy, Demons, Fantasy, Magic, Romanc...	NHK ni Youkoso!	7
4	Action, Adventure, Fantasy, Magic, Martial Arts, Shounen	Gankutsuou	7
5	Hentai, Parody, Sci-Fi	Ashita no Joe 2	7
6	Action, Adventure, Comedy, Ecchi, Fantasy, Magic, Sci-...	Rurouni Kenshin: Meiji Kenkaku Romantan - Tsuiok...	7
7	Action, Ecchi, Harem, Mecha, Romance, School, Sci-Fi	Ouran Koukou Host Club	7
8	Ecchi, Martial Arts, School, Super Power	Igano Kabamaru	7
9	Action, Adventure, Fantasy, Sci-Fi, Shounen	Hajime no Ippo	7
10	Comedy, Ecchi, Kids, School	Higurashi no Naku Koro ni Kai	7
11	Action, Adventure, Drama, Sci-Fi, Space	Samurai Champloo	7
12	Comedy, Demons, Fantasy, Historical, Shoujo, Supern...	Nodame Cantabile	7
13	Action, Adventure, Drama, Psychological, Sci-Fi	Sen to Chihiro no Kamikakushi	7
14	Action, Adventure, Fantasy, Magic, Romance, Shounen...	Great Teacher Onizuka	7
15	Action, Comedy, Ecchi, Martial Arts, School, Shounen, ...	Mushishi	7
16	Action, Hentai, Supernatural	Aria The Natural	7
17	Action, Adventure, Comedy, Drama, School, Sci-Fi, Sho...	Berserk	7
18	Historical, Mystery, Shoujo	Ookami to Koushinryou	7
19	Action, Adventure, Comedy, Police	Fullmetal Alchemist	7
20	Comedy, Game	Darker than Black: Kuro no Keiyakusha	7

Query executed successfully.

2. Total number of movies per genre

Query:

```
--total number of movies per genre
SELECT  TOP 10
   g.genre_name,
   COUNT(f.movie_id) AS total_movies
FROM
   Fact_Movie f
JOIN
   Dim_Genre g ON f.movie_id = g.genre_id
GROUP BY
   g.genre_name
ORDER BY
   total_movies DESC
```

Result:

100 %

Results | Messages

	genre_name	total_movies
1	Adventure, Fantasy, Josei, Romance	39340
2	Adventure, Historical	27718
3	Action, Adventure, Fantasy, Game, Shounen	27506
4	Action, Adventure, Comedy, Drama, Fantasy, Magic,...	25925
5	Action, Adventure, Comedy, Police	25032
6	Harem, School, Shoujo, Vampire	24242
7	Action, Adventure, Drama, Psychological, Sci-Fi	22974
8	Comedy, Martial Arts, Romance, Shounen	21721
9	Comedy, Fantasy, Mecha, Shounen	19337
10	Action, Ecchi, Harem, Mecha, Romance, School, Sc...	19064

Query executed successfully.

3. Top 10 movies by rating

Query:

```
--top 10 movies by rating
SELECT Top 10
    m.title,
    AVG(f.rating) AS avg_rating
FROM
    Fact_Movie f
            JOIN
    Dim_Movie m ON f.movie_id = m.movie_id
GROUP BY
    m.title
ORDER BY
    avg_rating DESC
```

Result:

100 %

Results | Messages

	title	avg_rating
1	Shiroi Zou	10
2	Warui no wo Taose!! Salaryman Man	10
3	Choegang Top Plate	10
4	STAR BEAT!: Hoshi no Kodou	10
5	Tang Lang Bu Chan	9
6	Yakushiji Ryouko no Kaiki Jikenbo: Hamachou, Voic...	9
7	Doukyuusei	9
8	Shiranpuri	9
9	Fullmetal Alchemist: Brotherhood	8
10	Sword Gai	8

Query executed successfully.

4. Get the most popular director by total movie views (view_count)

Query:

```
--Get the most popular director by total movie views (view_count):
SELECT
    d.director_name,
    SUM(CAST(f.view_count AS BIGINT)) AS total_views
FROM Fact_Movie f
JOIN Dim_Director d ON f.director_id = d.director_id
GROUP BY d.director_name
ORDER BY total_views DESC;
```

Result:

100 %

Results | Messages

	director_name	total_views
1	Michael Brown	3442427130
2	John Smith	3442295332
3	Emily Davis	3440941136
4	Elijah Walker	3440800914
5	Olivia Taylor	3440699433
6	Charlotte Thompson	3437269533
7	Sophia Garcia	3436985575
8	Noah White	3434923851
9	Mia Robinson	3434593916
10	Mason Clark	3433752126
11	James Wilson	3433410373
12	William Martinez	3432111960
13	Liam Anderson	3430605713
14	Isabella Harris	3430145616

Query executed successfully.

5. Count the number of unique users who rated movies by genre

Query:

```
--Count the number of unique users who rated movies by genre:
SELECT
    g.genre_name,
    COUNT(DISTINCT f.user_id) AS unique_users
FROM Fact_Movie f
JOIN Dim_Genre g ON f.genre_id = g.genre_id
GROUP BY g.genre_name
ORDER BY unique_users DESC;
```

Result:

100 %

Results | Messages

	genre_name	unique_users
1	Horror, Mystery, Psychological, Thriller	2433
2	Hentai, Magic, Supernatural	2431
3	Drama, Kids	2424
4	Action, Adventure, Comedy, Drama, Fantasy, Kids	2422
5	Music, Slice of Life	2422
6	Comedy, Seinen, Supernatural	2417
7	Action, Fantasy, Historical, Martial Arts, Supernatural	2417
8	Drama, Romance, Slice of Life, Supernatural	2416
9	Drama, Fantasy, Romance	2415
10	Action, Drama, Shounen	2414
11	Action, Hentai, Horror, Supernatural	2413
12	Action, Adventure, Comedy, Demons, Supernatural, ...	2413
13	Parody, Seinen	2411
14	Action, Mystery, Sci-Fi, Thriller	2409

Query executed successfully.

6. Find the most-watched movie (highest view_count)

Query:

```
--Find the most-watched movie (highest view_count):

SELECT top 10
    m.title AS movie_title,
    SUM(f.view_count) AS total_views
FROM Fact_Movie f
JOIN Dim_Movie m ON f.movie_id = m.movie_id
GROUP BY m.title
ORDER BY total_views DESC;
```

Result:

100 %

Results | Messages

	movie_title	total_views
1	Death Note	1547635600
2	Sword Art Online	935289306
3	Shingeki no Kyojin	875183472
4	Code Geass: Hangyaku no Lelouch	768259806
5	Elfen Lied	756580036
6	Angel Beats!	738915489
7	Naruto	672105625
8	Fullmetal Alchemist	626601024
9	Fullmetal Alchemist: Brotherhood	603881476
10	Toradora!	589664089

Query executed successfully.

7. Analyze user engagement by subscription type (total views per subscription type)

Query:

```
--Analyze user engagement by subscription type (total views per subscription type):
SELECT
    u.subscription_type,
    SUM(CAST(f.view_count AS BIGINT)) AS total_views
FROM Fact_Movie f
JOIN Dim_User u ON f.user_id = u.user_id
GROUP BY u.subscription_type
ORDER BY total_views DESC;
```

Result:

100 %

Results | Messages

	subscription_type	total_views
1	Standard	23068290264
2	Premium	16772266818
3	Free	11698141722

Query executed successfully.