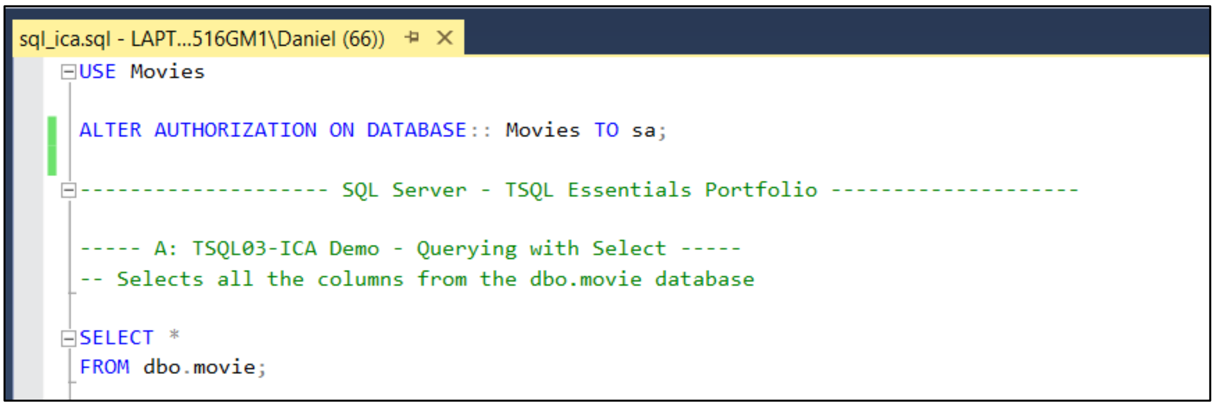
SQL Server: TSQL Module Portfolio

Name: Daniel Cyril Obon
Student ID: C2650218
Course: Computer Science (Year 2)
Assignment: Relational and NO SQL Databases (Task 1)

SQL Server - TSQL Essentials Portfolio:

A: TSQL03-ICA Demo - Querying with Select

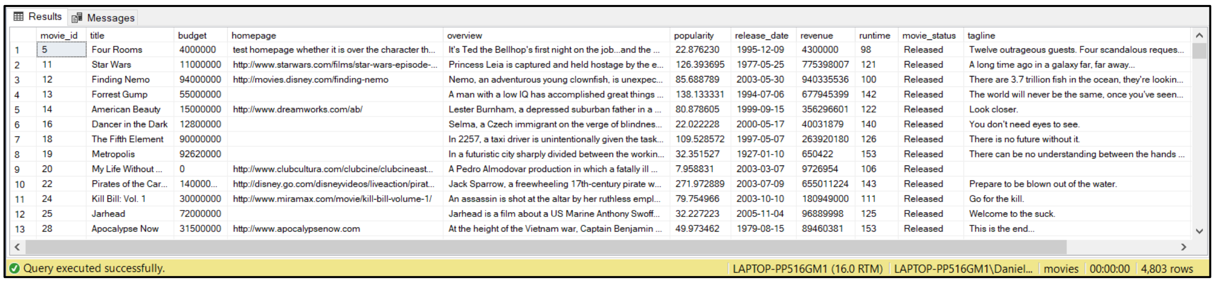
Code:



sql_ica.sql - LAPT...516GM1\Daniel (66))

```
USE Movies

ALTER AUTHORIZATION ON DATABASE:: Movies TO sa;

-------------------- SQL Server - TSQL Essentials Portfolio --------------------

----- A: TSQL03-ICA Demo - Querying with Select -----
-- Selects all the columns from the dbo.movie database

SELECT *
FROM dbo.movie;
```

Result:



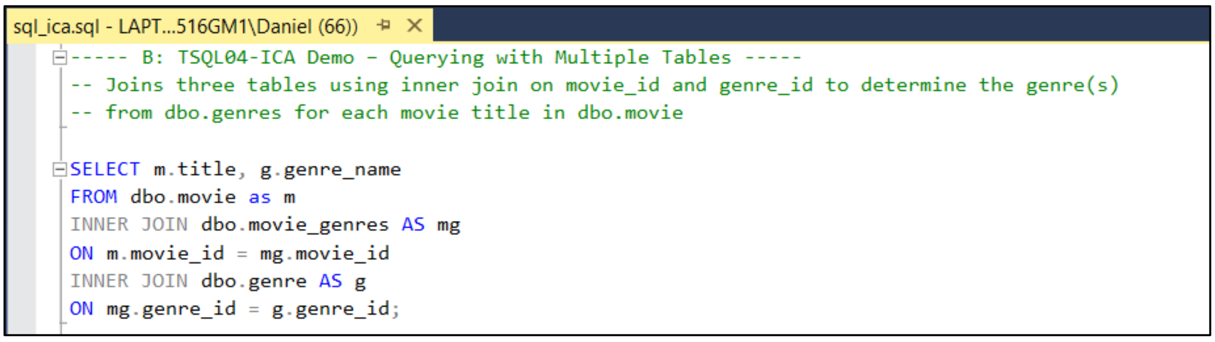
Results | Messages

	movie_id	title	budget	homepage	overview	popularity	release_date	revenue	runtime	movie_status	tagline
1	5	Four Rooms	4000000	test homepage whether it is over the character th...	It's Ted the Bellhop's first night on the job...and the ...	22.876230	1995-12-09	4300000	98	Released	Twelve outrageous guests. Four scandalous reques...
2	11	Star Wars	11000000	http://www.starwars.com/films/star-wars-episode-...	Princess Leia is captured and held hostage by the e...	126.393695	1977-05-25	775398007	121	Released	A long time ago in a galaxy far, far away...
3	12	Finding Nemo	94000000	http://movies.disney.com/finding-nemo	Nemo, an adventurous young clownfish, is unexpec...	85.688789	2003-05-30	940335536	100	Released	There are 3.7 trillion fish in the ocean, they're lookin...
4	13	Forrest Gump	55000000		A man with a low IQ has accomplished great things ...	138.133331	1994-07-06	677945399	142	Released	The world will never be the same, once you've seen...
5	14	American Beauty	15000000	http://www.dreamworks.com/ab/	Lester Burnham, a depressed suburban father in a ...	80.878605	1999-09-15	356296601	122	Released	Look closer.
6	16	Dancer in the Dark	12800000		Selma, a Czech immigrant on the verge of blindnes...	22.022228	2000-05-17	40031879	140	Released	You don't need eyes to see.
7	18	The Fifth Element	90000000		In 2257, a taxi driver is unintentionally given the task...	109.528572	1997-05-07	263920180	126	Released	There is no future without it.
8	19	Metropolis	92620000		In a futuristic city sharply divided between the workin...	32.351527	1927-01-10	650422	153	Released	There can be no understanding between the hands ...
9	20	My Life Without ...	0	http://www.clubcultura.com/clubcine/clubcineast...	A Pedro Almodovar production in which a fatally ill ...	7.958831	2003-03-07	9726954	106	Released	
10	22	Pirates of the Car...	140000...	http://disney.go.com/disneyvideos/liveaction/pirat...	Jack Sparrow, a freewheeling 17th-century pirate w...	271.972889	2003-07-09	655011224	143	Released	Prepare to be blown out of the water.
11	24	Kill Bill: Vol. 1	30000000	http://www.miramax.com/movie/kill-bill-volume-1/	An assassin is shot at the altar by her ruthless empl...	79.754966	2003-10-10	180949000	111	Released	Go for the kill.
12	25	Jarhead	72000000		Jarhead is a film about a US Marine Anthony Swoff...	32.227223	2005-11-04	96889998	125	Released	Welcome to the suck.
13	28	Apocalypse Now	31500000	http://www.apocalypsenow.com	At the height of the Vietnam war, Captain Benjamin ...	49.973462	1979-08-15	89460381	153	Released	This is the end...

```
Query executed successfully.    LAPTOP-PP516GM1 (16.0 RTM) | LAPTOP-PP516GM1\Daniel... | movies | 00:00:00 | 4,803 rows
```

B: TSQL04-ICA Demo – Querying with Multiple Tables

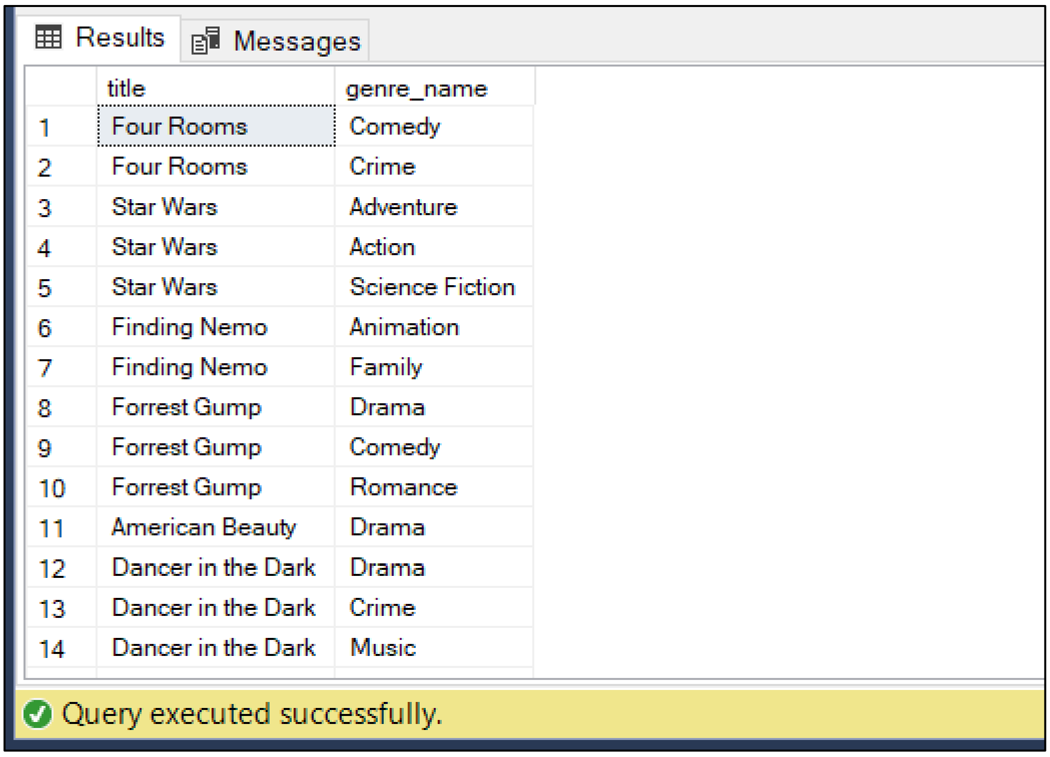
Code:



sql_ica.sql - LAPT...516GM1\Daniel (66))

```
----- B: TSQL04-ICA Demo - Querying with Multiple Tables -----
-- Joins three tables using inner join on movie_id and genre_id to determine the genre(s)
-- from dbo.genres for each movie title in dbo.movie

SELECT m.title, g.genre_name
FROM dbo.movie as m
INNER JOIN dbo.movie_genres AS mg
ON m.movie_id = mg.movie_id
INNER JOIN dbo.genre AS g
ON mg.genre_id = g.genre_id;
```

Result:



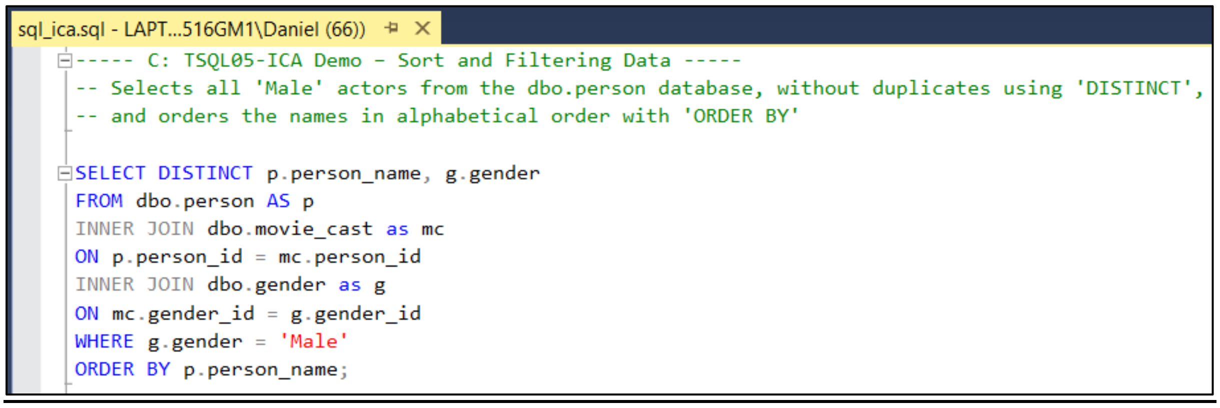
Results | Messages

	title	genre_name
1	Four Rooms	Comedy
2	Four Rooms	Crime
3	Star Wars	Adventure
4	Star Wars	Action
5	Star Wars	Science Fiction
6	Finding Nemo	Animation
7	Finding Nemo	Family
8	Forrest Gump	Drama
9	Forrest Gump	Comedy
10	Forrest Gump	Romance
11	American Beauty	Drama
12	Dancer in the Dark	Drama
13	Dancer in the Dark	Crime
14	Dancer in the Dark	Music

Query executed successfully.

C: TSQL05-ICA Demo – Sort and Filtering Data

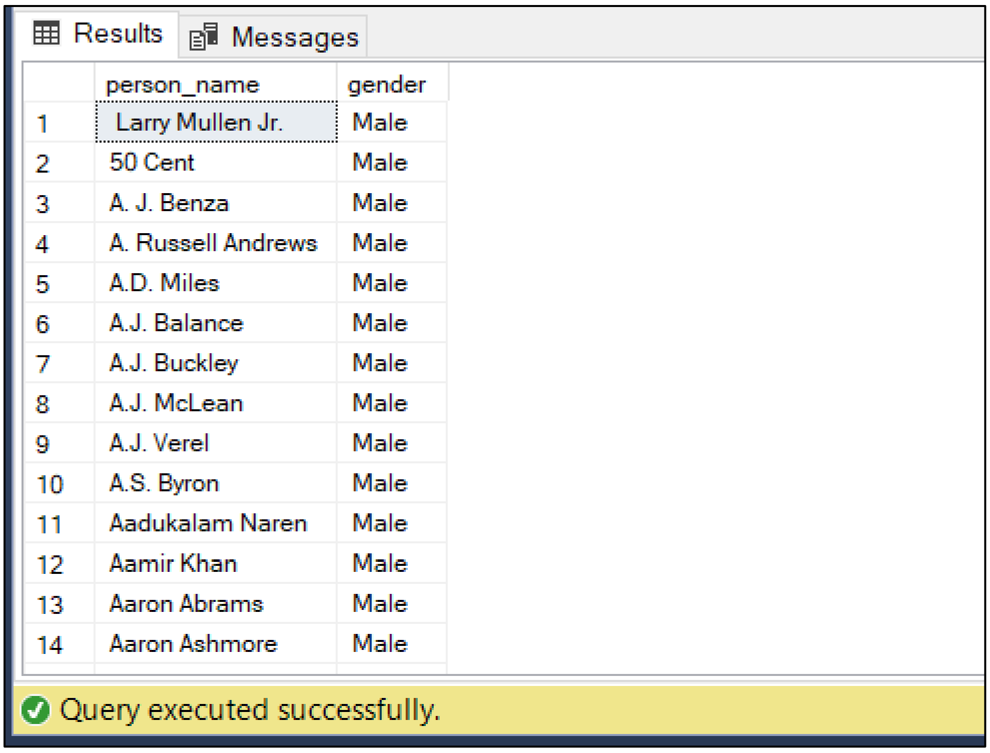
Code:



sql_ica.sql - LAPT...516GM1\Daniel (66))

```sql
----- C: TSQL05-ICA Demo - Sort and Filtering Data -----
-- Selects all 'Male' actors from the dbo.person database, without duplicates using 'DISTINCT',
-- and orders the names in alphabetical order with 'ORDER BY'

SELECT DISTINCT p.person_name, g.gender
FROM dbo.person AS p
INNER JOIN dbo.movie_cast as mc
ON p.person_id = mc.person_id
INNER JOIN dbo.gender as g
ON mc.gender_id = g.gender_id
WHERE g.gender = 'Male'
ORDER BY p.person_name;
```

Result:



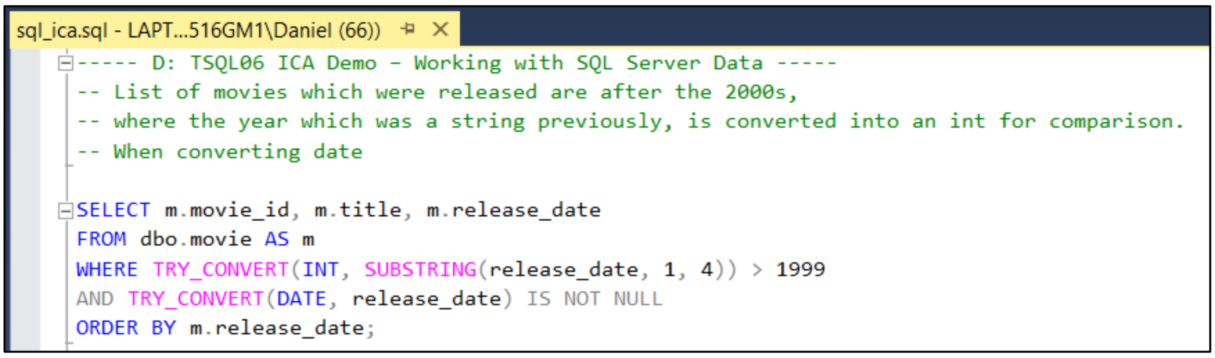
Results | Messages

	person_name	gender
1	Larry Mullen Jr.	Male
2	50 Cent	Male
3	A. J. Benza	Male
4	A. Russell Andrews	Male
5	A.D. Miles	Male
6	A.J. Balance	Male
7	A.J. Buckley	Male
8	A.J. McLean	Male
9	A.J. Verel	Male
10	A.S. Byron	Male
11	Aadukalam Naren	Male
12	Aamir Khan	Male
13	Aaron Abrams	Male
14	Aaron Ashmore	Male

Query executed successfully.

D: TSQL06 ICA Demo – Working with SQL Server Data

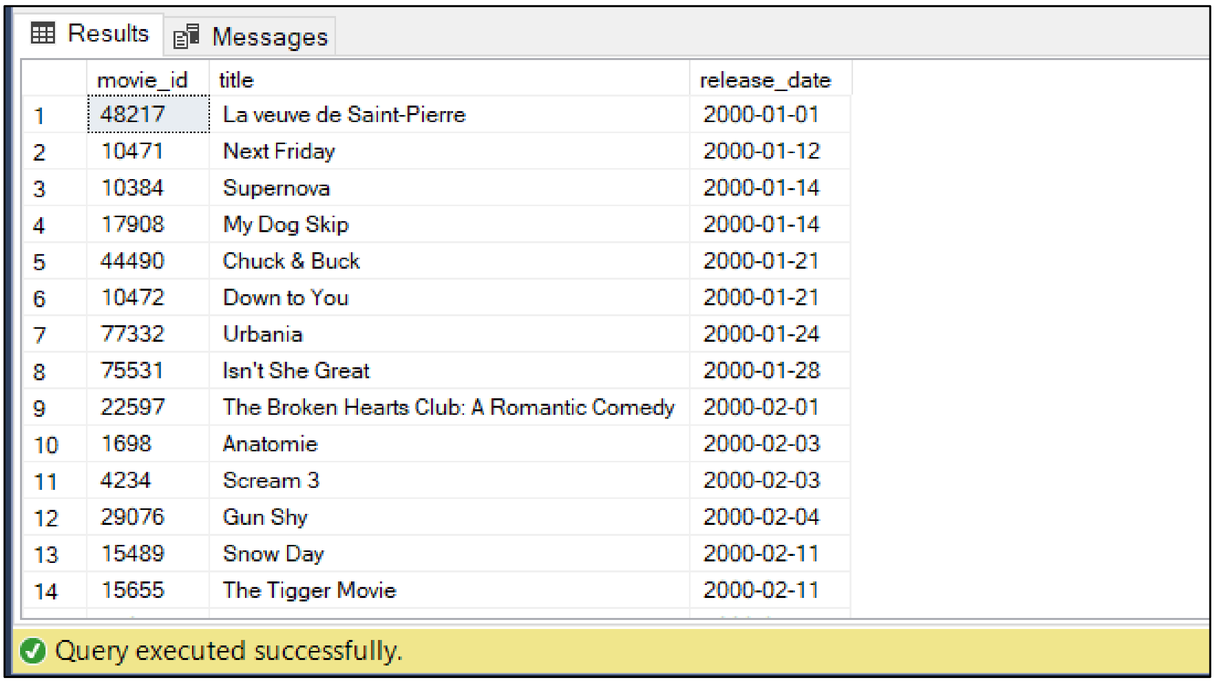
Code:



sql_ica.sql - LAPT...516GM1\Daniel (66))

```sql
----- D: TSQL06 ICA Demo - Working with SQL Server Data -----
-- List of movies which were released are after the 2000s,
-- where the year which was a string previously, is converted into an int for comparison.
-- When converting date

SELECT m.movie_id, m.title, m.release_date
FROM dbo.movie AS m
WHERE TRY_CONVERT(INT, SUBSTRING(release_date, 1, 4)) > 1999
AND TRY_CONVERT(DATE, release_date) IS NOT NULL
ORDER BY m.release_date;
```

Result:



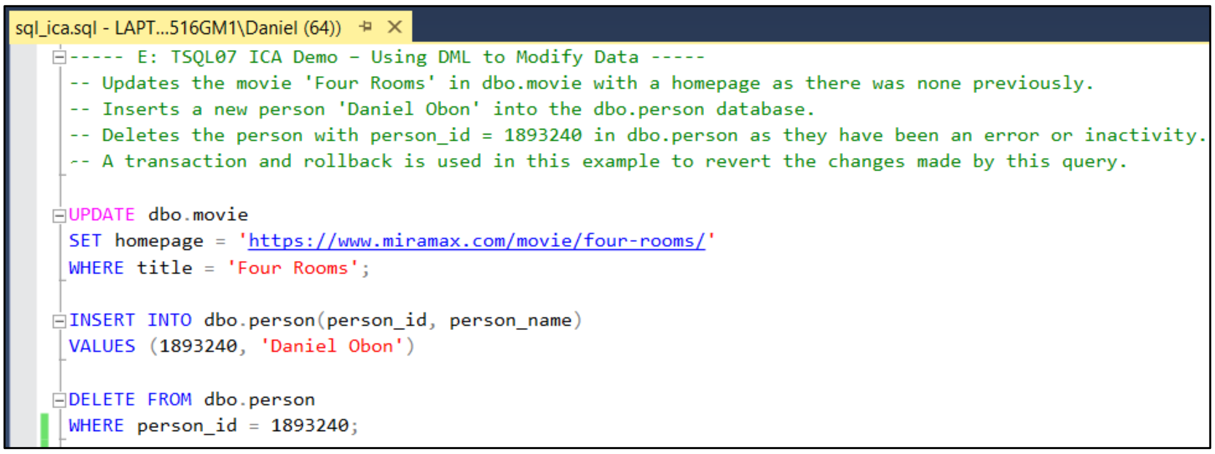
Results | Messages

	movie_id	title	release_date
1	48217	La veuve de Saint-Pierre	2000-01-01
2	10471	Next Friday	2000-01-12
3	10384	Supernova	2000-01-14
4	17908	My Dog Skip	2000-01-14
5	44490	Chuck & Buck	2000-01-21
6	10472	Down to You	2000-01-21
7	77332	Urbania	2000-01-24
8	75531	Isn't She Great	2000-01-28
9	22597	The Broken Hearts Club: A Romantic Comedy	2000-02-01
10	1698	Anatomie	2000-02-03
11	4234	Scream 3	2000-02-03
12	29076	Gun Shy	2000-02-04
13	15489	Snow Day	2000-02-11
14	15655	The Tigger Movie	2000-02-11

Query executed successfully.

E: TSQL07 ICA Demo – Using DML to Modify Data

Code:

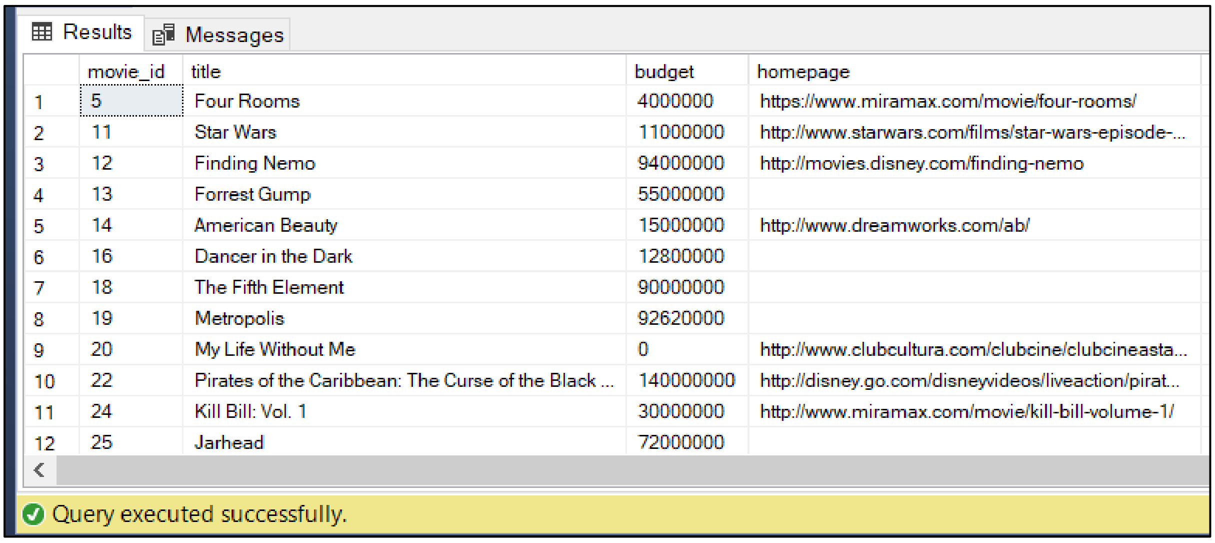


sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
----- E: TSQL07 ICA Demo - Using DML to Modify Data -----
-- Updates the movie 'Four Rooms' in dbo.movie with a homepage as there was none previously.
-- Inserts a new person 'Daniel Obon' into the dbo.person database.
-- Deletes the person with person_id = 1893240 in dbo.person as they have been an error or inactivity.
-- A transaction and rollback is used in this example to revert the changes made by this query.

UPDATE dbo.movie
SET homepage = 'https://www.miramax.com/movie/four-rooms/'
WHERE title = 'Four Rooms';

INSERT INTO dbo.person(person_id, person_name)
VALUES (1893240, 'Daniel Obon')

DELETE FROM dbo.person
WHERE person_id = 1893240;
```

Result:

[Update Result]

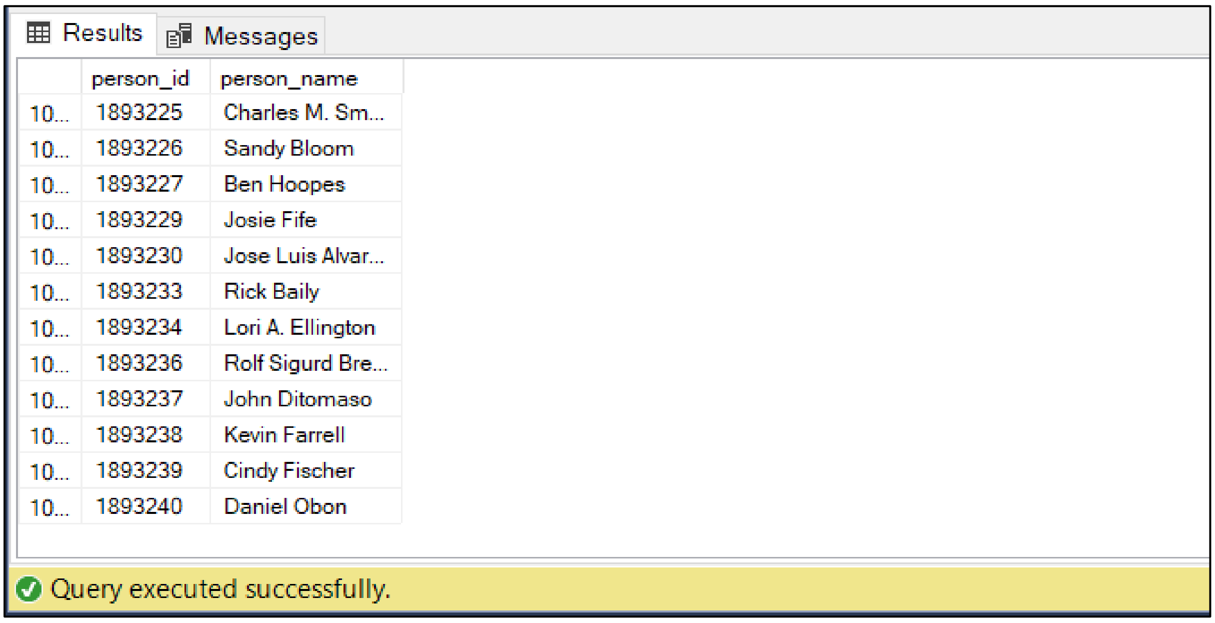


Results | Messages

	movie_id	title	budget	homepage
1	5	Four Rooms	4000000	https://www.miramax.com/movie/four-rooms/
2	11	Star Wars	11000000	http://www.starwars.com/films/star-wars-episode-...
3	12	Finding Nemo	94000000	http://movies.disney.com/finding-nemo
4	13	Forrest Gump	55000000	
5	14	American Beauty	15000000	http://www.dreamworks.com/ab/
6	16	Dancer in the Dark	12800000	
7	18	The Fifth Element	90000000	
8	19	Metropolis	92620000	
9	20	My Life Without Me	0	http://www.clubcultura.com/clubcine/clubcineasta...
10	22	Pirates of the Caribbean: The Curse of the Black ...	140000000	http://disney.go.com/disneyvideos/liveaction/pirat...
11	24	Kill Bill: Vol. 1	30000000	http://www.miramax.com/movie/kill-bill-volume-1/
12	25	Jarhead	72000000	

Query executed successfully.

[Insert Result]

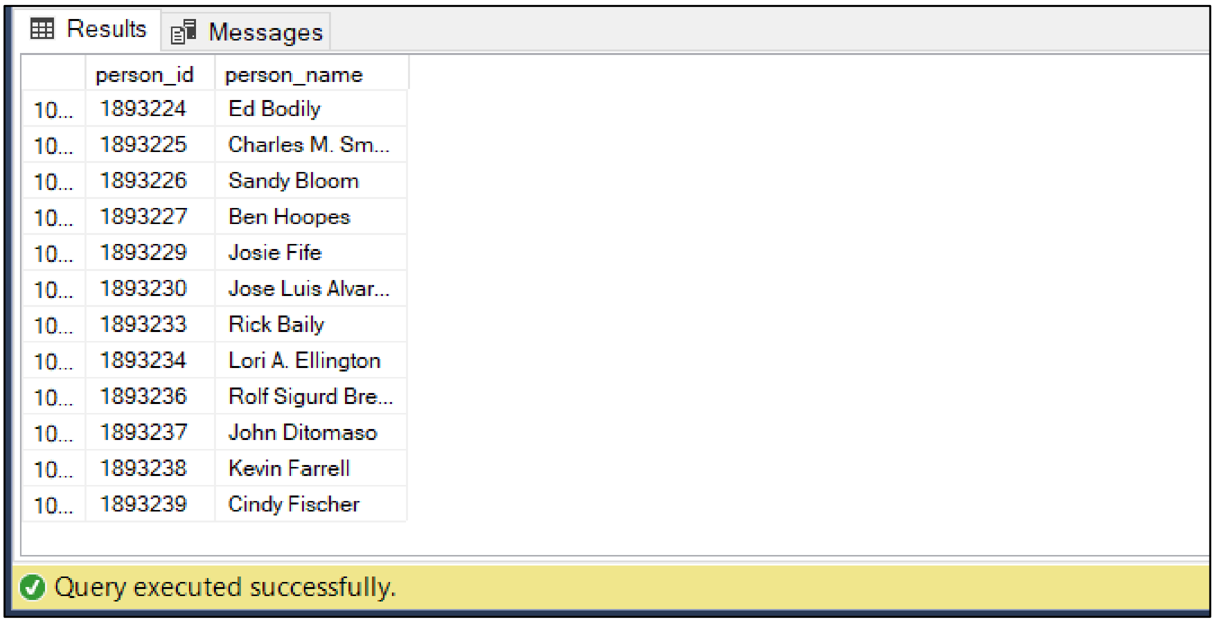


Results | Messages

	person_id	person_name
10...	1893225	Charles M. Sm...
10...	1893226	Sandy Bloom
10...	1893227	Ben Hoopes
10...	1893229	Josie Fife
10...	1893230	Jose Luis Alvar...
10...	1893233	Rick Baily
10...	1893234	Lori A. Ellington
10...	1893236	Rolf Sigurd Bre...
10...	1893237	John Ditomaso
10...	1893238	Kevin Farrell
10...	1893239	Cindy Fischer
10...	1893240	Daniel Obon

Query executed successfully.

[Delete Result]



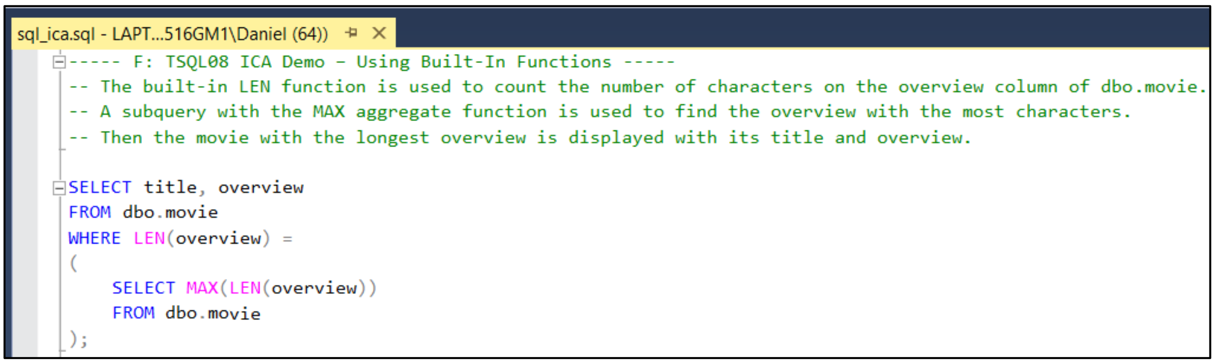
Results | Messages

	person_id	person_name
10...	1893224	Ed Bodily
10...	1893225	Charles M. Sm...
10...	1893226	Sandy Bloom
10...	1893227	Ben Hoopes
10...	1893229	Josie Fife
10...	1893230	Jose Luis Alvar...
10...	1893233	Rick Baily
10...	1893234	Lori A. Ellington
10...	1893236	Rolf Sigurd Bre...
10...	1893237	John Ditomaso
10...	1893238	Kevin Farrell
10...	1893239	Cindy Fischer

Query executed successfully.

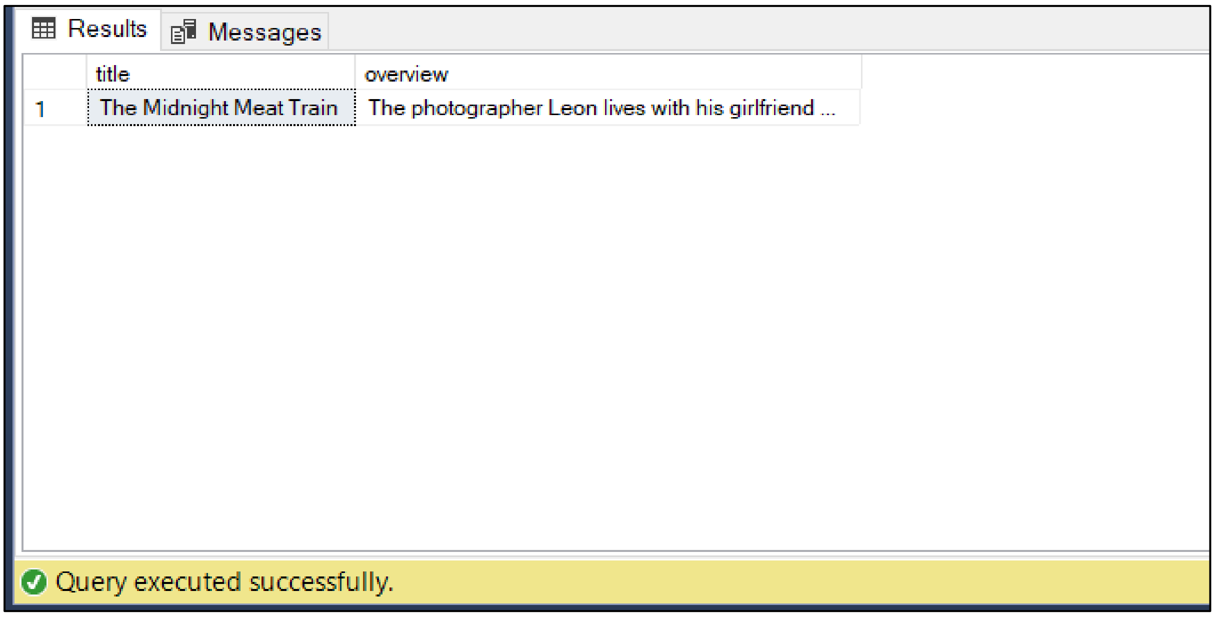
F: TSQL08 ICA Demo – Using Built-In Functions

Code:



```sql
----- F: TSQL08 ICA Demo - Using Built-In Functions -----
-- The built-in LEN function is used to count the number of characters on the overview column of dbo.movie.
-- A subquery with the MAX aggregate function is used to find the overview with the most characters.
-- Then the movie with the longest overview is displayed with its title and overview.

SELECT title, overview
FROM dbo.movie
WHERE LEN(overview) =
(
    SELECT MAX(LEN(overview))
    FROM dbo.movie
);
```

Result:



	title	overview
1	The Midnight Meat Train	The photographer Leon lives with his girlfriend ...

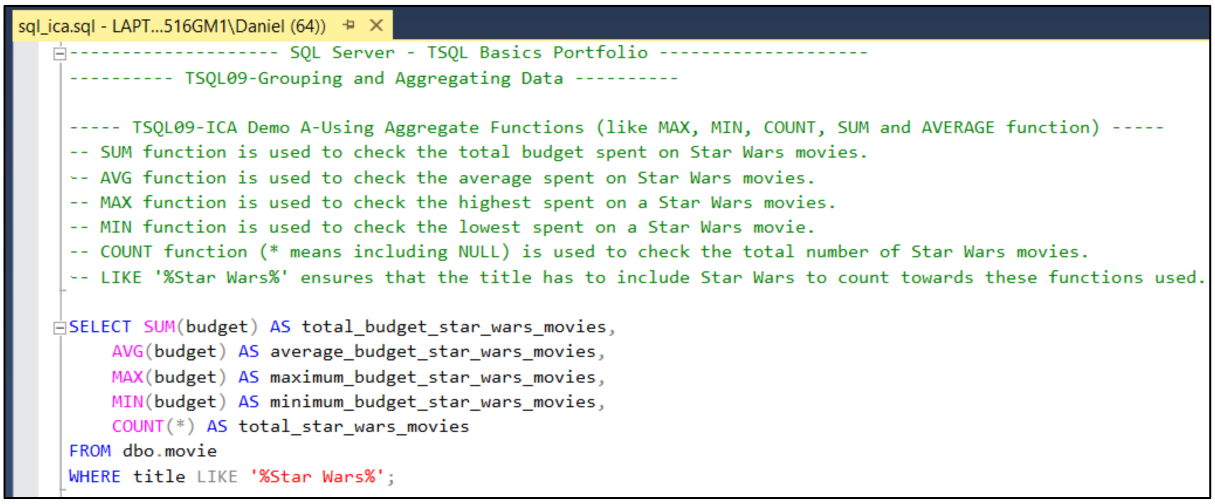
Query executed successfully.

SQL Server - TSQL Basics Portfolio:

TSQL09-Grouping and Aggregating Data

TSQL09-ICA Demo A-Using Aggregate Functions (like MAX, MIN, COUNT, SUM and AVERAGE function)

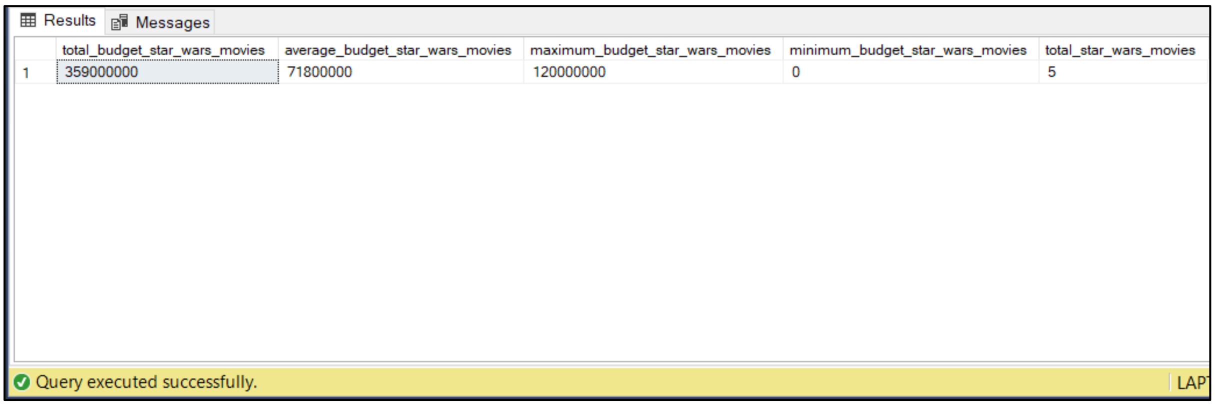
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
-------------------- SQL Server - TSQL Basics Portfolio --------------------
---------- TSQL09-Grouping and Aggregating Data ----------

----- TSQL09-ICA Demo A-Using Aggregate Functions (like MAX, MIN, COUNT, SUM and AVERAGE function) -----
-- SUM function is used to check the total budget spent on Star Wars movies.
-- AVG function is used to check the average spent on Star Wars movies.
-- MAX function is used to check the highest spent on a Star Wars movies.
-- MIN function is used to check the lowest spent on a Star Wars movie.
-- COUNT function (* means including NULL) is used to check the total number of Star Wars movies.
-- LIKE '%Star Wars%' ensures that the title has to include Star Wars to count towards these functions used.

SELECT SUM(budget) AS total_budget_star_wars_movies,
    AVG(budget) AS average_budget_star_wars_movies,
    MAX(budget) AS maximum_budget_star_wars_movies,
    MIN(budget) AS minimum_budget_star_wars_movies,
    COUNT(*) AS total_star_wars_movies
FROM dbo.movie
WHERE title LIKE '%Star Wars%';
```

Result:



	total_budget_star_wars_movies	average_budget_star_wars_movies	maximum_budget_star_wars_movies	minimum_budget_star_wars_movies	total_star_wars_movies
1	359000000	71800000	120000000	0	5

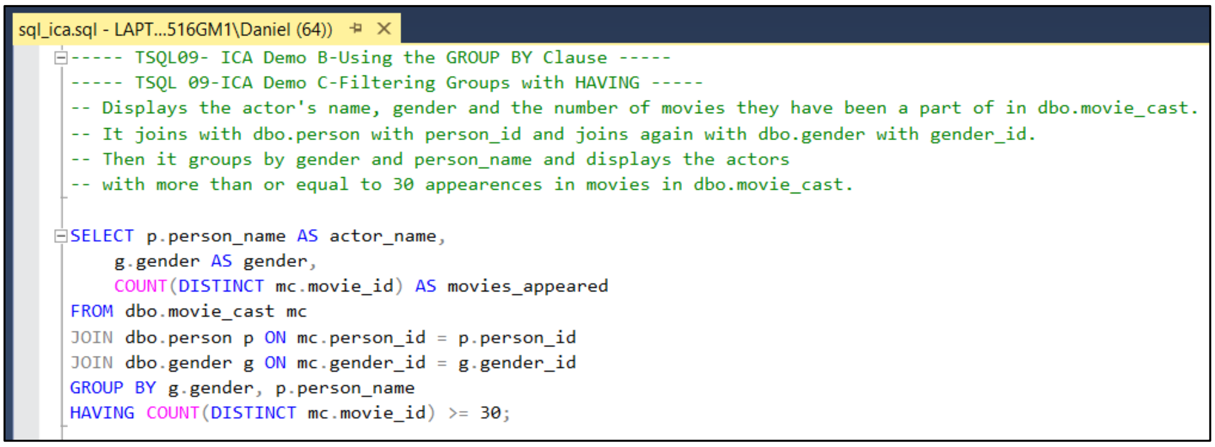
Query executed successfully.

TSQL09- ICA Demo B-Using the GROUP BY Clause

TSQL 09-ICA Demo C-Filtering Groups with HAVING

[TSQL09- ICA Demo B & TSQL 09-ICA Demo C Both Done Together]

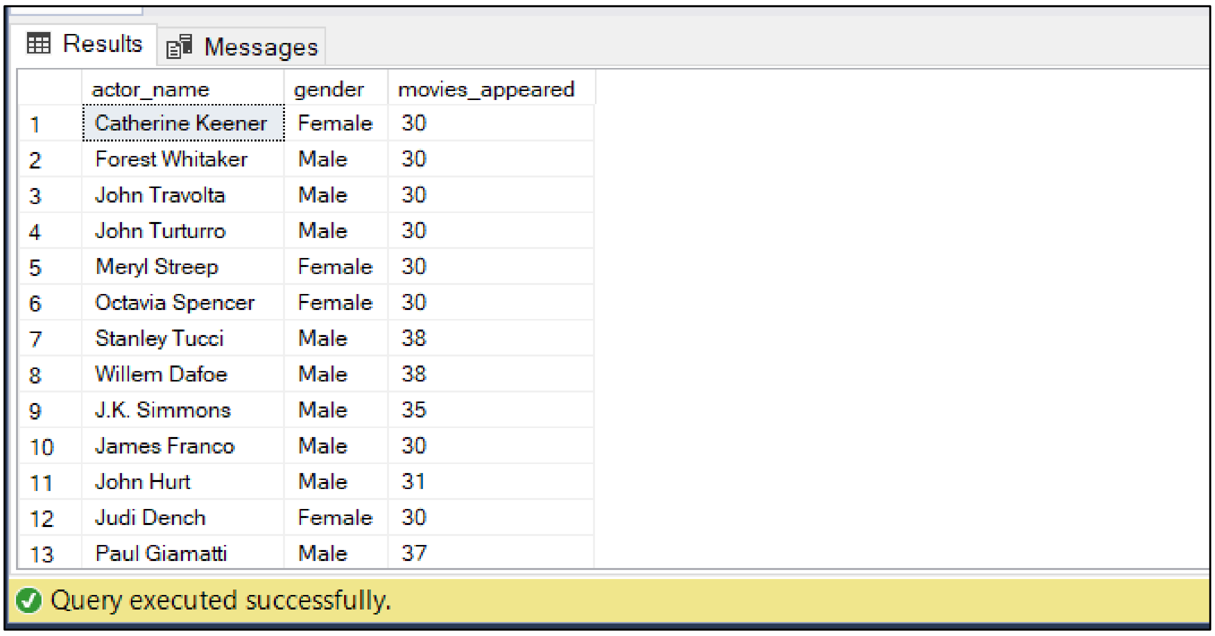
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```
----- TSQL09- ICA Demo B-Using the GROUP BY Clause -----
----- TSQL 09-ICA Demo C-Filtering Groups with HAVING -----
-- Displays the actor's name, gender and the number of movies they have been a part of in dbo.movie_cast.
-- It joins with dbo.person with person_id and joins again with dbo.gender with gender_id.
-- Then it groups by gender and person_name and displays the actors
-- with more than or equal to 30 appearences in movies in dbo.movie_cast.

SELECT p.person_name AS actor_name,
    g.gender AS gender,
    COUNT(DISTINCT mc.movie_id) AS movies_appeared
FROM dbo.movie_cast mc
JOIN dbo.person p ON mc.person_id = p.person_id
JOIN dbo.gender g ON mc.gender_id = g.gender_id
GROUP BY g.gender, p.person_name
HAVING COUNT(DISTINCT mc.movie_id) >= 30;
```

Result:



Results | Messages

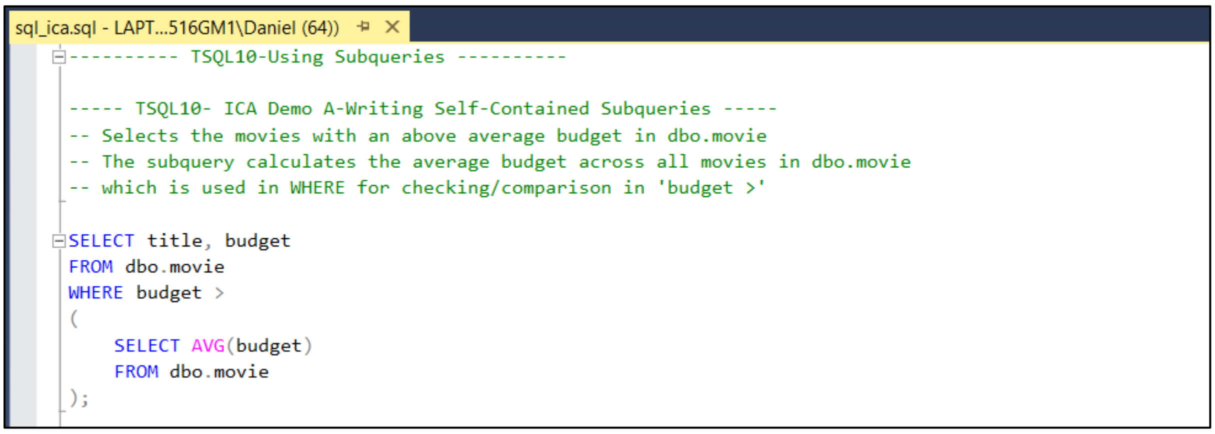
	actor_name	gender	movies_appeared
1	Catherine Keener	Female	30
2	Forest Whitaker	Male	30
3	John Travolta	Male	30
4	John Turturro	Male	30
5	Meryl Streep	Female	30
6	Octavia Spencer	Female	30
7	Stanley Tucci	Male	38
8	Willem Dafoe	Male	38
9	J.K. Simmons	Male	35
10	James Franco	Male	30
11	John Hurt	Male	31
12	Judi Dench	Female	30
13	Paul Giamatti	Male	37

Query executed successfully.

TSQL10-Using Subqueries

TSQL10- ICA Demo A-Writing Self-Contained Subqueries

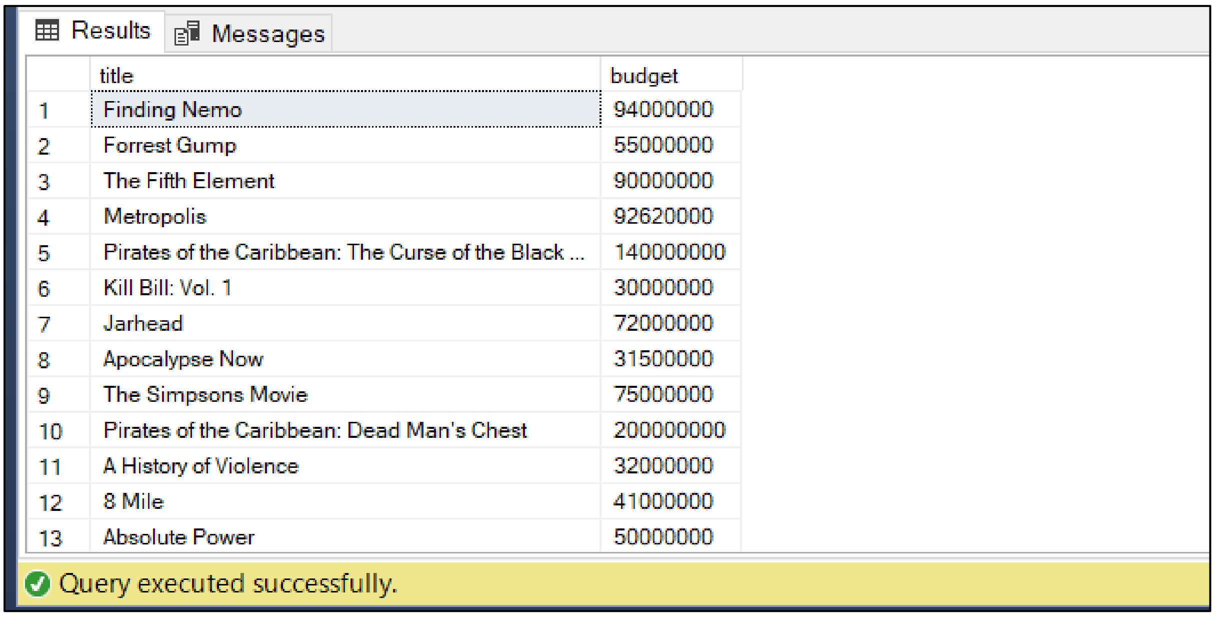
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```
---------- TSQL10-Using Subqueries ----------

----- TSQL10- ICA Demo A-Writing Self-Contained Subqueries -----
-- Selects the movies with an above average budget in dbo.movie
-- The subquery calculates the average budget across all movies in dbo.movie
-- which is used in WHERE for checking/comparison in 'budget >'

SELECT title, budget
FROM dbo.movie
WHERE budget >
(
    SELECT AVG(budget)
    FROM dbo.movie
);
```

Result:



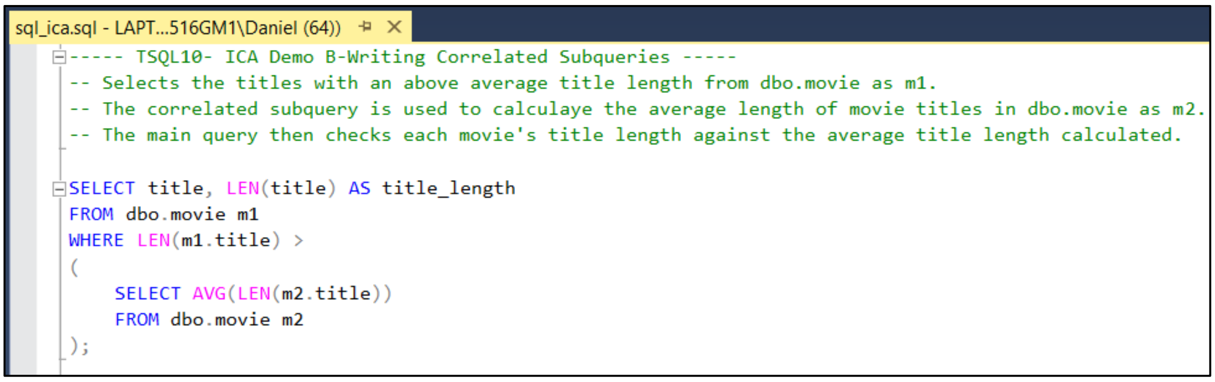
Results | Messages

	title	budget
1	Finding Nemo	94000000
2	Forrest Gump	55000000
3	The Fifth Element	90000000
4	Metropolis	92620000
5	Pirates of the Caribbean: The Curse of the Black ...	140000000
6	Kill Bill: Vol. 1	30000000
7	Jarhead	72000000
8	Apocalypse Now	31500000
9	The Simpsons Movie	75000000
10	Pirates of the Caribbean: Dead Man's Chest	200000000
11	A History of Violence	32000000
12	8 Mile	41000000
13	Absolute Power	50000000

Query executed successfully.

TSQL10- ICA Demo B-Writing Correlated Subqueries

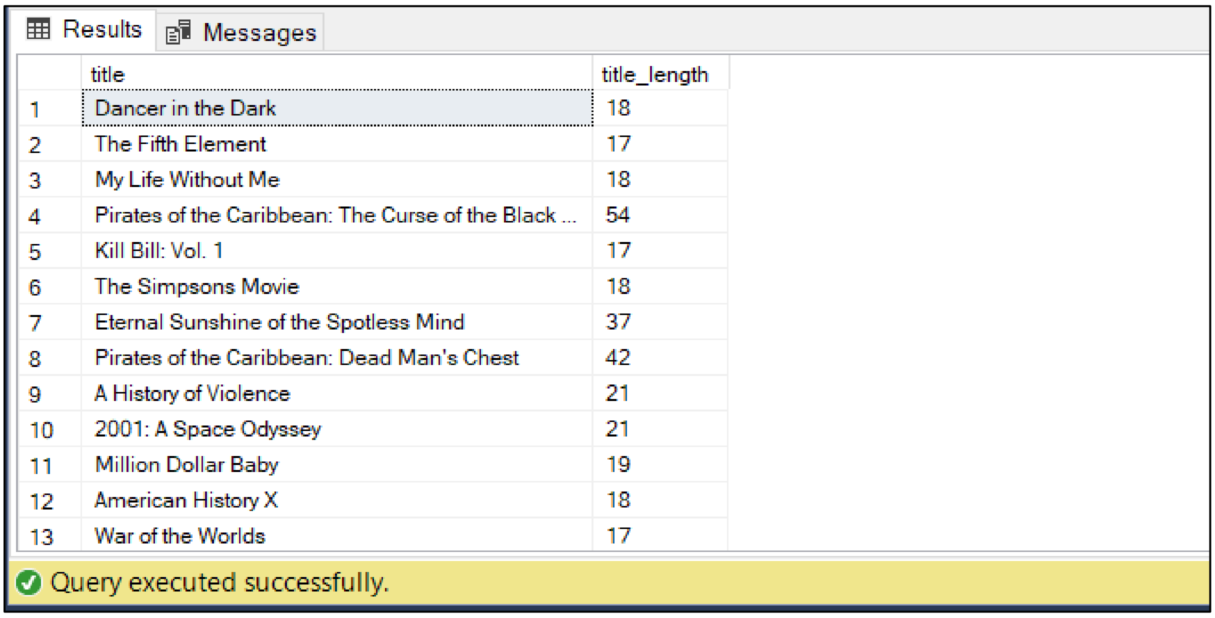
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```
----- TSQL10- ICA Demo B-Writing Correlated Subqueries -----
-- Selects the titles with an above average title length from dbo.movie as m1.
-- The correlated subquery is used to calculaye the average length of movie titles in dbo.movie as m2.
-- The main query then checks each movie's title length against the average title length calculated.

SELECT title, LEN(title) AS title_length
FROM dbo.movie m1
WHERE LEN(m1.title) >
(
    SELECT AVG(LEN(m2.title))
    FROM dbo.movie m2
);
```

Result:



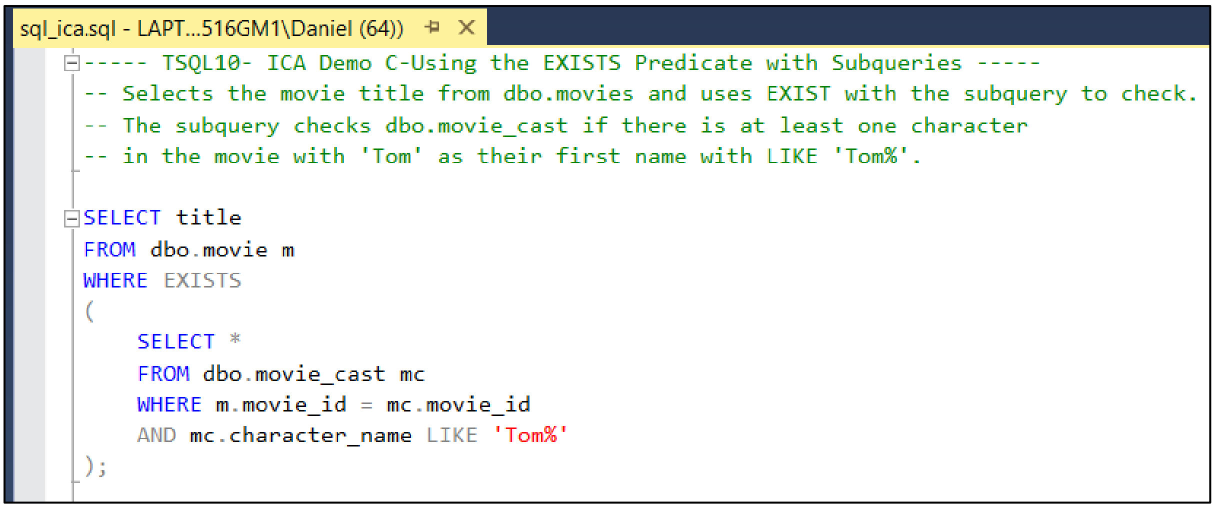
Results | Messages

	title	title_length
1	Dancer in the Dark	18
2	The Fifth Element	17
3	My Life Without Me	18
4	Pirates of the Caribbean: The Curse of the Black ...	54
5	Kill Bill: Vol. 1	17
6	The Simpsons Movie	18
7	Eternal Sunshine of the Spotless Mind	37
8	Pirates of the Caribbean: Dead Man's Chest	42
9	A History of Violence	21
10	2001: A Space Odyssey	21
11	Million Dollar Baby	19
12	American History X	18
13	War of the Worlds	17

Query executed successfully.

TSQL10- ICA Demo C-Using the EXISTS Predicate with Subqueries

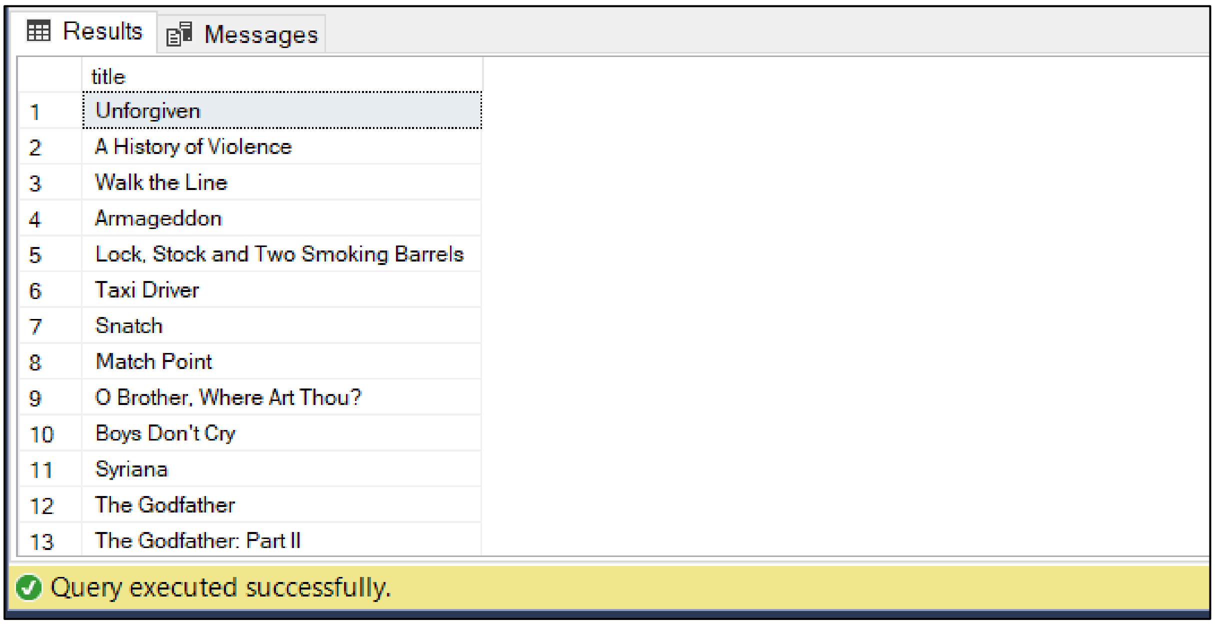
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
----- TSQL10- ICA Demo C-Using the EXISTS Predicate with Subqueries -----
-- Selects the movie title from dbo.movies and uses EXIST with the subquery to check.
-- The subquery checks dbo.movie_cast if there is at least one character
-- in the movie with 'Tom' as their first name with LIKE 'Tom%'.

SELECT title
FROM dbo.movie m
WHERE EXISTS
(
    SELECT *
    FROM dbo.movie_cast mc
    WHERE m.movie_id = mc.movie_id
    AND mc.character_name LIKE 'Tom%'
);
```

Result:



	title
1	Unforgiven
2	A History of Violence
3	Walk the Line
4	Armageddon
5	Lock, Stock and Two Smoking Barrels
6	Taxi Driver
7	Snatch
8	Match Point
9	O Brother, Where Art Thou?
10	Boys Don't Cry
11	Syriana
12	The Godfather
13	The Godfather: Part II

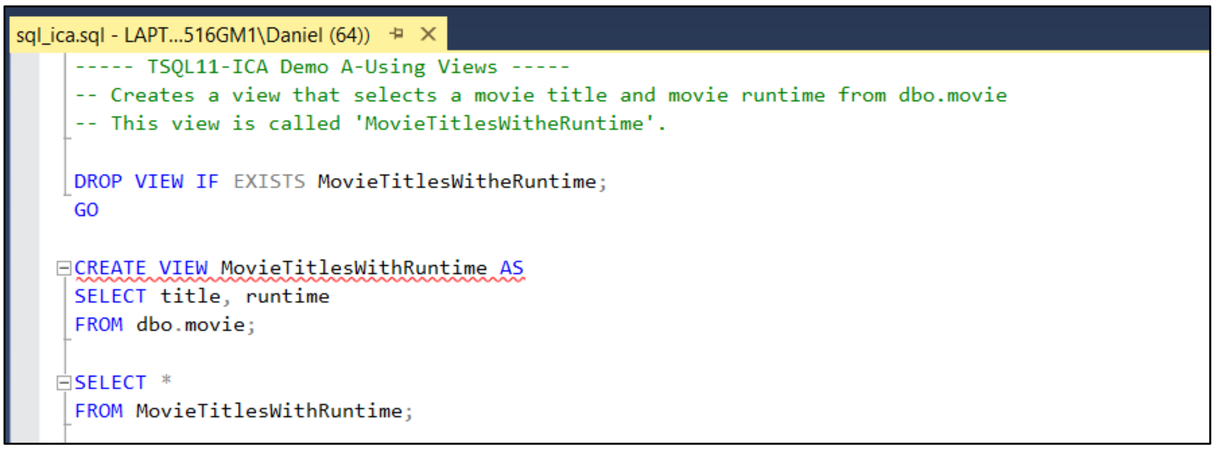
Query executed successfully.

SQL Server - TSQL Intermediate Portfolio:

TSQL11-Using Table Expressions

TSQL11-ICA Demo A-Using Views

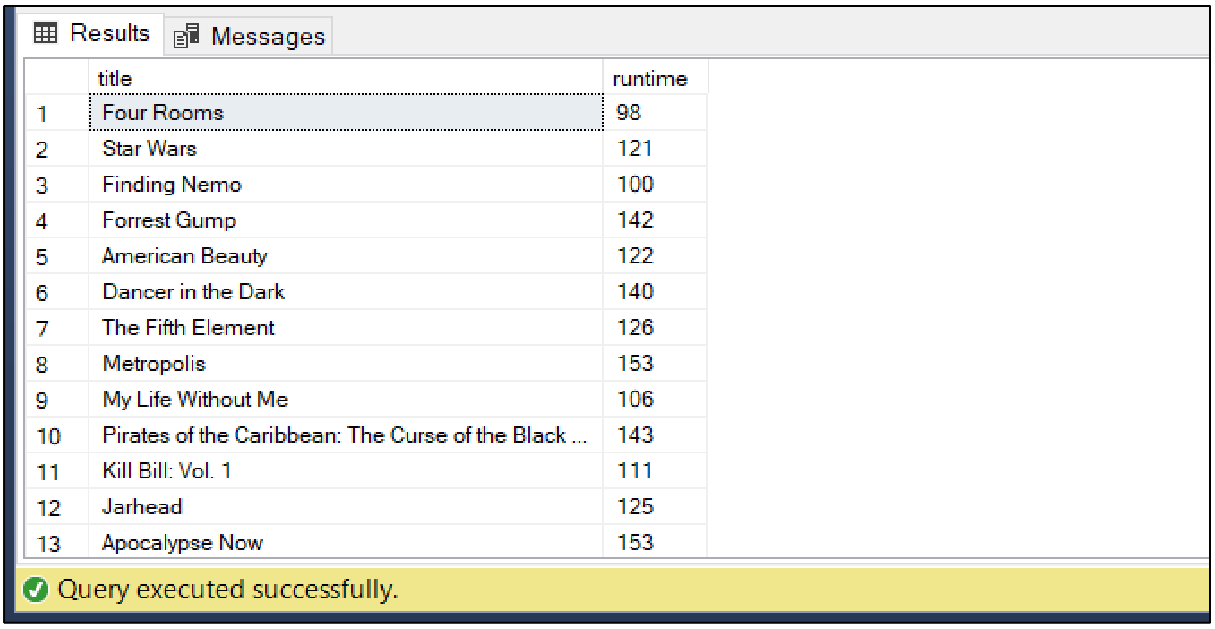
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
----- TSQL11-ICA Demo A-Using Views -----
-- Creates a view that selects a movie title and movie runtime from dbo.movie
-- This view is called 'MovieTitlesWitheRuntime'.

DROP VIEW IF EXISTS MovieTitlesWitheRuntime;
GO

CREATE VIEW MovieTitlesWithRuntime AS
SELECT title, runtime
FROM dbo.movie;

SELECT *
FROM MovieTitlesWithRuntime;
```

Result:



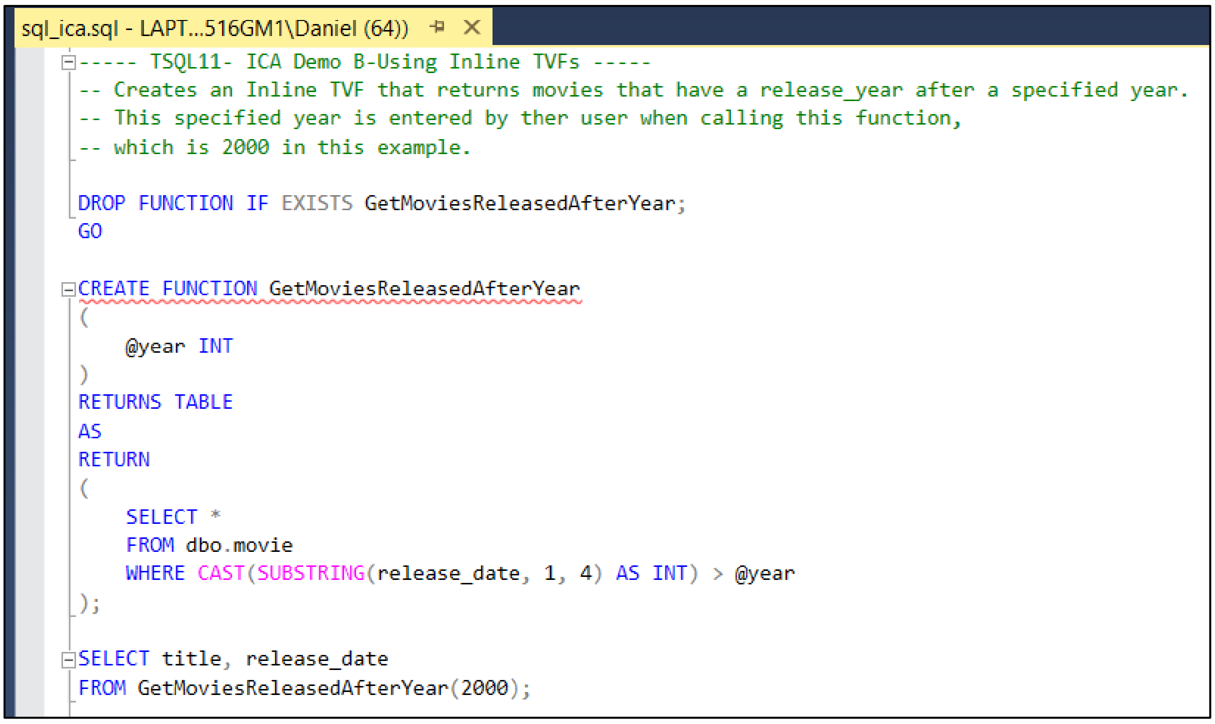
Results | Messages

	title	runtime
1	Four Rooms	98
2	Star Wars	121
3	Finding Nemo	100
4	Forrest Gump	142
5	American Beauty	122
6	Dancer in the Dark	140
7	The Fifth Element	126
8	Metropolis	153
9	My Life Without Me	106
10	Pirates of the Caribbean: The Curse of the Black ...	143
11	Kill Bill: Vol. 1	111
12	Jarhead	125
13	Apocalypse Now	153

Query executed successfully.

TSQL11- ICA Demo B-Using Inline TVFs

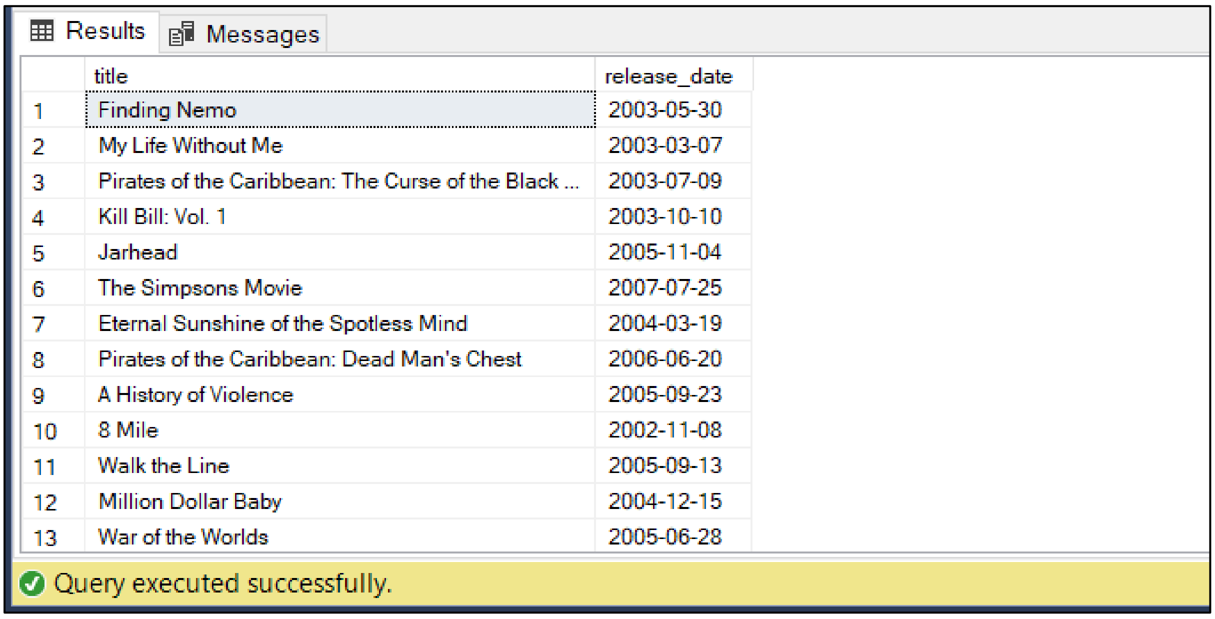
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```
----- TSQL11- ICA Demo B-Using Inline TVFs -----
-- Creates an Inline TVF that returns movies that have a release_year after a specified year.
-- This specified year is entered by ther user when calling this function,
-- which is 2000 in this example.

DROP FUNCTION IF EXISTS GetMoviesReleasedAfterYear;
GO

CREATE FUNCTION GetMoviesReleasedAfterYear
(
    @year INT
)
RETURNS TABLE
AS
RETURN
(
    SELECT *
    FROM dbo.movie
    WHERE CAST(SUBSTRING(release_date, 1, 4) AS INT) > @year
);

SELECT title, release_date
FROM GetMoviesReleasedAfterYear(2000);
```

Result:



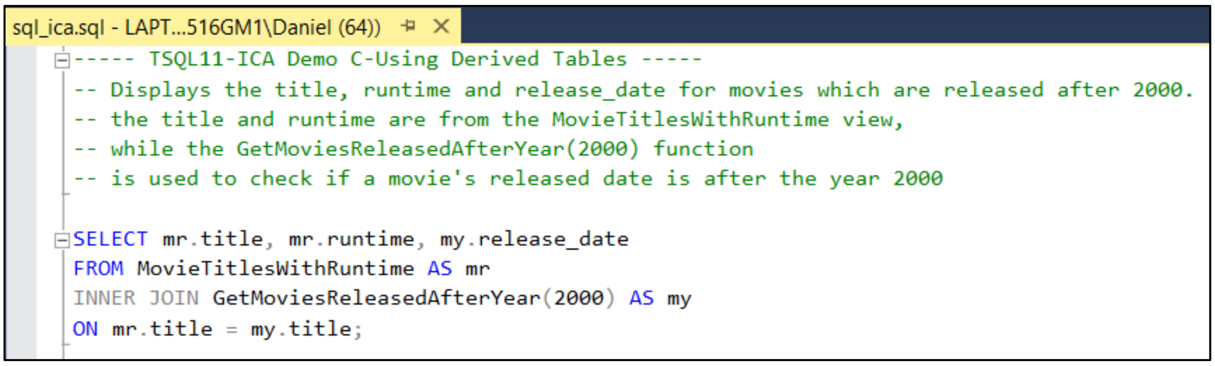
Results | Messages

	title	release_date
1	Finding Nemo	2003-05-30
2	My Life Without Me	2003-03-07
3	Pirates of the Caribbean: The Curse of the Black ...	2003-07-09
4	Kill Bill: Vol. 1	2003-10-10
5	Jarhead	2005-11-04
6	The Simpsons Movie	2007-07-25
7	Eternal Sunshine of the Spotless Mind	2004-03-19
8	Pirates of the Caribbean: Dead Man's Chest	2006-06-20
9	A History of Violence	2005-09-23
10	8 Mile	2002-11-08
11	Walk the Line	2005-09-13
12	Million Dollar Baby	2004-12-15
13	War of the Worlds	2005-06-28

Query executed successfully.

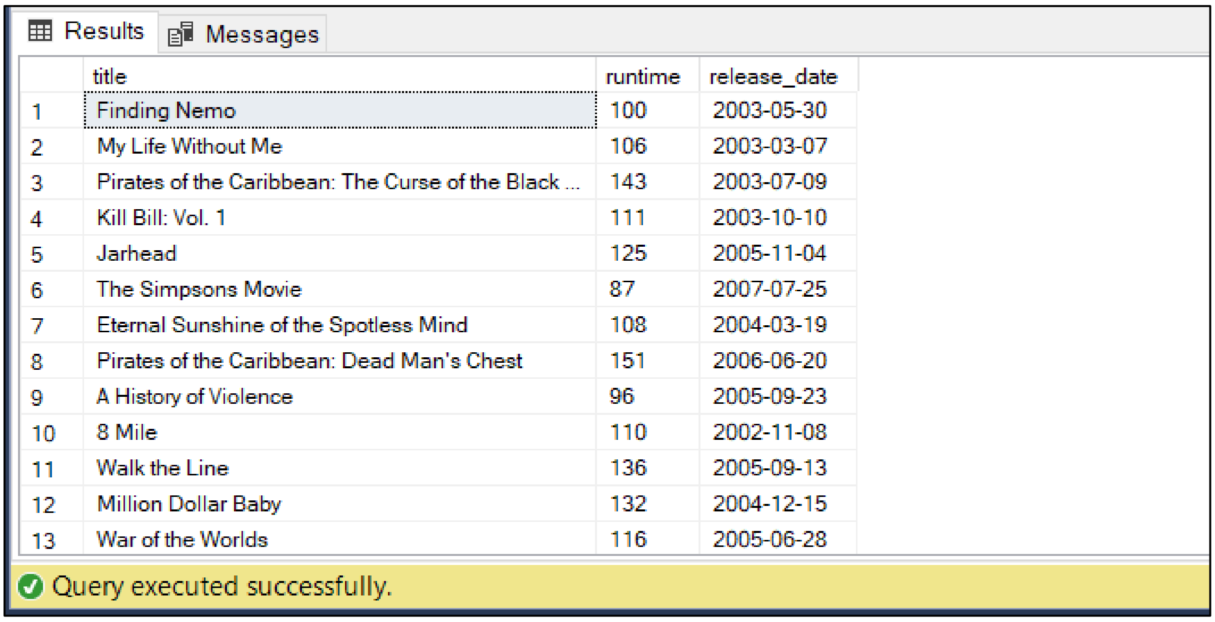
TSQL11-ICA Demo C-Using Derived Tables

Code:



```sql
----- TSQL11-ICA Demo C-Using Derived Tables -----
-- Displays the title, runtime and release_date for movies which are released after 2000.
-- the title and runtime are from the MovieTitlesWithRuntime view,
-- while the GetMoviesReleasedAfterYear(2000) function
-- is used to check if a movie's released date is after the year 2000

SELECT mr.title, mr.runtime, my.release_date
FROM MovieTitlesWithRuntime AS mr
INNER JOIN GetMoviesReleasedAfterYear(2000) AS my
ON mr.title = my.title;
```

Result:

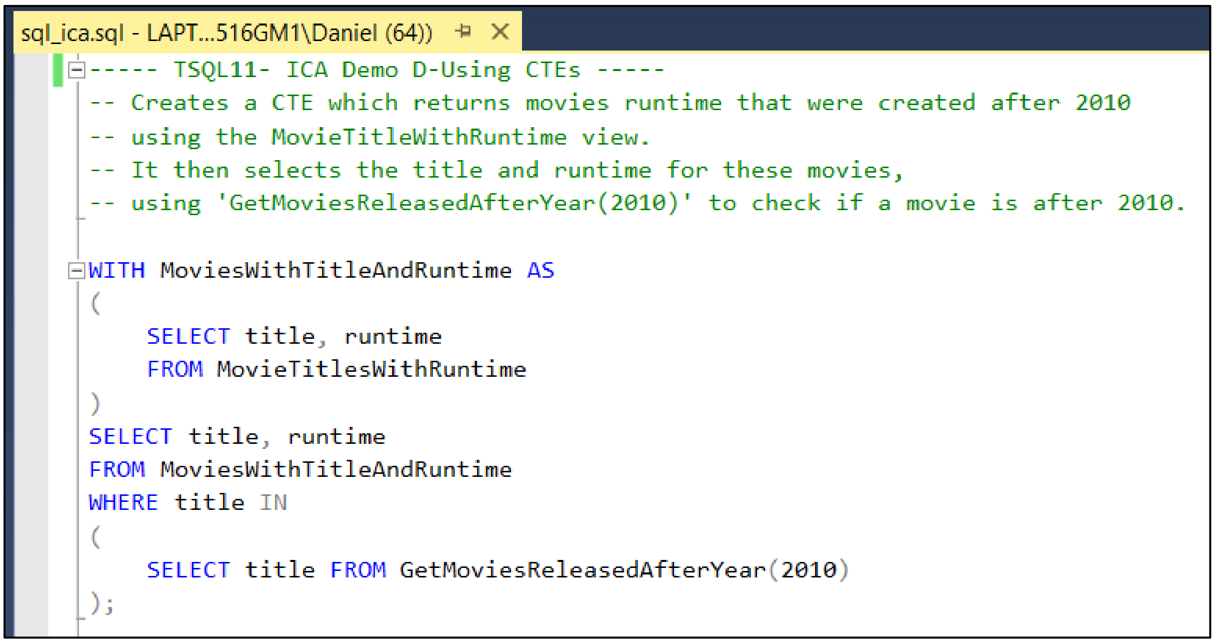


	title	runtime	release_date
1	Finding Nemo	100	2003-05-30
2	My Life Without Me	106	2003-03-07
3	Pirates of the Caribbean: The Curse of the Black ...	143	2003-07-09
4	Kill Bill: Vol. 1	111	2003-10-10
5	Jarhead	125	2005-11-04
6	The Simpsons Movie	87	2007-07-25
7	Eternal Sunshine of the Spotless Mind	108	2004-03-19
8	Pirates of the Caribbean: Dead Man's Chest	151	2006-06-20
9	A History of Violence	96	2005-09-23
10	8 Mile	110	2002-11-08
11	Walk the Line	136	2005-09-13
12	Million Dollar Baby	132	2004-12-15
13	War of the Worlds	116	2005-06-28

Query executed successfully.

TSQL11- CA Demo D-Using CTEs

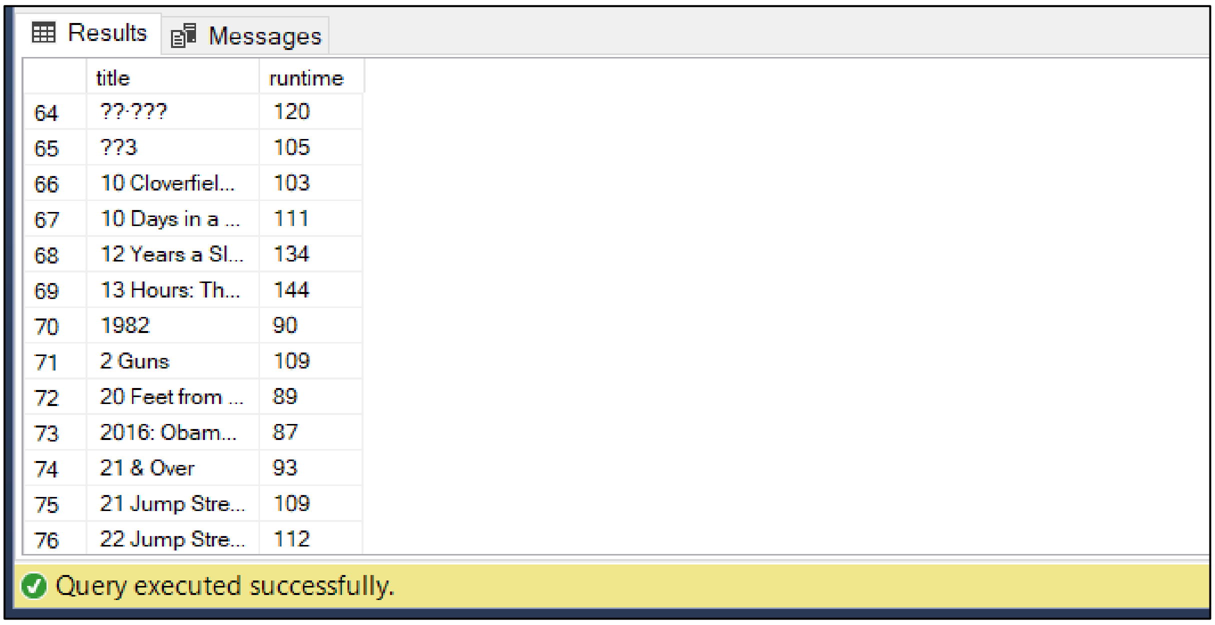
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
----- TSQL11- ICA Demo D-Using CTEs -----
-- Creates a CTE which returns movies runtime that were created after 2010
-- using the MovieTitleWithRuntime view.
-- It then selects the title and runtime for these movies,
-- using 'GetMoviesReleasedAfterYear(2010)' to check if a movie is after 2010.

WITH MoviesWithTitleAndRuntime AS
(
    SELECT title, runtime
    FROM MovieTitlesWithRuntime
)
SELECT title, runtime
FROM MoviesWithTitleAndRuntime
WHERE title IN
(
    SELECT title FROM GetMoviesReleasedAfterYear(2010)
);
```

Result:



Results | Messages

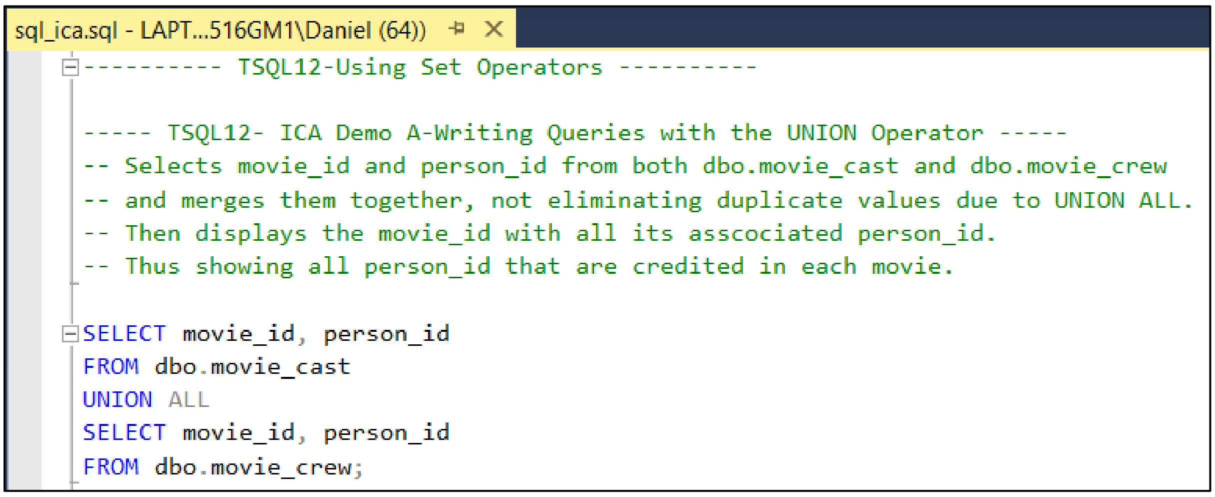
	title	runtime
64	??·???	120
65	??3	105
66	10 Cloverfiel...	103
67	10 Days in a ...	111
68	12 Years a Sl...	134
69	13 Hours: Th...	144
70	1982	90
71	2 Guns	109
72	20 Feet from ...	89
73	2016: Obam...	87
74	21 & Over	93
75	21 Jump Stre...	109
76	22 Jump Stre...	112

Query executed successfully.

TSQL12-Using Set Operators

TSQL12- ICA Demo A-Writing Queries with the UNION Operator

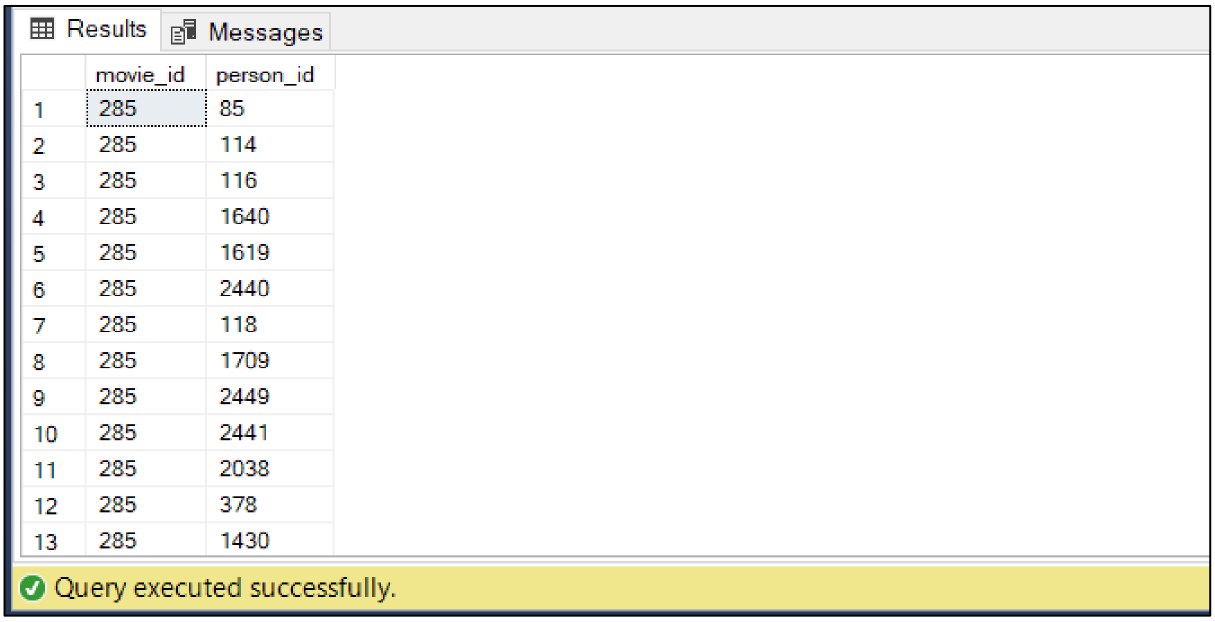
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```
---------- TSQL12-Using Set Operators ----------

----- TSQL12- ICA Demo A-Writing Queries with the UNION Operator -----
-- Selects movie_id and person_id from both dbo.movie_cast and dbo.movie_crew
-- and merges them together, not eliminating duplicate values due to UNION ALL.
-- Then displays the movie_id with all its asscociated person_id.
-- Thus showing all person_id that are credited in each movie.

SELECT movie_id, person_id
FROM dbo.movie_cast
UNION ALL
SELECT movie_id, person_id
FROM dbo.movie_crew;
```

Result:

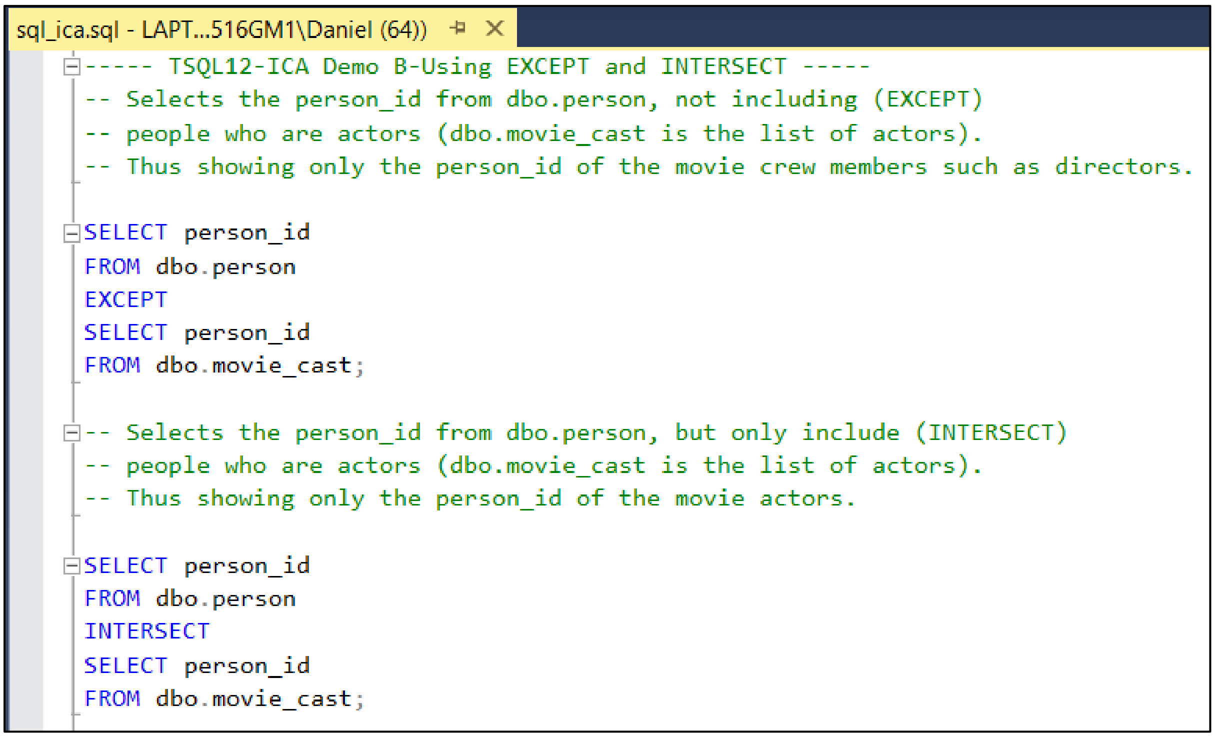


	movie_id	person_id
1	285	85
2	285	114
3	285	116
4	285	1640
5	285	1619
6	285	2440
7	285	118
8	285	1709
9	285	2449
10	285	2441
11	285	2038
12	285	378
13	285	1430

Query executed successfully.

TSQL12-ICA Demo B-Using EXCEPT and INTERSECT

Code:

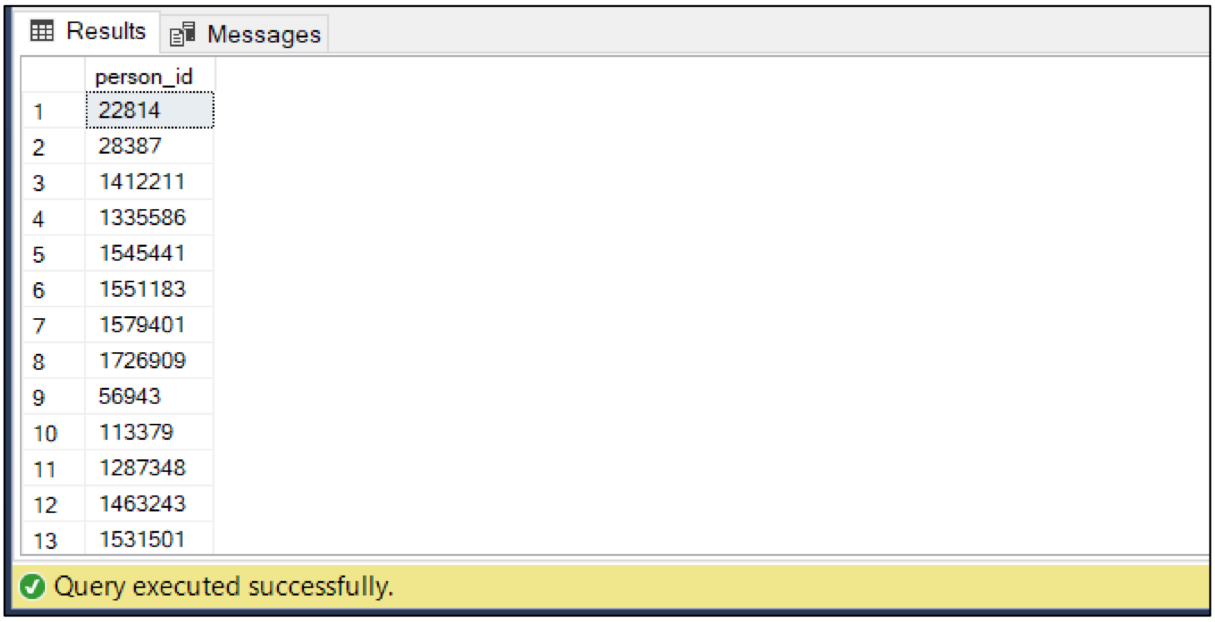


sql_ica.sql - LAPT...516GM1\Daniel (64))

```
----- TSQL12-ICA Demo B-Using EXCEPT and INTERSECT -----
-- Selects the person_id from dbo.person, not including (EXCEPT)
-- people who are actors (dbo.movie_cast is the list of actors).
-- Thus showing only the person_id of the movie crew members such as directors.

SELECT person_id
FROM dbo.person
EXCEPT
SELECT person_id
FROM dbo.movie_cast;

-- Selects the person_id from dbo.person, but only include (INTERSECT)
-- people who are actors (dbo.movie_cast is the list of actors).
-- Thus showing only the person_id of the movie actors.

SELECT person_id
FROM dbo.person
INTERSECT
SELECT person_id
FROM dbo.movie_cast;
```

Result:

[EXCEPT Result]

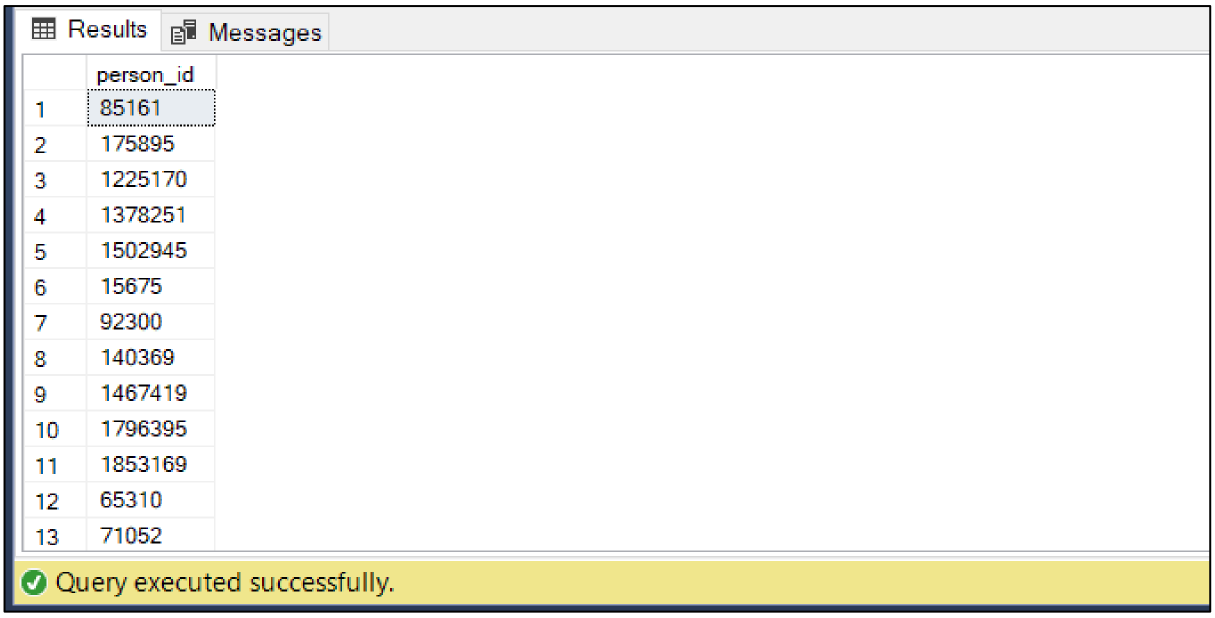


Results | Messages

	person_id
1	22814
2	28387
3	1412211
4	1335586
5	1545441
6	1551183
7	1579401
8	1726909
9	56943
10	113379
11	1287348
12	1463243
13	1531501

Query executed successfully.

[INTERSECT Result]

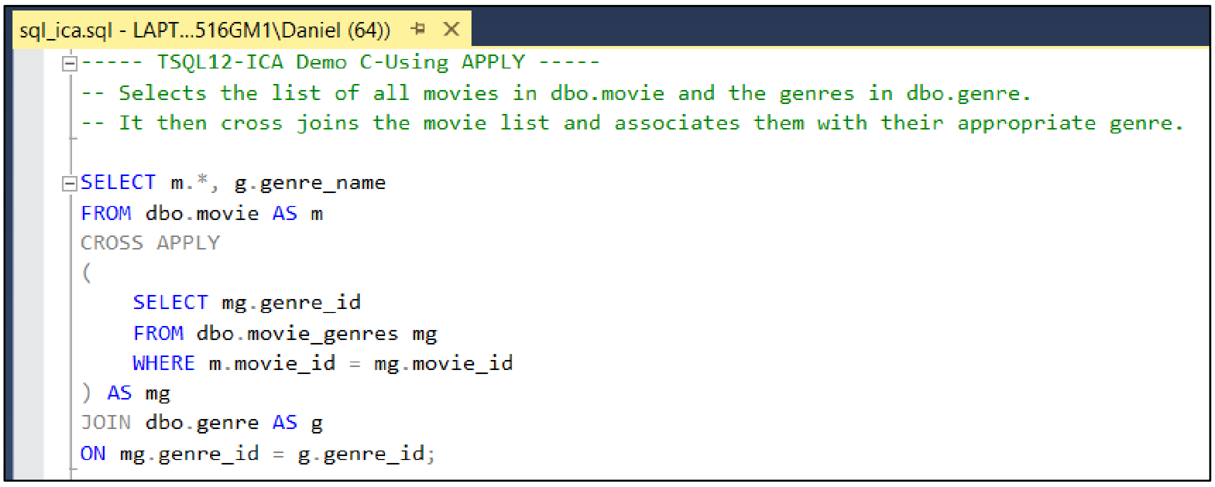

Results | Messages

	person_id
1	85161
2	175895
3	1225170
4	1378251
5	1502945
6	15675
7	92300
8	140369
9	1467419
10	1796395
11	1853169
12	65310
13	71052

Query executed successfully.

TSQL12-ICA Demo C-Using APPLY

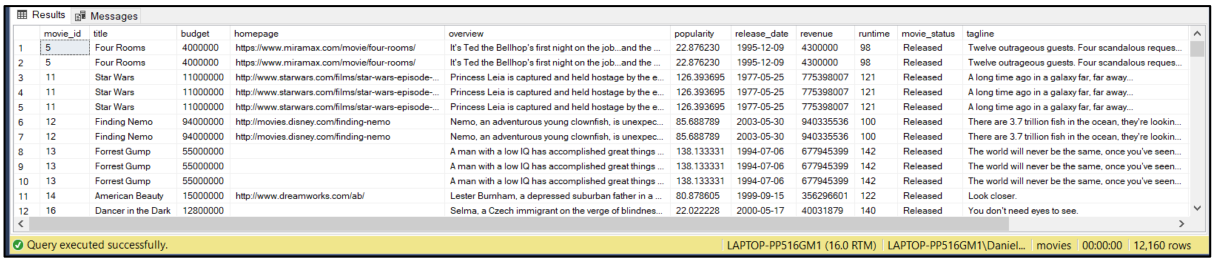
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
----- TSQL12-ICA Demo C-Using APPLY -----
-- Selects the list of all movies in dbo.movie and the genres in dbo.genre.
-- It then cross joins the movie list and associates them with their appropriate genre.

SELECT m.*, g.genre_name
FROM dbo.movie AS m
CROSS APPLY
(
    SELECT mg.genre_id
    FROM dbo.movie_genres mg
    WHERE m.movie_id = mg.movie_id
) AS mg
JOIN dbo.genre AS g
ON mg.genre_id = g.genre_id;
```

Result:



Results | Messages

	movie_id	title	budget	homepage	overview	popularity	release_date	revenue	runtime	movie_status	tagline
1	5	Four Rooms	4000000	https://www.miramax.com/movie/four-rooms/	It's Ted the Bellhop's first night on the job...and the ...	22.876230	1995-12-09	4300000	98	Released	Twelve outrageous guests. Four scandalous reques...
2	5	Four Rooms	4000000	https://www.miramax.com/movie/four-rooms/	It's Ted the Bellhop's first night on the job...and the ...	22.876230	1995-12-09	4300000	98	Released	Twelve outrageous guests. Four scandalous reques...
3	11	Star Wars	11000000	http://www.starwars.com/films/star-wars-episode-...	Princess Leia is captured and held hostage by the e...	126.393695	1977-05-25	775398007	121	Released	A long time ago in a galaxy far, far away...
4	11	Star Wars	11000000	http://www.starwars.com/films/star-wars-episode-...	Princess Leia is captured and held hostage by the e...	126.393695	1977-05-25	775398007	121	Released	A long time ago in a galaxy far, far away...
5	11	Star Wars	11000000	http://www.starwars.com/films/star-wars-episode-...	Princess Leia is captured and held hostage by the e...	126.393695	1977-05-25	775398007	121	Released	A long time ago in a galaxy far, far away...
6	12	Finding Nemo	94000000	http://movies.disney.com/finding-nemo	Nemo, an adventurous young clownfish, is unexpec...	85.688789	2003-05-30	940335536	100	Released	There are 3.7 trillion fish in the ocean, they're lookin...
7	12	Finding Nemo	94000000	http://movies.disney.com/finding-nemo	Nemo, an adventurous young clownfish, is unexpec...	85.688789	2003-05-30	940335536	100	Released	There are 3.7 trillion fish in the ocean, they're lookin...
8	13	Forrest Gump	55000000		A man with a low IQ has accomplished great things ...	138.133331	1994-07-06	677945399	142	Released	The world will never be the same, once you've seen...
9	13	Forrest Gump	55000000		A man with a low IQ has accomplished great things ...	138.133331	1994-07-06	677945399	142	Released	The world will never be the same, once you've seen...
10	13	Forrest Gump	55000000		A man with a low IQ has accomplished great things ...	138.133331	1994-07-06	677945399	142	Released	The world will never be the same, once you've seen...
11	14	American Beauty	15000000	http://www.dreamworks.com/ab/	Lester Burnham, a depressed suburban father in a ...	80.878605	1999-09-15	356296601	122	Released	Look closer.
12	16	Dancer in the Dark	12800000		Selma, a Czech immigrant on the verge of blindnes...	22.022228	2000-05-17	40031879	140	Released	You don't need eyes to see.

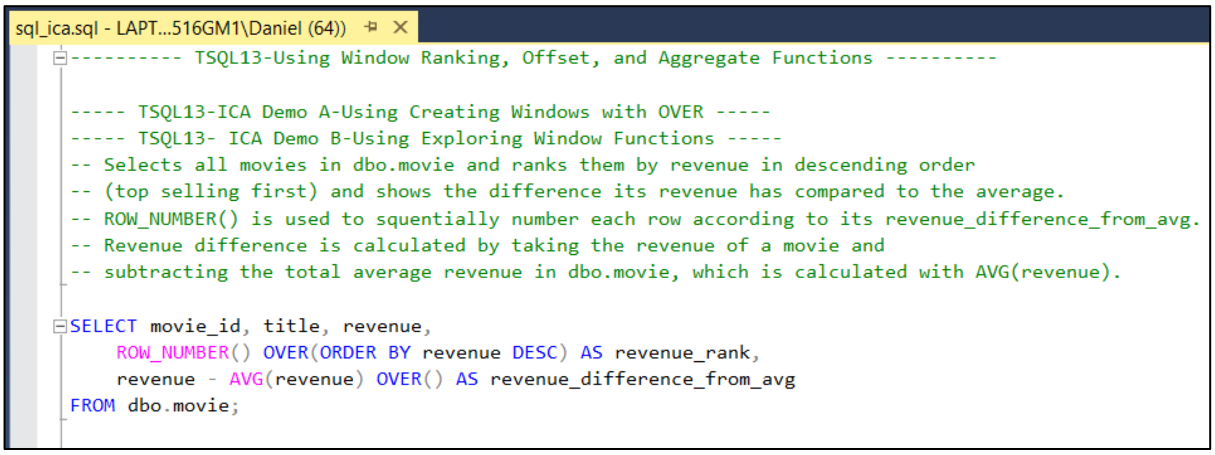
Query executed successfully. | LAPTOP-PP516GM1 (16.0 RTM) | LAPTOP-PP516GM1\Daniel... | movies | 00:00:00 | 12,160 rows

TSQL13-Using Window Ranking, Offset, and Aggregate Functions

TSQL13-ICA Demo A-Using Creating Windows with OVER
TSQL13- ICA Demo B-Using Exploring Window Functions

[TSQL13-ICA Demo A & TSQL13- ICA Demo B Both Done Together]

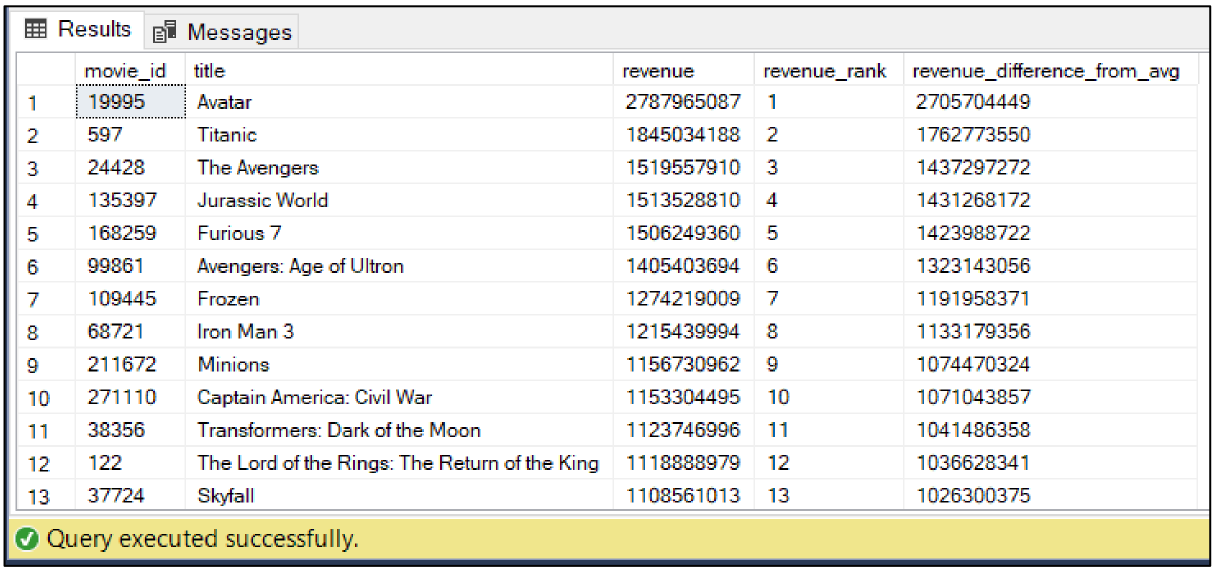
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
---------- TSQL13-Using Window Ranking, Offset, and Aggregate Functions ----------

----- TSQL13-ICA Demo A-Using Creating Windows with OVER -----
----- TSQL13- ICA Demo B-Using Exploring Window Functions -----
-- Selects all movies in dbo.movie and ranks them by revenue in descending order
-- (top selling first) and shows the difference its revenue has compared to the average.
-- ROW_NUMBER() is used to squentially number each row according to its revenue_difference_from_avg.
-- Revenue difference is calculated by taking the revenue of a movie and
-- subtracting the total average revenue in dbo.movie, which is calculated with AVG(revenue).

SELECT movie_id, title, revenue,
    ROW_NUMBER() OVER(ORDER BY revenue DESC) AS revenue_rank,
    revenue - AVG(revenue) OVER() AS revenue_difference_from_avg
FROM dbo.movie;
```

Result:



Results | Messages

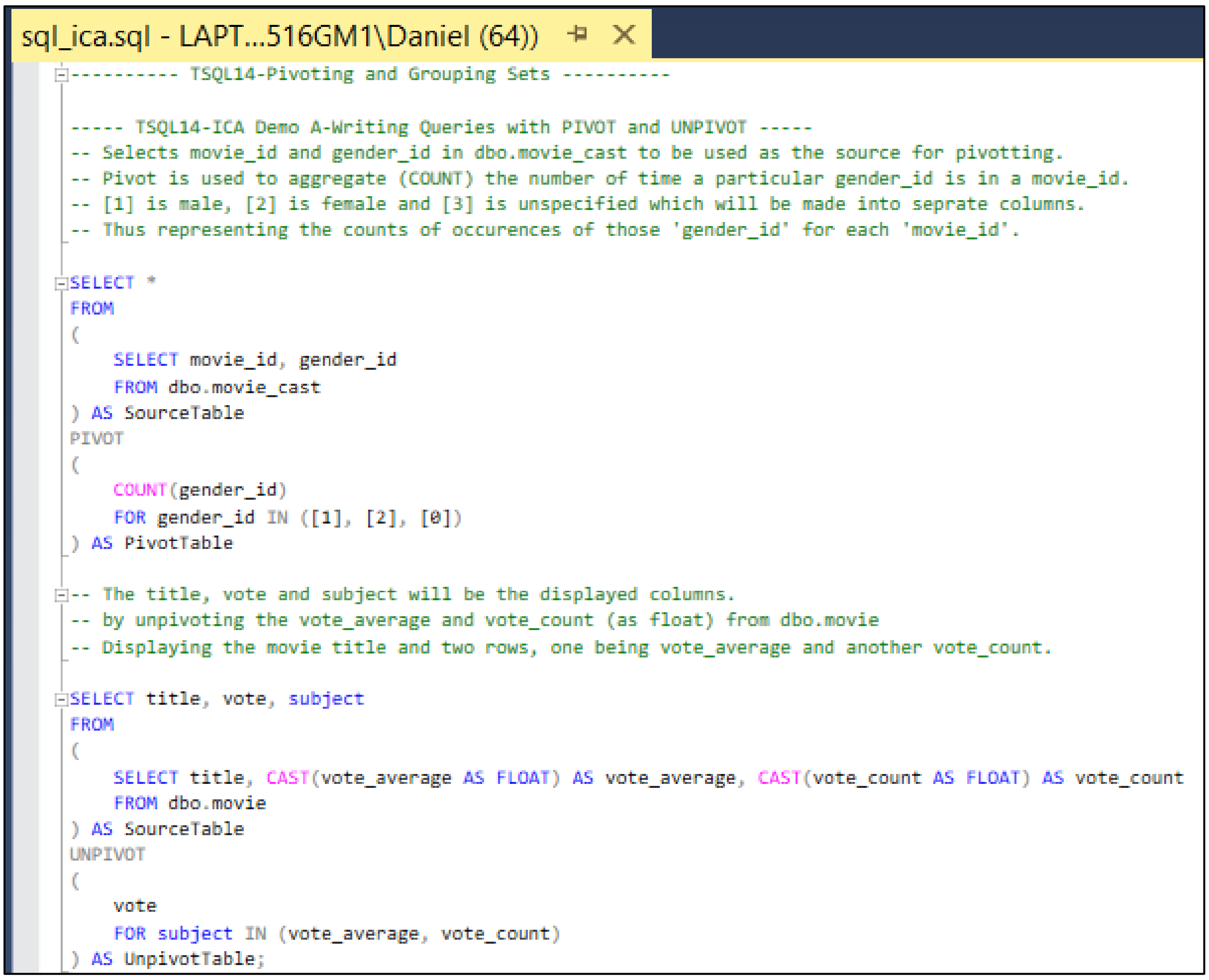
	movie_id	title	revenue	revenue_rank	revenue_difference_from_avg
1	19995	Avatar	2787965087	1	2705704449
2	597	Titanic	1845034188	2	1762773550
3	24428	The Avengers	1519557910	3	1437297272
4	135397	Jurassic World	1513528810	4	1431268172
5	168259	Furious 7	1506249360	5	1423988722
6	99861	Avengers: Age of Ultron	1405403694	6	1323143056
7	109445	Frozen	1274219009	7	1191958371
8	68721	Iron Man 3	1215439994	8	1133179356
9	211672	Minions	1156730962	9	1074470324
10	271110	Captain America: Civil War	1153304495	10	1071043857
11	38356	Transformers: Dark of the Moon	1123746996	11	1041486358
12	122	The Lord of the Rings: The Return of the King	1118888979	12	1036628341
13	37724	Skyfall	1108561013	13	1026300375

Query executed successfully.

TSQL14-Pivoting and Grouping Sets

TSQL14-ICA Demo A-Writing Queries with PIVOT and UNPIVOT

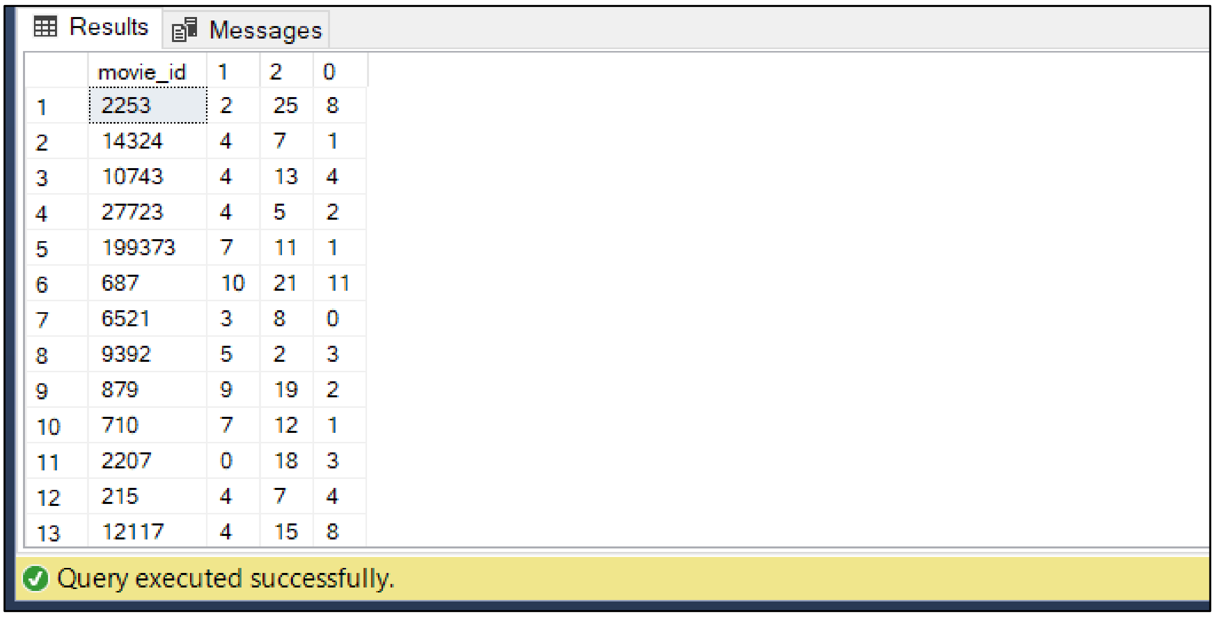
Code:



```
---------- TSQL14-Pivoting and Grouping Sets ----------

----- TSQL14-ICA Demo A-Writing Queries with PIVOT and UNPIVOT -----
-- Selects movie_id and gender_id in dbo.movie_cast to be used as the source for pivotting.
-- Pivot is used to aggregate (COUNT) the number of time a particular gender_id is in a movie_id.
-- [1] is male, [2] is female and [3] is unspecified which will be made into seprate columns.
-- Thus representing the counts of occurences of those 'gender_id' for each 'movie_id'.

SELECT *
FROM
(
    SELECT movie_id, gender_id
    FROM dbo.movie_cast
) AS SourceTable
PIVOT
(
    COUNT(gender_id)
    FOR gender_id IN ([1], [2], [0])
) AS PivotTable

-- The title, vote and subject will be the displayed columns.
-- by unpivoting the vote_average and vote_count (as float) from dbo.movie
-- Displaying the movie title and two rows, one being vote_average and another vote_count.

SELECT title, vote, subject
FROM
(
    SELECT title, CAST(vote_average AS FLOAT) AS vote_average, CAST(vote_count AS FLOAT) AS vote_count
    FROM dbo.movie
) AS SourceTable
UNPIVOT
(
    vote
    FOR subject IN (vote_average, vote_count)
) AS UnpivotTable;
```

Result:

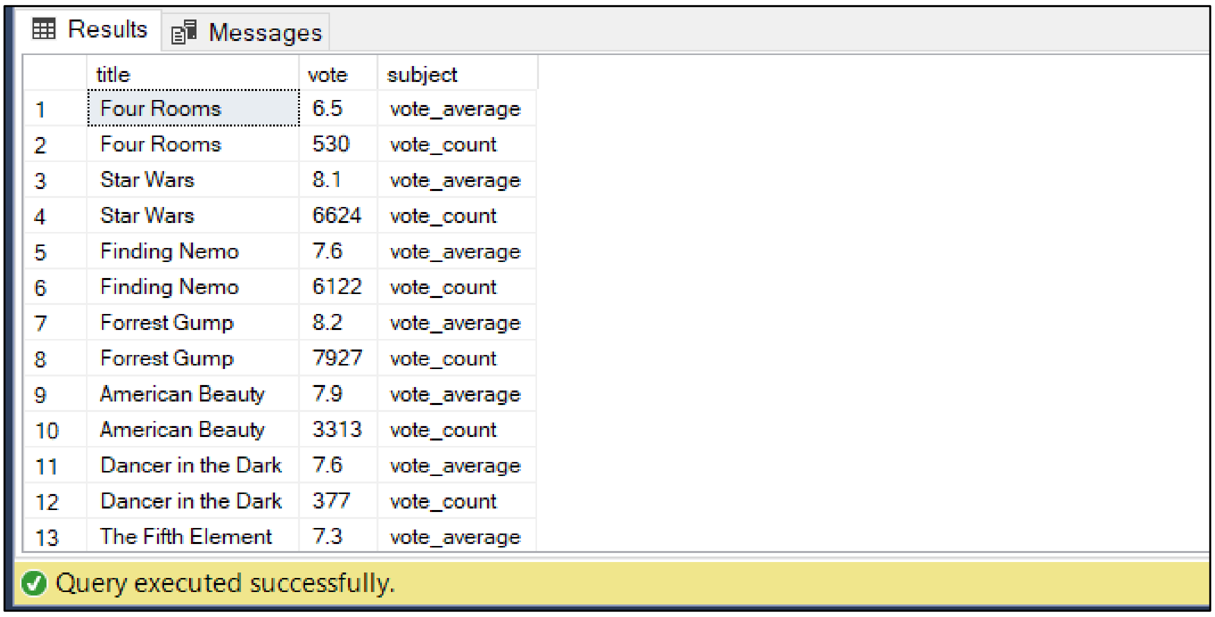
[PIVOT Result]


Results | Messages

	movie_id	1	2	0
1	2253	2	25	8
2	14324	4	7	1
3	10743	4	13	4
4	27723	4	5	2
5	199373	7	11	1
6	687	10	21	11
7	6521	3	8	0
8	9392	5	2	3
9	879	9	19	2
10	710	7	12	1
11	2207	0	18	3
12	215	4	7	4
13	12117	4	15	8

Query executed successfully.

[UNPIVOT Result]

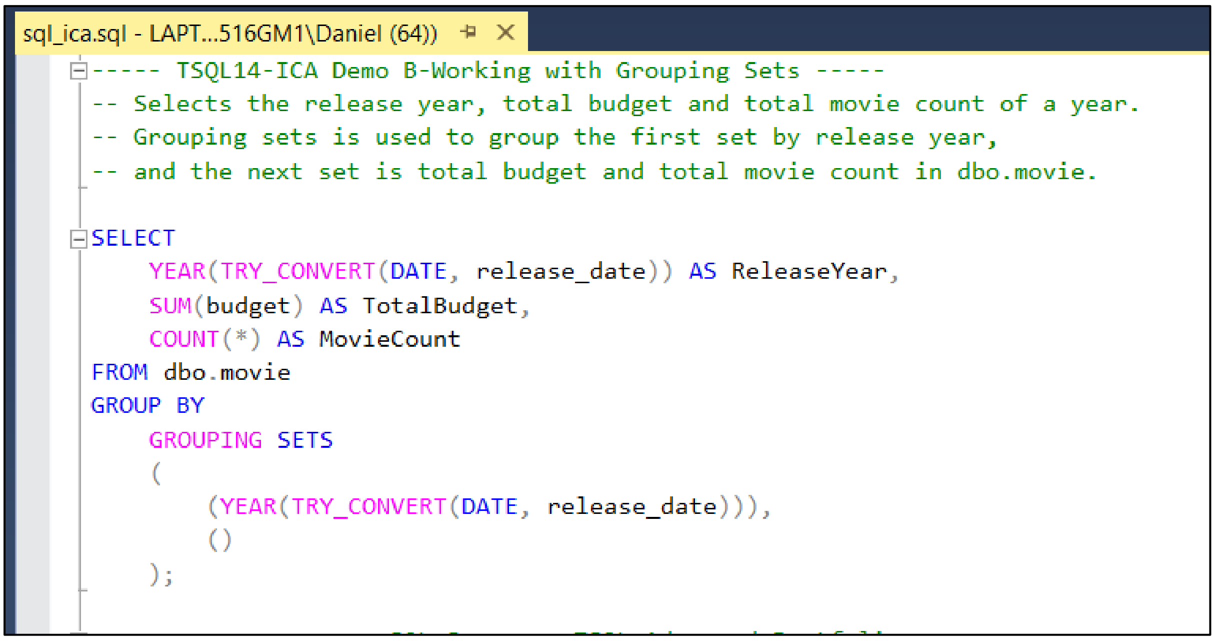

Results | Messages

	title	vote	subject
1	Four Rooms	6.5	vote_average
2	Four Rooms	530	vote_count
3	Star Wars	8.1	vote_average
4	Star Wars	6624	vote_count
5	Finding Nemo	7.6	vote_average
6	Finding Nemo	6122	vote_count
7	Forrest Gump	8.2	vote_average
8	Forrest Gump	7927	vote_count
9	American Beauty	7.9	vote_average
10	American Beauty	3313	vote_count
11	Dancer in the Dark	7.6	vote_average
12	Dancer in the Dark	377	vote_count
13	The Fifth Element	7.3	vote_average

Query executed successfully.

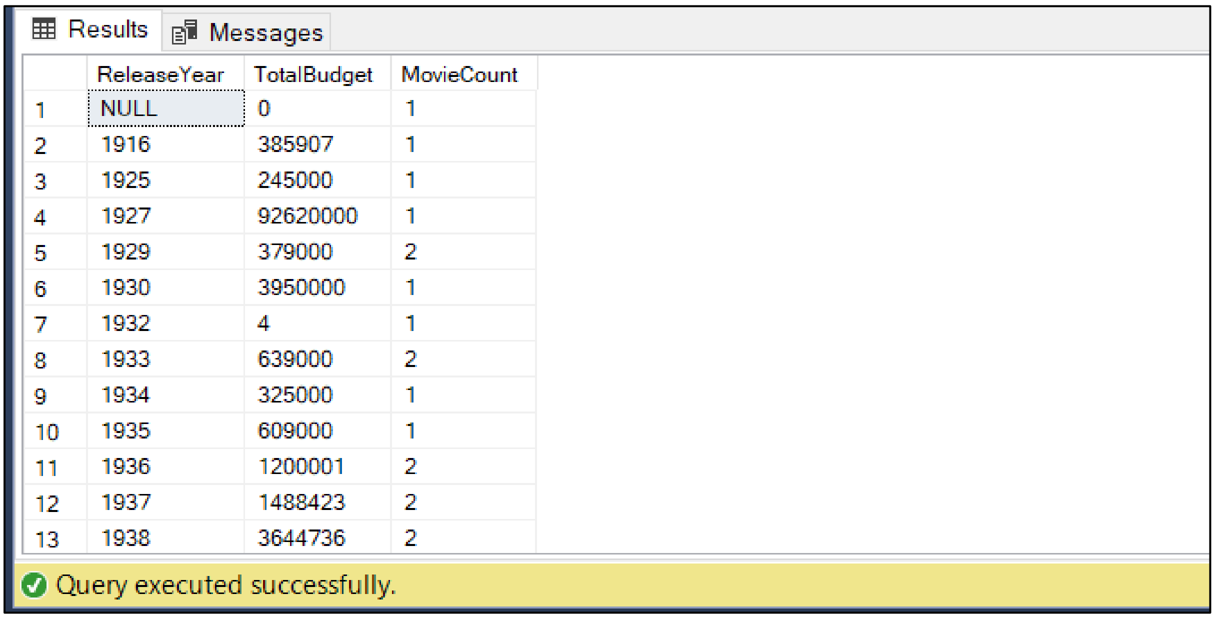
TSQL14-ICA Demo B-Working with Grouping Sets

Code:


sql_ica.sql - LAPT...516GM1\Daniel (64))

```
----- TSQL14-ICA Demo B-Working with Grouping Sets -----
-- Selects the release year, total budget and total movie count of a year.
-- Grouping sets is used to group the first set by release year,
-- and the next set is total budget and total movie count in dbo.movie.

SELECT
    YEAR(TRY_CONVERT(DATE, release_date)) AS ReleaseYear,
    SUM(budget) AS TotalBudget,
    COUNT(*) AS MovieCount
FROM dbo.movie
GROUP BY
    GROUPING SETS
    (
        (YEAR(TRY_CONVERT(DATE, release_date))),
        ()
    );
```

Result:



	ReleaseYear	TotalBudget	MovieCount
1	NULL	0	1
2	1916	385907	1
3	1925	245000	1
4	1927	92620000	1
5	1929	379000	2
6	1930	3950000	1
7	1932	4	1
8	1933	639000	2
9	1934	325000	1
10	1935	609000	1
11	1936	1200001	2
12	1937	1488423	2
13	1938	3644736	2

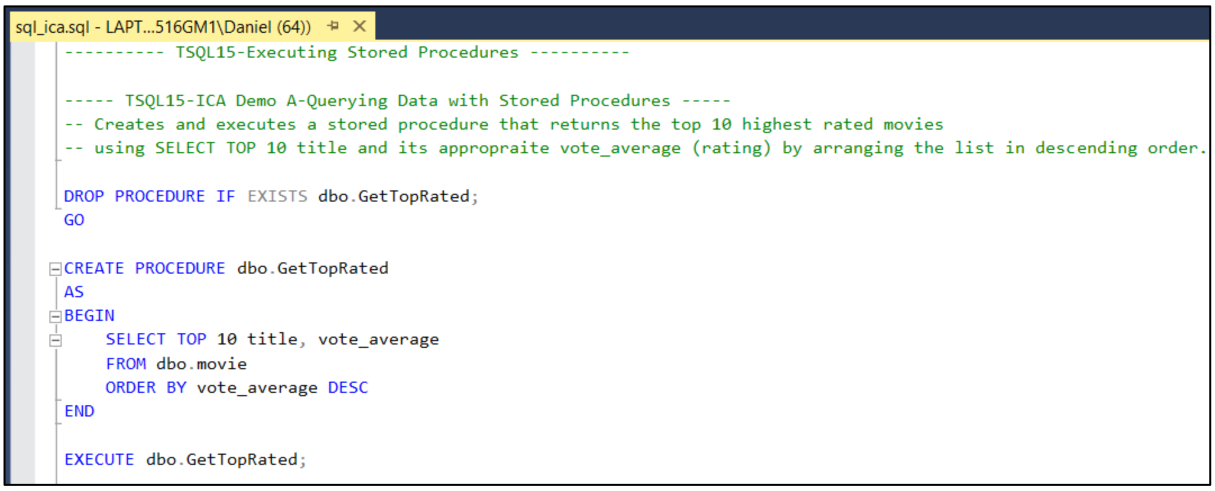
Query executed successfully.

SQL Server - TSQL Advanced Portfolio:

TSQL15-Executing Stored Procedures

TSQL15-ICA Demo A-Querying Data with Stored Procedures

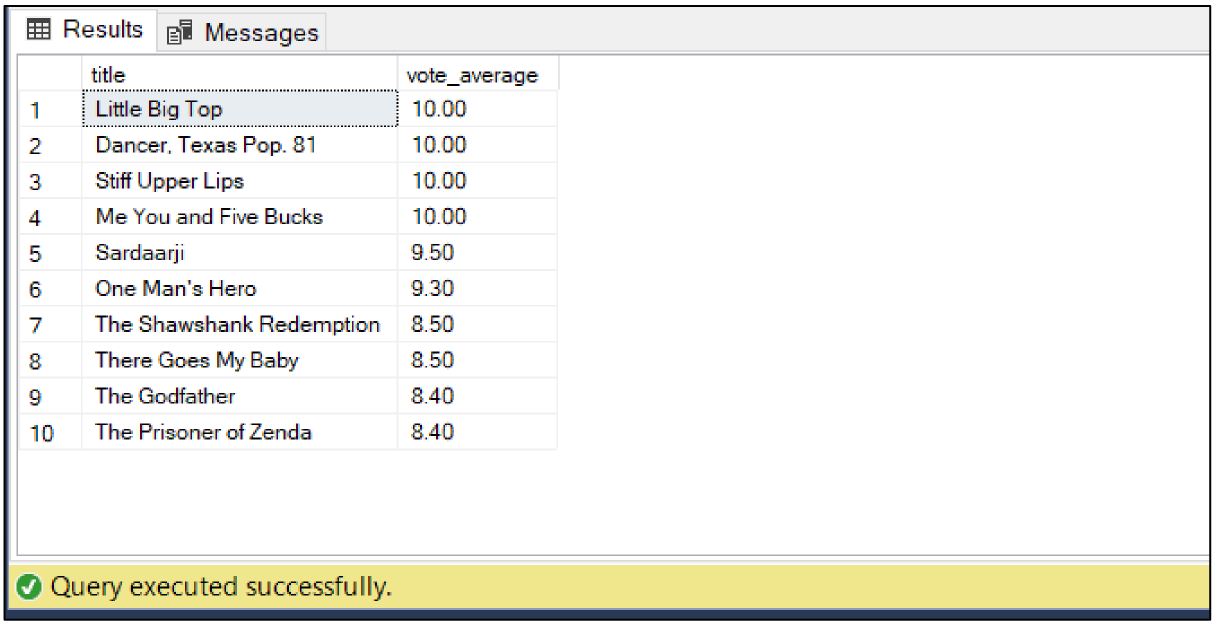
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
---------- TSQL15-Executing Stored Procedures ----------

----- TSQL15-ICA Demo A-Querying Data with Stored Procedures -----
-- Creates and executes a stored procedure that returns the top 10 highest rated movies
-- using SELECT TOP 10 title and its appropraite vote_average (rating) by arranging the list in descending order.

DROP PROCEDURE IF EXISTS dbo.GetTopRated;
GO

CREATE PROCEDURE dbo.GetTopRated
AS
BEGIN
    SELECT TOP 10 title, vote_average
    FROM dbo.movie
    ORDER BY vote_average DESC
END

EXECUTE dbo.GetTopRated;
```

Result:



	title	vote_average
1	Little Big Top	10.00
2	Dancer, Texas Pop. 81	10.00
3	Stiff Upper Lips	10.00
4	Me You and Five Bucks	10.00
5	Sardaarji	9.50
6	One Man's Hero	9.30
7	The Shawshank Redemption	8.50
8	There Goes My Baby	8.50
9	The Godfather	8.40
10	The Prisoner of Zenda	8.40

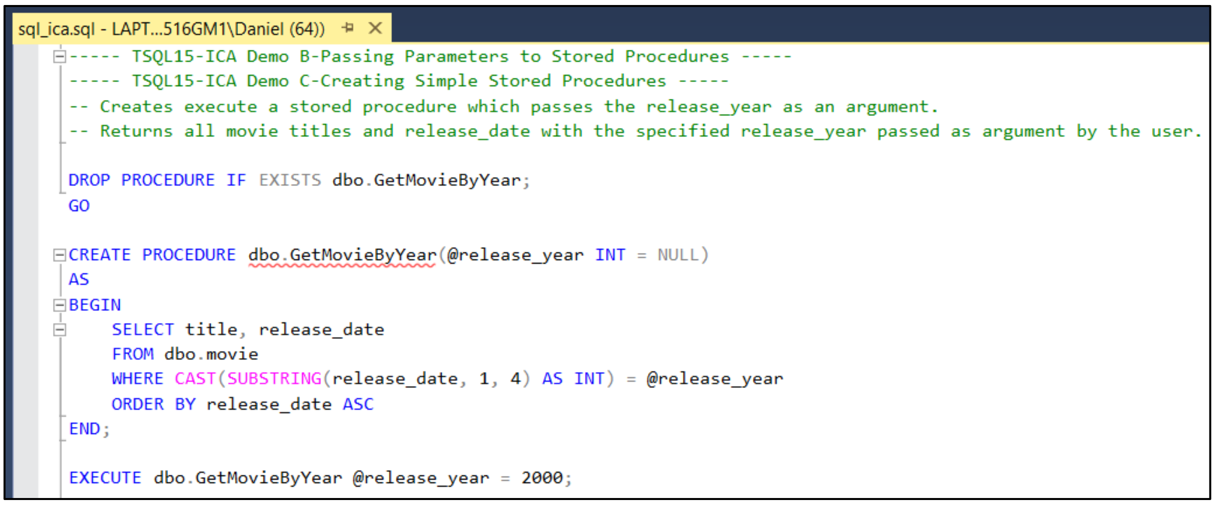
Query executed successfully.

TSQL15-ICA Demo B-Passing Parameters to Stored Procedures

TSQL15-ICA Demo C-Creating Simple Stored Procedures

[TSQL15-ICA Demo B & TSQL15-ICA Demo C Both Done Together]

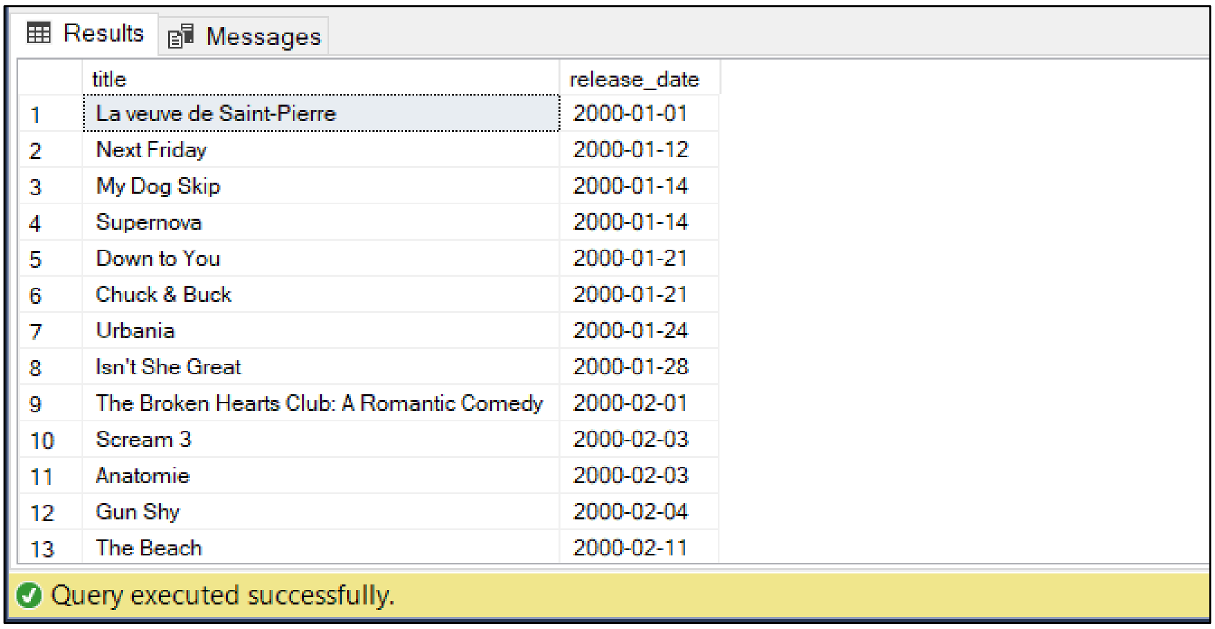
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```
----- TSQL15-ICA Demo B-Passing Parameters to Stored Procedures -----
----- TSQL15-ICA Demo C-Creating Simple Stored Procedures -----
-- Creates execute a stored procedure which passes the release_year as an argument.
-- Returns all movie titles and release_date with the specified release_year passed as argument by the user.

DROP PROCEDURE IF EXISTS dbo.GetMovieByYear;
GO

CREATE PROCEDURE dbo.GetMovieByYear(@release_year INT = NULL)
AS
BEGIN
    SELECT title, release_date
    FROM dbo.movie
    WHERE CAST(SUBSTRING(release_date, 1, 4) AS INT) = @release_year
    ORDER BY release_date ASC
END;

EXECUTE dbo.GetMovieByYear @release_year = 2000;
```

Result:



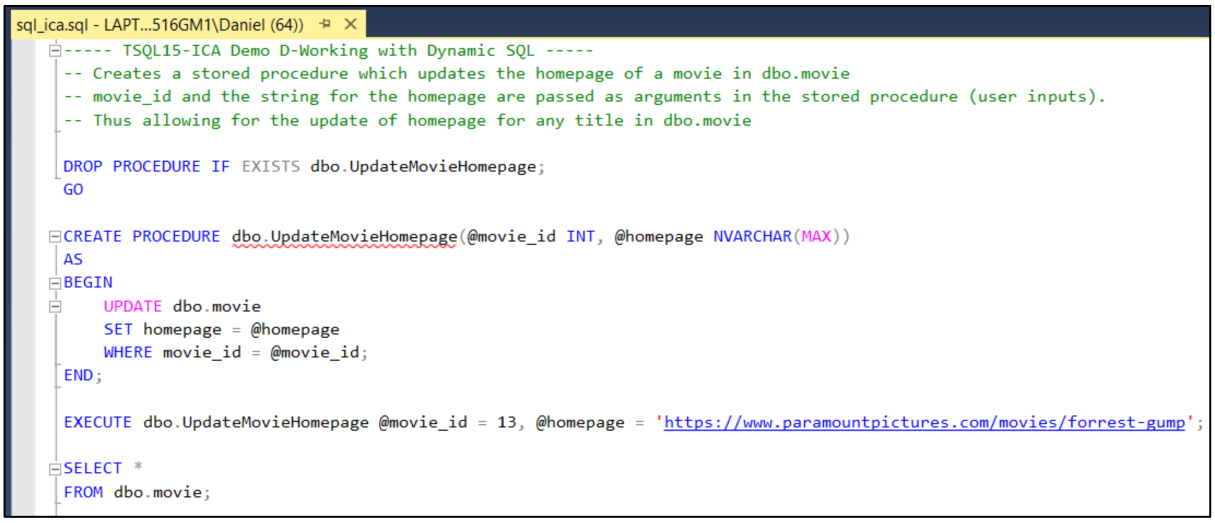
Results | Messages

	title	release_date
1	La veuve de Saint-Pierre	2000-01-01
2	Next Friday	2000-01-12
3	My Dog Skip	2000-01-14
4	Supernova	2000-01-14
5	Down to You	2000-01-21
6	Chuck & Buck	2000-01-21
7	Urbania	2000-01-24
8	Isn't She Great	2000-01-28
9	The Broken Hearts Club: A Romantic Comedy	2000-02-01
10	Scream 3	2000-02-03
11	Anatomie	2000-02-03
12	Gun Shy	2000-02-04
13	The Beach	2000-02-11

Query executed successfully.

TSQL15-ICA Demo D-Working with Dynamic SQL

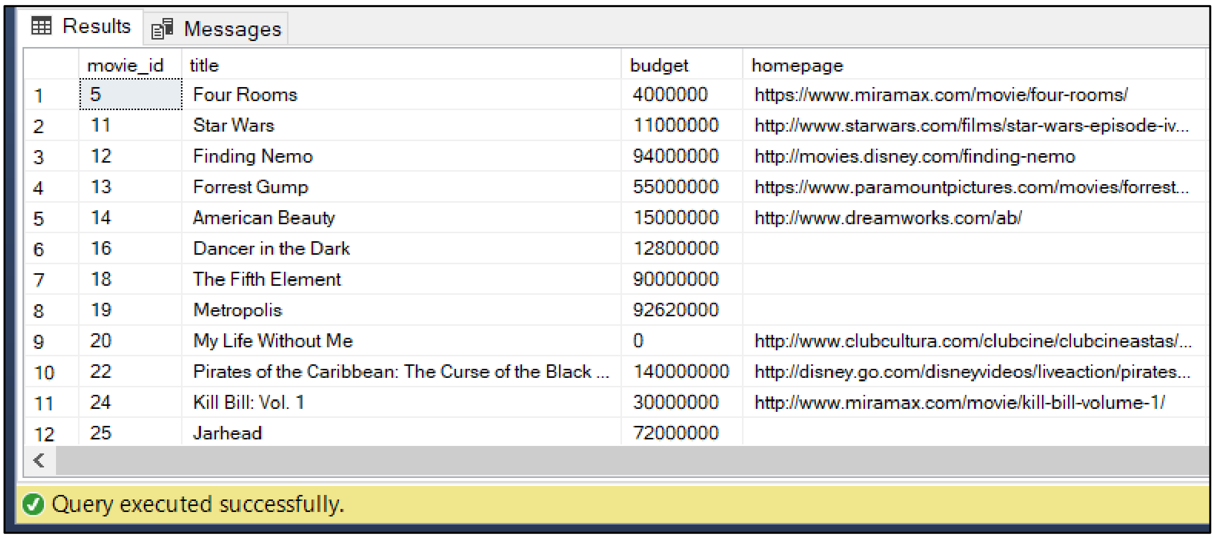
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
----- TSQL15-ICA Demo D-Working with Dynamic SQL -----
-- Creates a stored procedure which updates the homepage of a movie in dbo.movie
-- movie_id and the string for the homepage are passed as arguments in the stored procedure (user inputs).
-- Thus allowing for the update of homepage for any title in dbo.movie

DROP PROCEDURE IF EXISTS dbo.UpdateMovieHomepage;
GO

CREATE PROCEDURE dbo.UpdateMovieHomepage(@movie_id INT, @homepage NVARCHAR(MAX))
AS
BEGIN
    UPDATE dbo.movie
    SET homepage = @homepage
    WHERE movie_id = @movie_id;
END;

EXECUTE dbo.UpdateMovieHomepage @movie_id = 13, @homepage = 'https://www.paramountpictures.com/movies/forrest-gump';

SELECT *
FROM dbo.movie;
```

Result:



	movie_id	title	budget	homepage
1	5	Four Rooms	4000000	https://www.miramax.com/movie/four-rooms/
2	11	Star Wars	11000000	http://www.starwars.com/films/star-wars-episode-iv...
3	12	Finding Nemo	94000000	http://movies.disney.com/finding-nemo
4	13	Forrest Gump	55000000	https://www.paramountpictures.com/movies/forrest...
5	14	American Beauty	15000000	http://www.dreamworks.com/ab/
6	16	Dancer in the Dark	12800000	
7	18	The Fifth Element	90000000	
8	19	Metropolis	92620000	
9	20	My Life Without Me	0	http://www.clubcultura.com/clubcine/clubcineastas/...
10	22	Pirates of the Caribbean: The Curse of the Black ...	140000000	http://disney.go.com/disneyvideos/liveaction/pirates...
11	24	Kill Bill: Vol. 1	30000000	http://www.miramax.com/movie/kill-bill-volume-1/
12	25	Jarhead	72000000	

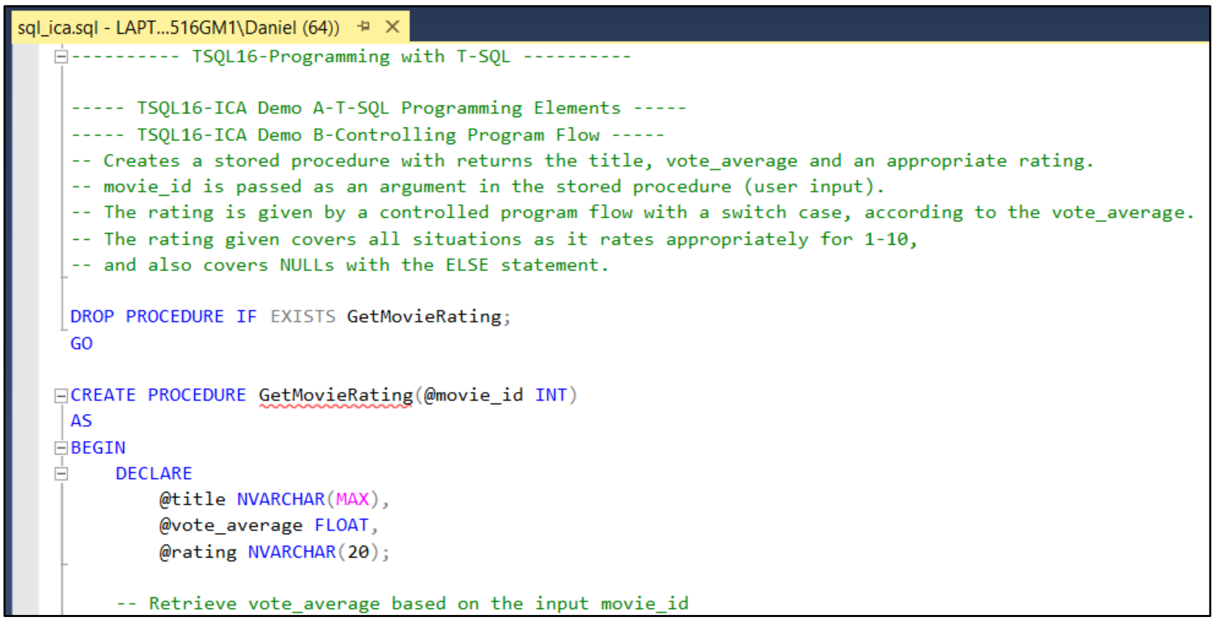
Query executed successfully.

TSQL16-Programming with T-SQL

TSQL16-ICA Demo A-T-SQL Programming Elements
TSQL16-ICA Demo B-Controlling Program Flow

[TSQL16-ICA Demo A & TSQL16-ICA Demo B Both Done Together]

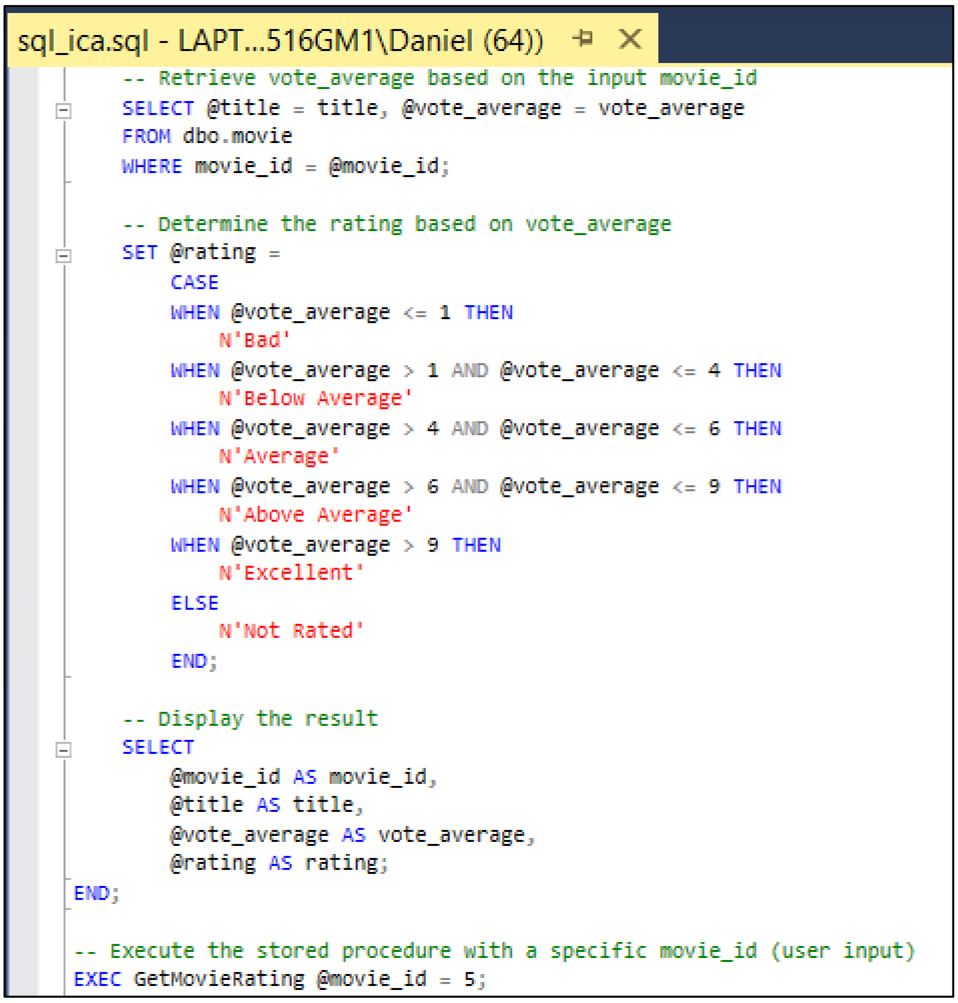
Code:
[Part 1/2]



sql_ica.sql - LAPT...516GM1\Daniel (64))

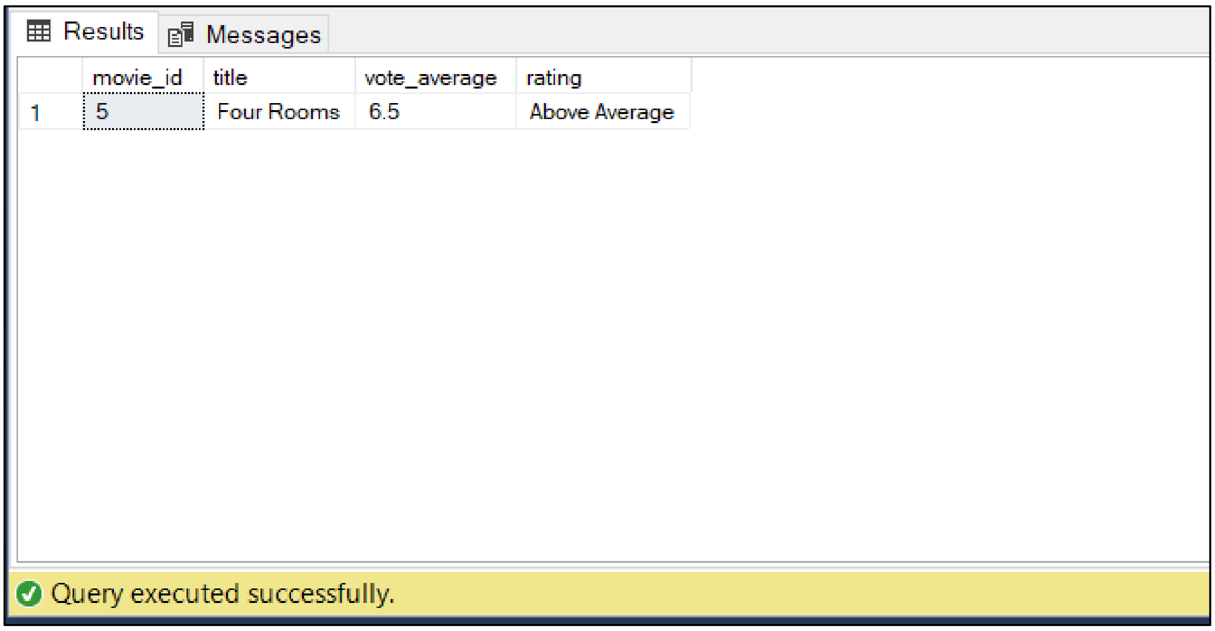
```sql
---------- TSQL16-Programming with T-SQL ----------

----- TSQL16-ICA Demo A-T-SQL Programming Elements -----
----- TSQL16-ICA Demo B-Controlling Program Flow -----
-- Creates a stored procedure with returns the title, vote_average and an appropriate rating.
-- movie_id is passed as an argument in the stored procedure (user input).
-- The rating is given by a controlled program flow with a switch case, according to the vote_average.
-- The rating given covers all situations as it rates appropriately for 1-10,
-- and also covers NULLs with the ELSE statement.

DROP PROCEDURE IF EXISTS GetMovieRating;
GO

CREATE PROCEDURE GetMovieRating(@movie_id INT)
AS
BEGIN
    DECLARE
        @title NVARCHAR(MAX),
        @vote_average FLOAT,
        @rating NVARCHAR(20);

    -- Retrieve vote_average based on the input movie_id
```

[Part 2/2]


sql_ica.sql - LAPT...516GM1\Daniel (64))

```
    -- Retrieve vote_average based on the input movie_id
    SELECT @title = title, @vote_average = vote_average
    FROM dbo.movie
    WHERE movie_id = @movie_id;

    -- Determine the rating based on vote_average
    SET @rating =
        CASE
        WHEN @vote_average <= 1 THEN
            N'Bad'
        WHEN @vote_average > 1 AND @vote_average <= 4 THEN
            N'Below Average'
        WHEN @vote_average > 4 AND @vote_average <= 6 THEN
            N'Average'
        WHEN @vote_average > 6 AND @vote_average <= 9 THEN
            N'Above Average'
        WHEN @vote_average > 9 THEN
            N'Excellent'
        ELSE
            N'Not Rated'
        END;

    -- Display the result
    SELECT
        @movie_id AS movie_id,
        @title AS title,
        @vote_average AS vote_average,
        @rating AS rating;
END;

-- Execute the stored procedure with a specific movie_id (user input)
EXEC GetMovieRating @movie_id = 5;
```

Result:



	movie_id	title	vote_average	rating
1	5	Four Rooms	6.5	Above Average

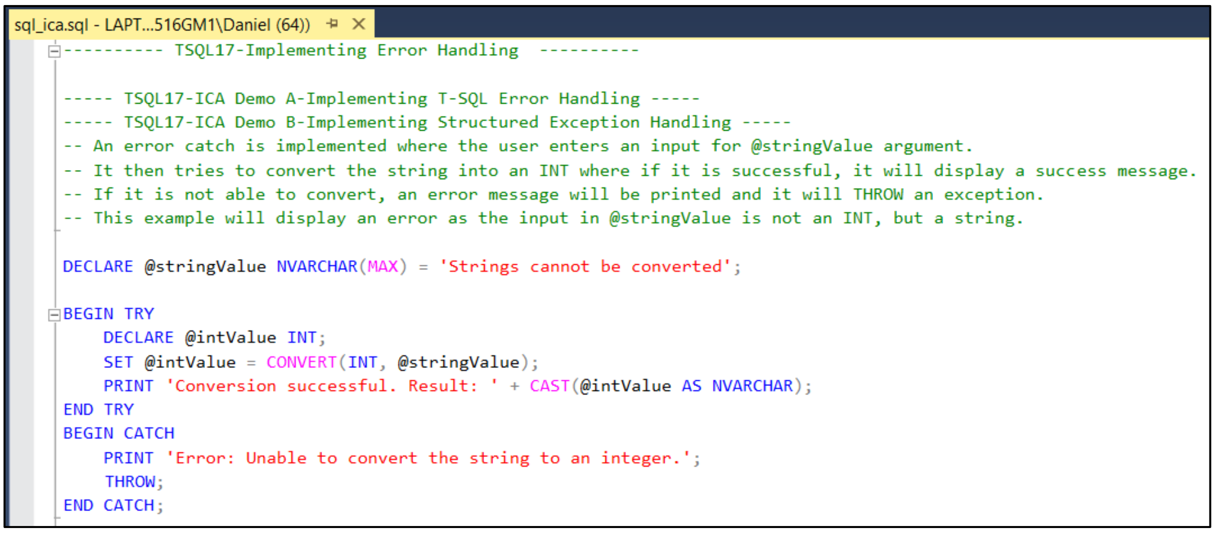
Query executed successfully.

TSQL 7-Implementing Error Handling

TSQL17-ICA Demo A-Implementing T-SQL Error Handling
TSQL17-ICA Demo B-Implementing Structured Exception Handling

[TSQL17-ICA Demo A & TSQL17-ICA Demo B Both Done Together]

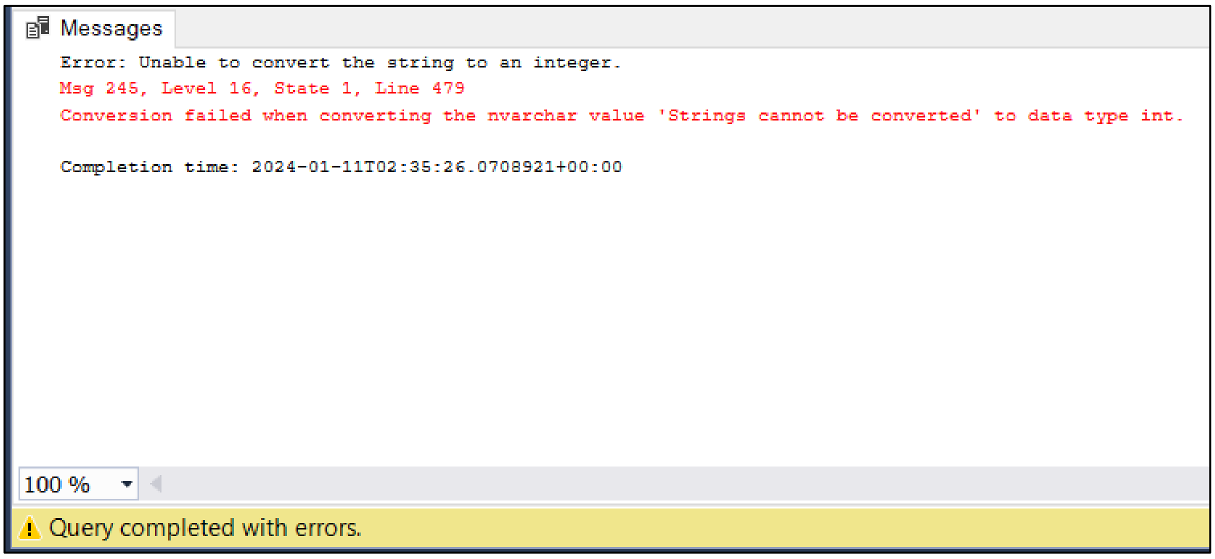
Code:



sql_ica.sql - LAPT...516GM1\Daniel (64))

```sql
---------- TSQL17-Implementing Error Handling  ----------

----- TSQL17-ICA Demo A-Implementing T-SQL Error Handling -----
----- TSQL17-ICA Demo B-Implementing Structured Exception Handling -----
-- An error catch is implemented where the user enters an input for @stringValue argument.
-- It then tries to convert the string into an INT where if it is successful, it will display a success message.
-- If it is not able to convert, an error message will be printed and it will THROW an exception.
-- This example will display an error as the input in @stringValue is not an INT, but a string.

DECLARE @stringValue NVARCHAR(MAX) = 'Strings cannot be converted';

BEGIN TRY
    DECLARE @intValue INT;
    SET @intValue = CONVERT(INT, @stringValue);
    PRINT 'Conversion successful. Result: ' + CAST(@intValue AS NVARCHAR);
END TRY
BEGIN CATCH
    PRINT 'Error: Unable to convert the string to an integer.';
    THROW;
END CATCH;
```

Result:



Messages

```
Error: Unable to convert the string to an integer.
Msg 245, Level 16, State 1, Line 479
Conversion failed when converting the nvarchar value 'Strings cannot be converted' to data type int.

Completion time: 2024-01-11T02:35:26.0708921+00:00
```

100 %

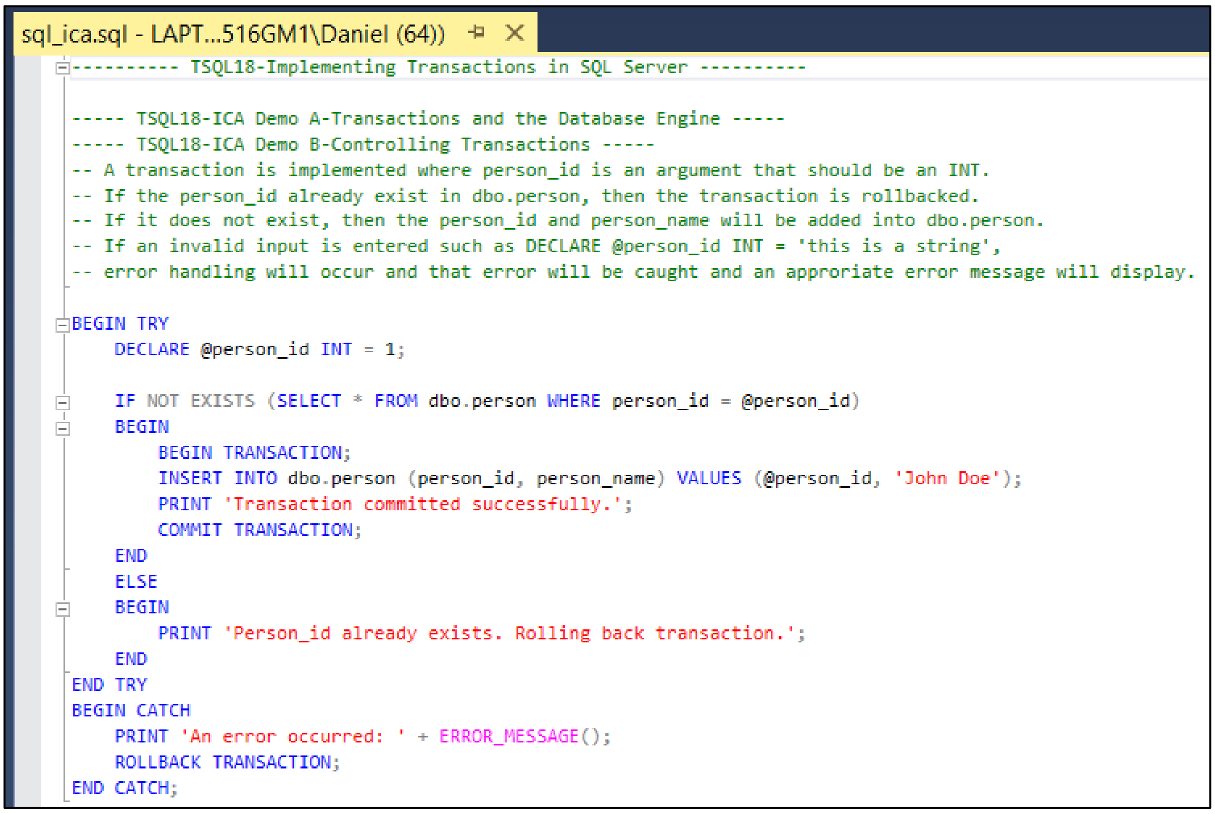
Query completed with errors.

TSQL18-Implementing Transactions in SQL Server

TSQL18-ICA Demo A-Transactions and the Database Engine
TSQL18-ICA Demo B-Controlling Transactions

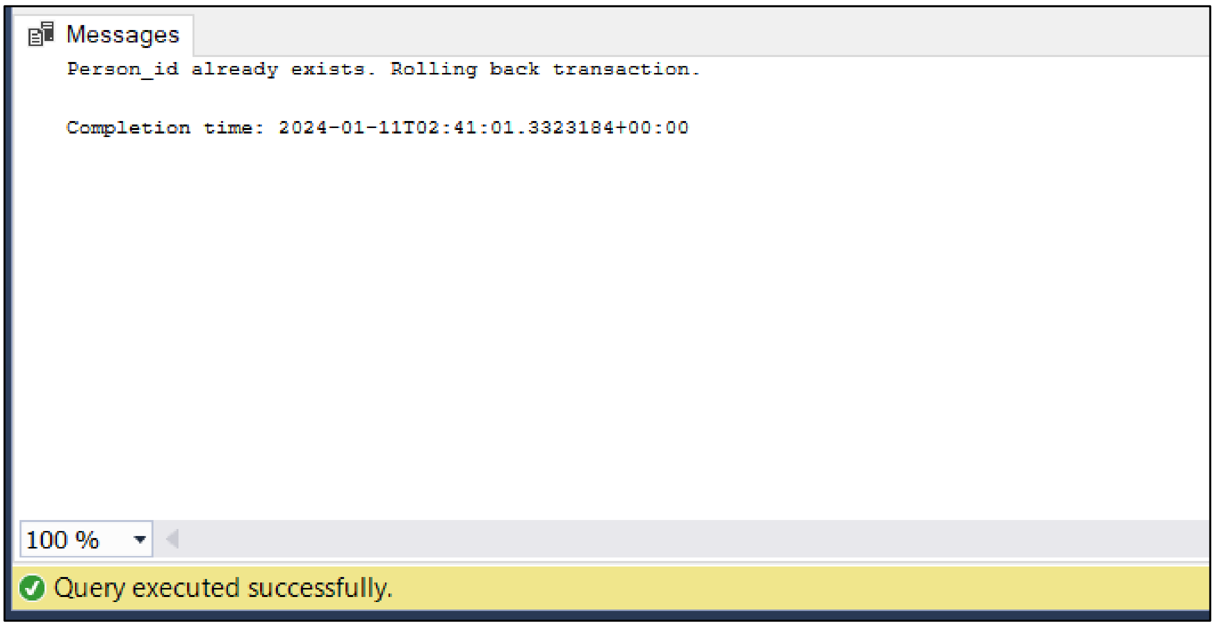
[TSQL18-ICA Demo A & TSQL18-ICA Demo B Both Done Together]

Code:



```
---------- TSQL18-Implementing Transactions in SQL Server ----------

----- TSQL18-ICA Demo A-Transactions and the Database Engine -----
----- TSQL18-ICA Demo B-Controlling Transactions -----
-- A transaction is implemented where person_id is an argument that should be an INT.
-- If the person_id already exist in dbo.person, then the transaction is rollbacked.
-- If it does not exist, then the person_id and person_name will be added into dbo.person.
-- If an invalid input is entered such as DECLARE @person_id INT = 'this is a string',
-- error handling will occur and that error will be caught and an appropriate error message will display.

BEGIN TRY
    DECLARE @person_id INT = 1;

    IF NOT EXISTS (SELECT * FROM dbo.person WHERE person_id = @person_id)
    BEGIN
        BEGIN TRANSACTION;
        INSERT INTO dbo.person (person_id, person_name) VALUES (@person_id, 'John Doe');
        PRINT 'Transaction committed successfully.';
        COMMIT TRANSACTION;
    END
    ELSE
    BEGIN
        PRINT 'Person_id already exists. Rolling back transaction.';
    END
END TRY
BEGIN CATCH
    PRINT 'An error occurred: ' + ERROR_MESSAGE();
    ROLLBACK TRANSACTION;
END CATCH;
```

Result:



```
Person_id already exists. Rolling back transaction.

Completion time: 2024-01-11T02:41:01.3323184+00:00
```