Data Engineering Project
End-to-End Data Modelling ETL and Visualization
Phase-1

Deliverable 1.1. **Dataset**

IMDB Movie Dataset contains about 5000 movies with around 28 attributes, including the director, actors, and plot of the movie. The dataset has scope of cleaning data, 1NF, 2NF, 3NF, NULLs, creating ETL scripts and good visualisation.

Cleaning of Dataset:

Dataset has been studied and it require cleaning before imported to SQL Server. Following cleaning activities will be performed on data using Tableau Prep as first step:

- **Handling null values**
  - **Removed rows with missing title_year, actor_1_name, or duration:**
    Missing values in three important columns title_year, actor_1_name, or duration, need cleaning. Although there are many mechanisms to handle missing values, however, keeping the normalization requirement these records are been removed.
  - **Filling missing country to USA:**
    There is only one record with missing country after removing missing data rows. After searching over internet, it is verified that country for this movie was USA, so, one record modified accordingly.
- **Remove all non-ASCII characters**
  It is found that few non-ASCII characters been added at end of each movie title, that in not making any value addition, removing all non-ASCII characters from dataset.
- **Cleaning same movie rows with minor differences**
  It is found that there exist almost similar records. Such records have almost same value for categorical variables and very minor difference in numerical fields like facebook likes etc. Such records are considered as dirty, and have been removed from dataset. Removing these records will not loss any information.
- **Cleaning specific record for movie 'Ben-Hur'**
  There are 3 records for same movie 'Ben-Hur', where one record doesn't contain plot_keywords, and others differ in only few like numbers. Removed record with missing plot_keywords.
- **Cleaning specific record for movie 'Brothers'**
  There are 2 records for same movie 'Brothers', where records contain different actor_3_name and actor_3_facebook_likes. It is assumed that actor with higher actor_3_facebook_likes as 3rd main actor for movie, which is 'Bailee Madison'.

Important points regarding Final Dataset are:

- 249 records out of 5043 are removed from dataset. There were less than 127 records, that are removed due to cleaning issue, which is approximately 2.5% of total rows. Rest records are duplicate.
- Modified dataset contains 4794 records and 28 attributes.
- 14 columns out of 28 contains null values, however, missing values are less and not been ignored.
- There are cases where duplicate movie name exists, this is because same movie name being used in different release year.

Why Data is in Zero-NF?

For a dataset to be in 1NF, each and every cell in dataset should have atomic identifiable value. However, in the IMDB dataset two columns' genres and plot_keywords contain multiple values separated with '|' (pipe). For this reason, dataset is in Zero-NF.

Dataset in Zero-NF:

All the columns in datasets are given a alphanumeric code for normalisation process and are listed below:

(A1) movie_title
(A2) title_year
(A3) genres
(B) color(*)
(C) aspect_ratio(*)
(D) duration
(E) language(*)
(F) country
(G) plot_keywords(*)
(H) budget(*)
(I) gross(*)
(J) movie_facebook_likes
(K) cast_total_facebook_likes
(L) movie_imdb_link
(M) facenumber_in_poster(*)
(N) content_rating(*)
(O) num_critic_for_reviews(*)
(P) num_user_for_reviews(*)
(Q) num_voted_users
(R) imdb_score
(S) director_name
(T) director_facebook_likes
(U) actor_1_name
(V) actor_1_facebook_likes
(W) actor_2_name(*)
(X) actor_2_facebook_likes(*)
(Y) actor_3_name(*)
(Z) actor_3_facebook_likes(*)

Deliverable 1.2. Find all functional dependencies, minimum cover and normalize the datasets to the 3NF:

Normalization to 1NF:

There exist multiple methods to convert data to 1NF. We will be using below two methods in normalization process from Zero-NF to 1NF in our project:

- Creating duplicate records will distribute the atomic values from one cell to multiple rows, and thus will create dataset that will be in 1NF. For column 'genres' in dataset, this approach is being used.
- Creating separate table will remove the column with non-atomic value and keep only reference in source table, and thus will create dataset that will be in 1NF. For column 'plot_keywords' in dataset, this approach is being used.

Dataset in 1NF:

{**A1, A2, G**}
{**A1, A2, A3**, B, C, D, E, F, H, I, J, L, M, L, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}

Functional Dependencies:

Description	**Relations**
(A1) movie_title, (A2) title_year, and (G) plot_keywords are only attributes in first table, and doesn't derive any other attribute.	(A1, A2, G) → {}
(A1) movie_title, (A2) title_year, and (A3) genres can be used to determine all other attributes in second table.	(A1, A2, A3) → {B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
(A1) movie_title, (A2) title_year, and (L) movie_imdb_link can be used to determine all other attributes except (A3) genres in second table.	(A1, A2, L) → {B, C, D, E, F, H, I, J, K, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
(A1) movie_title, and (A2) title_year can be used to determine all other attributes except (A3) genres in second table.	(A1, A2) → {B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
(L) movie_imdb_link can only be used to determine all other attributes except (A3) genres in second table.	(L) → {A1, A2, B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
(A1) movie_title, (A2) title_year, (A3) genres, and (L) movie_imdb_link can be used to determine all other attributes in second table.	(A1, A2, A3, L) → {B, C, D, E, F, H, I, J, K, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z}
(S) director_name can be used to determine (T) director_facebook_likes.	(S) → {T}
(U) actor_1_name can be used to determine (V) actor_1_facebook_likes.	(U) → {V}
(W) actor_2_name can be used to determine (X) actor_2_facebook_likes.	(W) → {X}
(Y) actor_3_name can be used to determine (Z) actor_3_facebook_likes.	(Y) → {Z}

Minimum cover:

Step 1	**Step 2**	**Step 3**	**Step 4**
Remove Trivial FD	**Reduce Right Side**	**Reduce Left Side**	**Eliminate Redundancy**
No Change			**Read NOTE (*)**

1. (A1, A2, G) → {} 2. (A1, A2, A3) → {B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z} 3. (A1, A2, L) → {B, C, D, E, F, H, I, J, K, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z} 4. (A1, A2) → {B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z} 5. (L) → {A1, A2, B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z} 6. (A1, A2, A3, L) → {B, C, D, E, F, H, I, J, K, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z} 7. (S) → {T} 8. (U) → {V} 9. (W) → {X} 10. (Y) → {Z}	1.1. (A1, A2, G) → {} 2.1. (A1, A2, A3) → B 2.2. (A1, A2, A3) → C 2.3. (A1, A2, A3) → D 2.4. (A1, A2, A3) → E 2.5. (A1, A2, A3) → F 2.6. (A1, A2, A3) → H 2.7. (A1, A2, A3) → I 2.8. (A1, A2, A3) → J 2.9. (A1, A2, A3) → K 2.10. (A1, A2, A3) → L 2.11. (A1, A2, A3) → M 2.12. (A1, A2, A3) → N 2.13. (A1, A2, A3) → O 2.14. (A1, A2, A3) → P 2.15. (A1, A2, A3) → Q 2.16. (A1, A2, A3) → R 2.17. (A1, A2, A3) → S 2.18. (A1, A2, A3) → T 2.19. (A1, A2, A3) → U 2.20. (A1, A2, A3) → V 2.21. (A1, A2, A3) → W 2.22. (A1, A2, A3) → X 2.23. (A1, A2, A3) → Y 2.24. (A1, A2, A3) → Z 3.1. (A1, A2, L) → B 3.2. (A1, A2, L) → C 3.3. (A1, A2, L) → D 3.4. (A1, A2, L) → E 3.5. (A1, A2, L) → F 3.6. (A1, A2, L) → H 3.7. (A1, A2, L) → I 3.8. (A1, A2, L) → J 3.9. (A1, A2, L) → K 3.10. (A1, A2, L) → M 3.11. (A1, A2, L) → N 3.12. (A1, A2, L) → O 3.13. (A1, A2, L) → P 3.14. (A1, A2, L) → Q 3.15. (A1, A2, L) → R 3.16. (A1, A2, L) → S 3.17. (A1, A2, L) → T 3.18. (A1, A2, L) → U 3.19. (A1, A2, L) → V 3.20. (A1, A2, L) → W 3.21. (A1, A2, L) → X 3.22. (A1, A2, L) → Y 3.23. (A1, A2, L) → Z 4.1. (A1, A2) → B 4.2. (A1, A2) → C 4.3. (A1, A2) → D 4.4. (A1, A2) → E 4.5. (A1, A2) → F	1.1. (A1, A2, G) → {} 2.1. (A1, A2, A3) → {} 2.2. (A1, A2) → B 2.3. (A1, A2) → C 2.4. (A1, A2) → D 2.5. (A1, A2) → E 2.6. (A1, A2) → F 2.7. (A1, A2) → H 2.8. (A1, A2) → I 2.9. (A1, A2) → J 2.10. (A1, A2) → K 2.11. (A1, A2) → L 2.12. (A1, A2) → M 2.13. (A1, A2) → N 2.14. (A1, A2) → O 2.15. (A1, A2) → P 2.16. (A1, A2) → Q 2.17. (A1, A2) → R 2.18. (A1, A2) → S 2.19. (A1, A2) → T 2.20. (A1, A2) → U 2.21. (A1, A2) → V 2.22. (A1, A2) → W 2.23. (A1, A2) → X 2.24. (A1, A2) → Y 2.25. (A1, A2) → Z 3.1. (A1, A2) → B 3.2. (A1, A2) → C 3.3. (A1, A2) → D 3.4. (A1, A2) → E 3.5. (A1, A2) → F 3.6. (A1, A2) → H 3.7. (A1, A2) → I 3.8. (A1, A2) → J 3.9. (A1, A2) → K 3.10. (A1, A2) → M 3.11. (A1, A2) → N 3.12. (A1, A2) → O 3.13. (A1, A2) → P 3.14. (A1, A2) → Q 3.15. (A1, A2) → R 3.16. (A1, A2) → S 3.17. (A1, A2) → T 3.18. (A1, A2) → U 3.19. (A1, A2) → V 3.20. (A1, A2) → W 3.21. (A1, A2) → X 3.22. (A1, A2) → Y 3.23. (A1, A2) → Z 3.24. (L) → B 3.25. (L) → C 3.26. (L) → D 3.27. (L) → E 3.28. (L) → F	1.1. (A1, A2, G) → {} 2.1. (A1, A2, A3) → {} 2.2. (A1, A2) → B 2.3. (A1, A2) → C 2.4. (A1, A2) → D 2.5. (A1, A2) → E 2.6. (A1, A2) → F 2.7. (A1, A2) → H 2.8. (A1, A2) → I 2.9. (A1, A2) → J 2.10. (A1, A2) → K ~~2.11. (A1, A2) → J~~ ~~2.12. (A1, A2) → M~~ ~~2.13. (A1, A2) → N~~ ~~2.14. (A1, A2) → O~~ ~~2.15. (A1, A2) → P~~ ~~2.16. (A1, A2) → Q~~ ~~2.17. (A1, A2) → R~~ 2.18. (A1, A2) → S ~~2.19. (A1, A2) → T~~ 2.20. (A1, A2) → U ~~2.21. (A1, A2) → V~~ 2.22. (A1, A2) → W ~~2.23. (A1, A2) → X~~ 2.24. (A1, A2) → Y ~~2.25. (A1, A2) → Z~~ ~~3.1. (A1, A2) → B~~ ~~3.2. (A1, A2) → C~~ ~~3.3. (A1, A2) → D~~ ~~3.4. (A1, A2) → E~~ ~~3.5. (A1, A2) → F~~ ~~3.6. (A1, A2) → H~~ ~~3.7. (A1, A2) → I~~ ~~3.8. (A1, A2) → J~~ ~~3.9. (A1, A2) → K~~ ~~3.10. (A1, A2) → M~~ ~~3.11. (A1, A2) → N~~ ~~3.12. (A1, A2) → O~~ ~~3.13. (A1, A2) → P~~ ~~3.14. (A1, A2) → Q~~ ~~3.15. (A1, A2) → R~~ ~~3.16. (A1, A2) → S~~ ~~3.17. (A1, A2) → T~~ ~~3.18. (A1, A2) → U~~ ~~3.19. (A1, A2) → V~~ ~~3.20. (A1, A2) → W~~ ~~3.21. (A1, A2) → X~~ ~~3.22. (A1, A2) → Y~~ ~~3.23. (A1, A2) → Z~~ ~~3.24. (L) → B~~ ~~3.25. (L) → C~~ ~~3.26. (L) → D~~ ~~3.27. (L) → E~~ ~~3.28. (L) → F~~

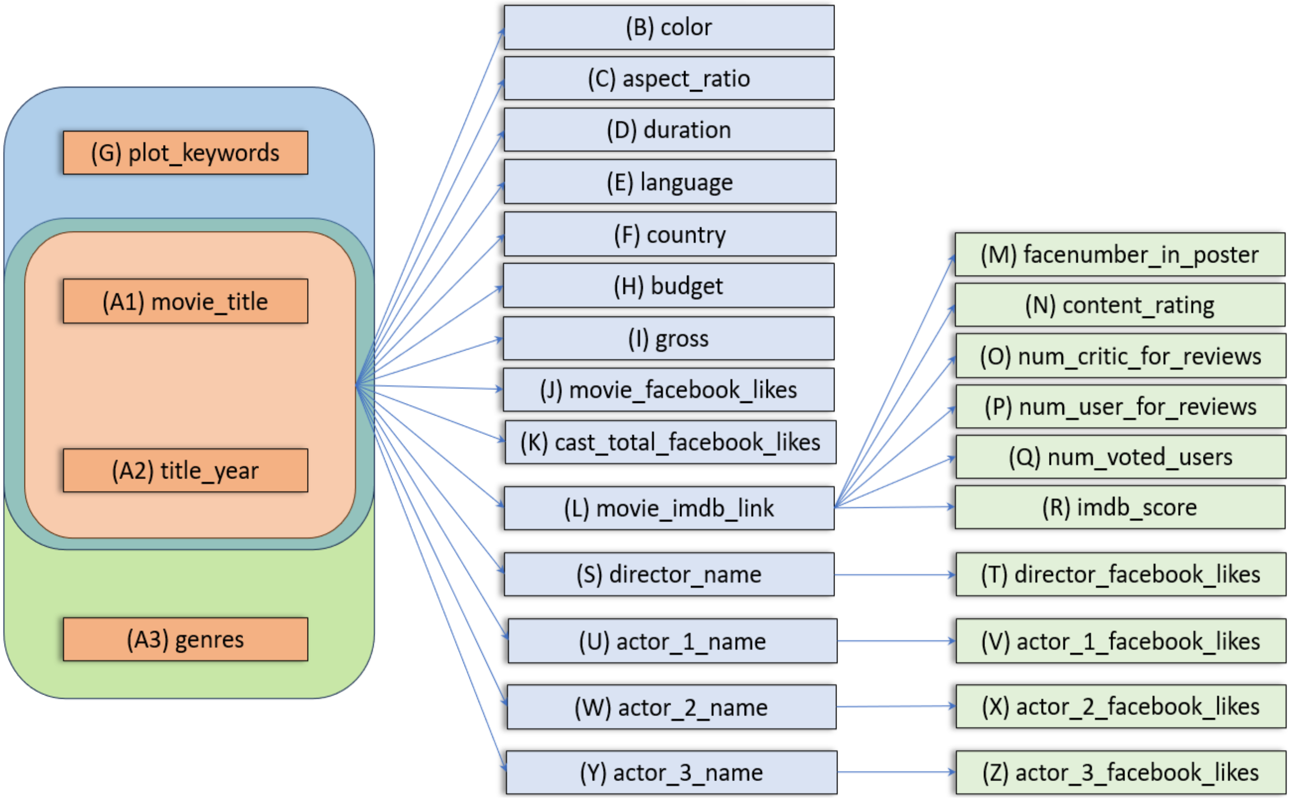
	4.6. (A1, A2) → H	3.29. (L) → H	~~3.29. (L) → H~~
	4.7. (A1, A2) → I	3.30. (L) → I	~~3.30. (L) → I~~
	4.8. (A1, A2) → J	3.31. (L) → J	~~3.31. (L) → J~~
	4.9. (A1, A2) → K	3.32. (L) → K	~~3.32. (L) → K~~
	4.10. (A1, A2) → L	3.33. (L) → M	3.33. (L) → M
	4.11. (A1, A2) → M	3.34. (L) → N	3.34. (L) → N
	4.12. (A1, A2) → N	3.35. (L) → O	3.35. (L) → O
	4.13. (A1, A2) → O	3.36. (L) → P	3.36. (L) → P
	4.14. (A1, A2) → P	3.37. (L) → Q	3.37. (L) → Q
	4.15. (A1, A2) → Q	3.38. (L) → R	3.38. (L) → R
	4.16. (A1, A2) → R	3.39. (L) → S	~~3.39. (L) → S~~
	4.17. (A1, A2) → S	3.40. (L) → T	~~3.40. (L) → T~~
	4.18. (A1, A2) → T	3.41. (L) → U	~~3.41. (L) → U~~
	4.19. (A1, A2) → U	3.42. (L) → V	~~3.42. (L) → V~~
	4.20. (A1, A2) → V	3.43. (L) → W	~~3.43. (L) → W~~
	4.21. (A1, A2) → W	3.44. (L) → X	~~3.44. (L) → X~~
	4.22. (A1, A2) → X	3.45. (L) → Y	~~3.45. (L) → Y~~
	4.23. (A1, A2) → Y	3.46. (L) → Z	~~3.46. (L) → Z~~
	4.24. (A1, A2) → Z		
		4.1. (A1, A2) → B	~~4.1. (A1, A2) → B~~
	5.1. (L) → A1	4.2. (A1, A2) → C	~~4.2. (A1, A2) → C~~
	5.2. (L) → A2	4.3. (A1, A2) → D	~~4.3. (A1, A2) → D~~
	5.3. (L) → B	4.4. (A1, A2) → E	~~4.4. (A1, A2) → E~~
	5.4. (L) → C	4.5. (A1, A2) → F	~~4.5. (A1, A2) → F~~
	5.5. (L) → D	4.6. (A1, A2) → H	~~4.6. (A1, A2) → H~~
	5.6. (L) → E	4.7. (A1, A2) → I	~~4.7. (A1, A2) → I~~
	5.7. (L) → F	4.8. (A1, A2) → J	~~4.8. (A1, A2) → J~~
	5.8. (L) → H	4.9. (A1, A2) → K	~~4.9. (A1, A2) → K~~
	5.9. (L) → I	4.10. (A1, A2) → L	4.10. (A1, A2) → L
	5.10. (L) → J	4.11. (A1, A2) → M	~~4.11. (A1, A2) → M~~
	5.11. (L) → K	4.12. (A1, A2) → N	~~4.12. (A1, A2) → N~~
	5.12. (L) → M	4.13. (A1, A2) → O	~~4.13. (A1, A2) → O~~
	5.13. (L) → N	4.14. (A1, A2) → P	~~4.14. (A1, A2) → P~~
	5.14. (L) → O	4.15. (A1, A2) → Q	~~4.15. (A1, A2) → Q~~
	5.15. (L) → P	4.16. (A1, A2) → R	~~4.16. (A1, A2) → R~~
	5.16. (L) → Q	4.17. (A1, A2) → S	~~4.17. (A1, A2) → S~~
	5.17. (L) → R	4.18. (A1, A2) → T	~~4.18. (A1, A2) → T~~
	5.18. (L) → S	4.19. (A1, A2) → U	~~4.19. (A1, A2) → U~~
	5.19. (L) → T	4.20. (A1, A2) → V	~~4.20. (A1, A2) → V~~
	5.20. (L) → U	4.21. (A1, A2) → W	~~4.21. (A1, A2) → W~~
	5.21. (L) → V	4.22. (A1, A2) → X	~~4.22. (A1, A2) → X~~
	5.22. (L) → W	4.23. (A1, A2) → Y	~~4.23. (A1, A2) → Y~~
	5.23. (L) → X	4.24. (A1, A2) → Z	~~4.24. (A1, A2) → Z~~
	5.24. (L) → Z		
		5.1. (L) → A1	~~5.1. (L) → A1~~
	6.1. (A1,A2,A3,L) → B	5.2. (L) → A2	~~5.2. (L) → A2~~
	6.2. (A1,A2,A3,L) → C	5.3. (L) → B	~~5.3. (L) → B~~
	6.3. (A1,A2,A3,L) → D	5.4. (L) → C	~~5.4. (L) → C~~
	6.4. (A1,A2,A3,L) → E	5.5. (L) → D	~~5.5. (L) → D~~
	6.5. (A1,A2,A3,L) → F	5.6. (L) → E	~~5.6. (L) → E~~
	6.6. (A1,A2,A3,L) → H	5.7. (L) → F	~~5.7. (L) → F~~
	6.7. (A1,A2,A3,L) → I	5.8. (L) → H	~~5.8. (L) → H~~
	6.8. (A1,A2,A3,L) → J	5.9. (L) → I	~~5.9. (L) → I~~
	6.9. (A1,A2,A3,L) → K	5.10. (L) → J	~~5.10. (L) → J~~
	6.10. (A1,A2,A3,L) → M	5.11. (L) → K	~~5.11. (L) → K~~
	6.11. (A1,A2,A3,L) → N	5.12. (L) → M	~~5.12. (L) → M~~

	6.12. (A1,A2,A3,L) → O	5.13. (L) → N	~~5.13. (L) → N~~
	6.13. (A1,A2,A3,L) → P	5.14. (L) → O	~~5.14. (L) → O~~
	6.14. (A1,A2,A3,L) → Q	5.15. (L) → P	~~5.15. (L) → P~~
	6.15. (A1,A2,A3,L) → R	5.16. (L) → Q	~~5.16. (L) → Q~~
	6.16. (A1,A2,A3,L) → S	5.17. (L) → R	~~5.17. (L) → R~~
	6.17. (A1,A2,A3,L) → T	5.18. (L) → S	~~5.18. (L) → S~~
	6.18. (A1,A2,A3,L) → U	5.19. (L) → T	~~5.19. (L) → T~~
	6.19. (A1,A2,A3,L) → V	5.20. (L) → U	~~5.20. (L) → U~~
	6.20. (A1,A2,A3,L) → W	5.21. (L) → V	~~5.21. (L) → V~~
	6.21. (A1,A2,A3,L) → X	5.22. (L) → W	~~5.22. (L) → W~~
	6.22. (A1,A2,A3,L) → Y	5.23. (L) → X	~~5.23. (L) → X~~
	6.23. (A1,A2,A3,L) → Z	5.24. (L) → Y	~~5.24. (L) → Y~~
		5.25. (L) → Z	~~5.25. (L) → Z~~
	7.1. (S) → T		
		6.1. (A1, A2) → B	~~6.1. (A1, A2) → B~~
	8.1. (U) → V	6.2. (A1, A2) → C	~~6.2. (A1, A2) → C~~
		6.3. (A1, A2) → D	~~6.3. (A1, A2) → D~~
	9.1. (W) → X	6.4. (A1, A2) → E	~~6.4. (A1, A2) → E~~
		6.5. (A1, A2) → F	~~6.5. (A1, A2) → F~~
	10.1. (Y) → Z	6.6. (A1, A2) → H	~~6.6. (A1, A2) → H~~
		6.7. (A1, A2) → I	~~6.7. (A1, A2) → I~~
		6.8. (A1, A2) → J	~~6.8. (A1, A2) → J~~
		6.9. (A1, A2) → K	~~6.9. (A1, A2) → K~~
		6.10. (A1, A2) → M	~~6.10. (A1, A2) → M~~
		6.11. (A1, A2) → N	~~6.11. (A1, A2) → N~~
		6.12. (A1, A2) → O	~~6.12. (A1, A2) → O~~
		6.13. (A1, A2) → P	~~6.13. (A1, A2) → P~~
		6.14. (A1, A2) → Q	~~6.14. (A1, A2) → Q~~
		6.15. (A1, A2) → R	~~6.15. (A1, A2) → R~~
		6.16. (A1, A2) → S	~~6.16. (A1, A2) → S~~
		6.17. (A1, A2) → T	~~6.17. (A1, A2) → T~~
		6.18. (A1, A2) → U	~~6.18. (A1, A2) → U~~
		6.19. (A1, A2) → V	~~6.19. (A1, A2) → V~~
		6.20. (A1, A2) → W	~~6.20. (A1, A2) → W~~
		6.21. (A1, A2) → X	~~6.21. (A1, A2) → X~~
		6.22. (A1, A2) → Y	~~6.22. (A1, A2) → Y~~
		6.23. (A1, A2) → Z	~~6.23. (A1, A2) → Z~~
		6.24. (L) → B	~~6.24. (L) → B~~
		6.25. (L) → C	~~6.25. (L) → C~~
		6.26. (L) → D	~~6.26. (L) → D~~
		6.27. (L) → E	~~6.27. (L) → E~~
		6.28. (L) → F	~~6.28. (L) → F~~
		6.29. (L) → H	~~6.29. (L) → H~~
		6.30. (L) → I	~~6.30. (L) → I~~
		6.31. (L) → J	~~6.31. (L) → J~~
		6.32. (L) → K	~~6.32. (L) → K~~
		6.33. (L) → M	~~6.33. (L) → M~~
		6.34. (L) → N	~~6.34. (L) → N~~
		6.35. (L) → O	~~6.35. (L) → O~~
		6.36. (L) → P	~~6.36. (L) → P~~
		6.37. (L) → Q	~~6.37. (L) → Q~~
		6.38. (L) → R	~~6.38. (L) → R~~
		6.39. (L) → S	~~6.39. (L) → S~~
		6.40. (L) → T	~~6.40. (L) → T~~
		6.41. (L) → U	~~6.41. (L) → U~~
		6.42. (L) → V	~~6.42. (L) → V~~

		6.43. (L) → W 6.44. (L) → X 6.45. (L) → Y 6.46. (L) → Z 7.1. (S) → T 8.1. (U) → V 9.1. (W) → X 10.1. (Y) → Z	~~6.43. (L) → W~~ ~~6.44. (L) → X~~ ~~6.45. (L) → Y~~ ~~6.46. (L) → Z~~ 7.1. (S) → T 8.1. (U) → V 9.1. (W) → X 10.1. (Y) → Z

NOTE (*): Due to one-to-one correspondence between {(A1) movie_title, (A2) title_year} and { (L) movie_imdb_link}, we need to assume one as primary key, and another as candidate key. For simplicity it is assumed {(A1) movie_title, (A2) title_year} is kept as primary attribute. Also, the attributes primary dependent on movie_imdb_link web page are kept associated on same.

Functional Dependency Diagram:



Super key identification:

Super key for full dataset (whether in Zero-NF or any other normal form) is as follows:

{(A1) movie_title, (A2) title_year, (A3) genres, (G) plot_keywords}

Which is union of below two relations:

- Unique key for first relation is all three attributes i.e. {(A1) movie_title, (A2) title_year, (G) plot_keywords}, and
- Unique key for second relation is {(A1) movie_title, (A2) title_year, (A3) genres}.

Converting relation to 2NF:

For a relation to be in 2NF, all non-keys attributes should be fully functionally dependent on primary key. Here, there exist attributes who are dependent on part of primary key, so final relation needs to be splitted in following relations based on part of key attributes are dependent: -

- R(**A1, A2, G**)
- R(**A1, A2, A3**)
- R(**A1, A2**, B, C, D, E, F, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z)

Now, above relation is in 2NF.

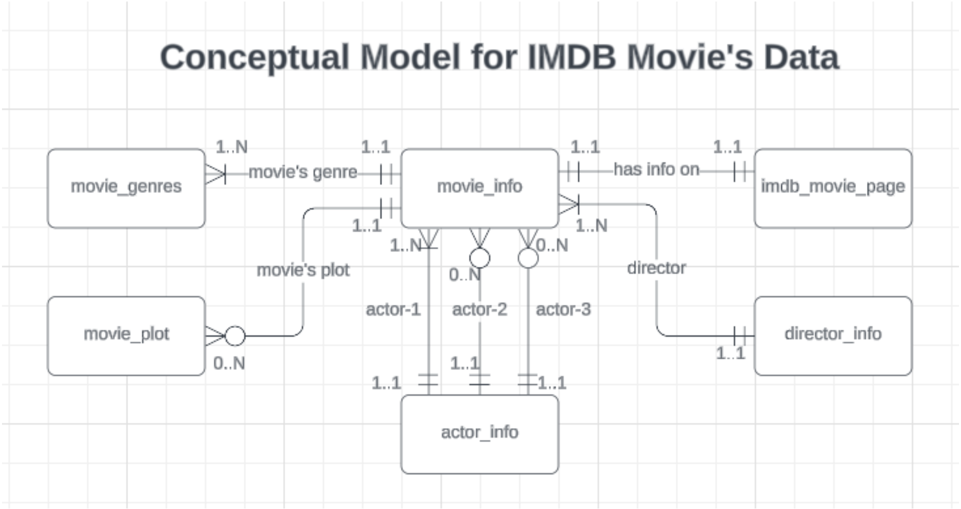
Converting relation to 3NF:

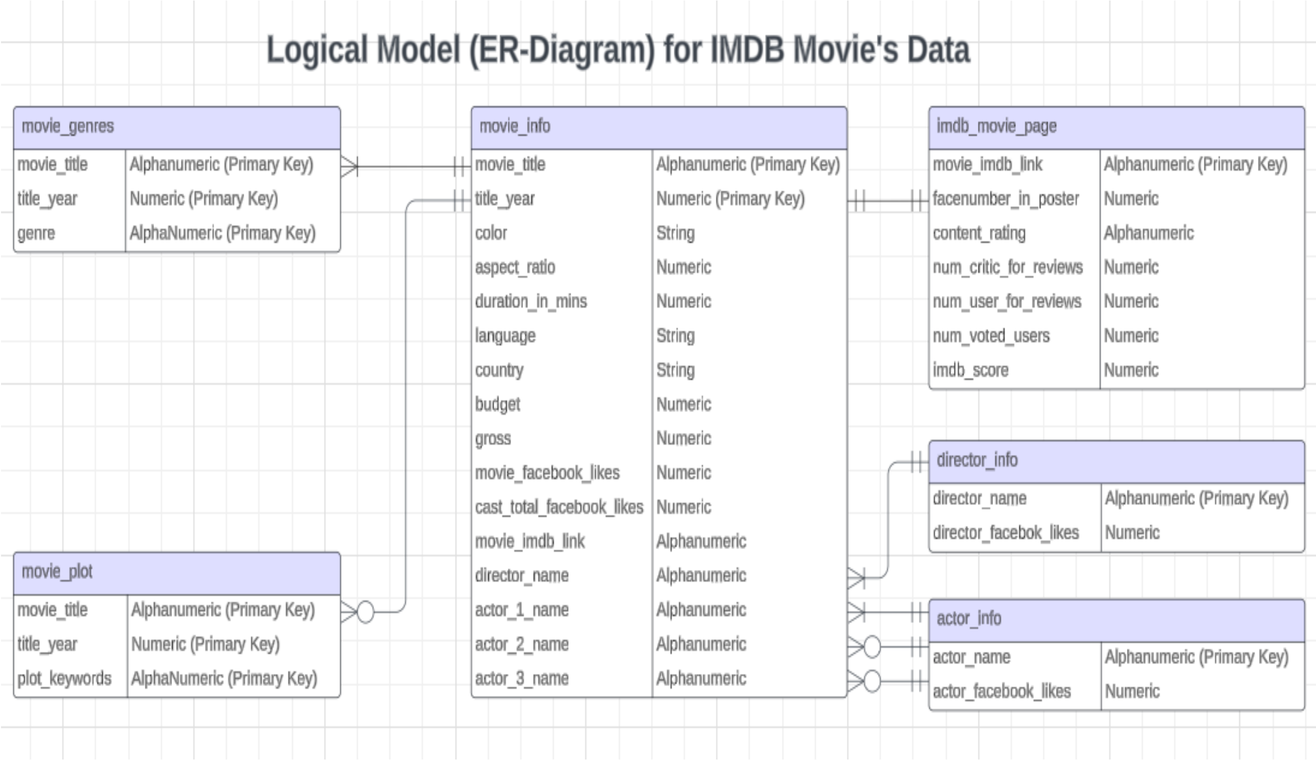
Above relation is in 2NF, however it is not in 3NF because there exists a non-key attribute that transitively dependent on primary key. So, we need to further break the relation in below mentioned relations:

- R(**A1, A2, G**)
- R(**A1, A2, A3**)
- R(**A1, A2**, B, C, D, E, F, H, I, J, K, L, S, U, W, Y)
- R(**L**, M, N, O, P, Q, R)
- R(**S**, T)
- R(**U**, V)
- R(**W**, X)
- R(**Y**, Z)

Now relation is in 3NF.

Deliverable 1.3. Create logical data model and ER





Logical Model (ER-Diagram) for IMDB Movie's Data

movie_genres

movie_title	Alphanumeric (Primary Key)
title_year	Numeric (Primary Key)
genre	AlphaNumeric (Primary Key)

movie_info

movie_title	Alphanumeric (Primary Key)
title_year	Numeric (Primary Key)
color	String
aspect_ratio	Numeric
duration_in_mins	Numeric
language	String
country	String
budget	Numeric
gross	Numeric
movie_facebook_likes	Numeric
cast_total_facebook_likes	Numeric
movie_imdb_link	Alphanumeric
director_name	Alphanumeric
actor_1_name	Alphanumeric
actor_2_name	Alphanumeric
actor_3_name	Alphanumeric

imdb_movie_page

movie_imdb_link	Alphanumeric (Primary Key)
facenumber_in_poster	Numeric
content_rating	Alphanumeric
num_critic_for_reviews	Numeric
num_user_for_reviews	Numeric
num_voted_users	Numeric
imdb_score	Numeric

director_info

director_name	Alphanumeric (Primary Key)
director_facebok_likes	Numeric

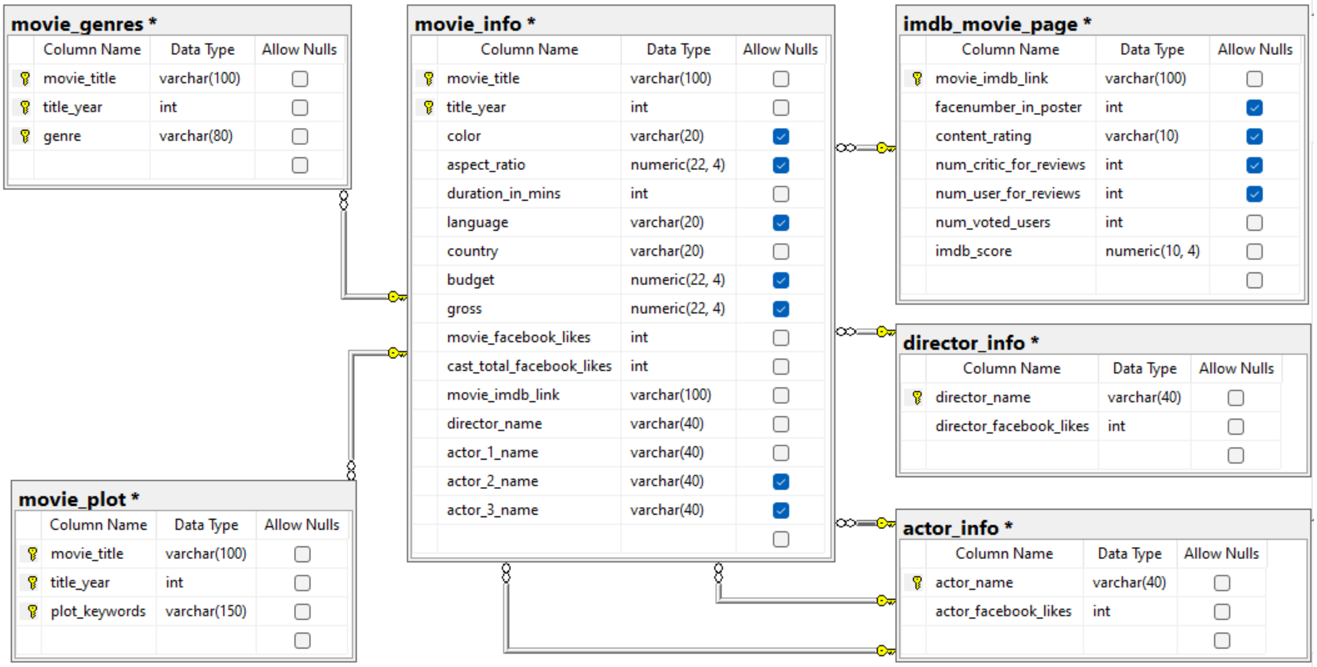
movie_plot

movie_title	Alphanumeric (Primary Key)
title_year	Numeric (Primary Key)
plot_keywords	AlphaNumeric (Primary Key)

actor_info

actor_name	Alphanumeric (Primary Key)
actor_facebook_likes	Numeric

Deliverable 1.4. <u>Create physical data model for all entities</u>



movie_genres *

	Column Name	Data Type	Allow Nulls
🔑	movie_title	varchar(100)	☐
🔑	title_year	int	☐
🔑	genre	varchar(80)	☐
			☐

movie_info *

	Column Name	Data Type	Allow Nulls
🔑	movie_title	varchar(100)	☐
🔑	title_year	int	☐
	color	varchar(20)	☑
	aspect_ratio	numeric(22, 4)	☑
	duration_in_mins	int	☐
	language	varchar(20)	☑
	country	varchar(20)	☐
	budget	numeric(22, 4)	☑
	gross	numeric(22, 4)	☑
	movie_facebook_likes	int	☐
	cast_total_facebook_likes	int	☐
	movie_imdb_link	varchar(100)	☐
	director_name	varchar(40)	☐
	actor_1_name	varchar(40)	☐
	actor_2_name	varchar(40)	☑
	actor_3_name	varchar(40)	☑
			☐

imdb_movie_page *

	Column Name	Data Type	Allow Nulls
🔑	movie_imdb_link	varchar(100)	☐
	facenumber_in_poster	int	☑
	content_rating	varchar(10)	☑
	num_critic_for_reviews	int	☑
	num_user_for_reviews	int	☑
	num_voted_users	int	☐
	imdb_score	numeric(10, 4)	☐
			☐

director_info *

	Column Name	Data Type	Allow Nulls
🔑	director_name	varchar(40)	☐
	director_facebook_likes	int	☐
			☐

movie_plot *

	Column Name	Data Type	Allow Nulls
🔑	movie_title	varchar(100)	☐
🔑	title_year	int	☐
🔑	plot_keywords	varchar(150)	☐
			☐

actor_info *

	Column Name	Data Type	Allow Nulls
🔑	actor_name	varchar(40)	☐
	actor_facebook_likes	int	☐
			☐

Deliverable 1.5. Create DDL statement

DDL script file is also shared and can be used to run or validate create table scripts. DDL script for each table is as follows:

DDL Script for table actor_info:

```
CREATE TABLE dbimdb.actor_info
(
      actor_name                  VARCHAR(40)       NOT NULL,
      actor_facebook_likes        INT               NOT NULL,
      CONSTRAINT PK_actor_info
                 PRIMARY KEY (actor_name)     /* Primary Key */
);
```

DDL Script for table director_info:

```
CREATE TABLE dbimdb.director_info
(
      director_name               VARCHAR(40)       NOT NULL,
      director_facebook_likes     INT               NOT NULL,
      CONSTRAINT PK_director_info
                 PRIMARY KEY (director_name)        /* Primary Key */
);
```

DDL Script for table imdb_movie_page:

```
CREATE TABLE dbimdb.imdb_movie_page
(
      movie_imdb_link             VARCHAR(100)      NOT NULL,
      facenumber_in_poster        INT,
      content_rating              VARCHAR(10),
      num_critic_for_reviews      INT,
      num_user_for_reviews        INT,
      num_voted_users             INT               NOT NULL,
      imdb_score                  NUMERIC(10, 4)    NOT NULL,
      CONSTRAINT PK_imdb_movie_page
                 PRIMARY KEY (movie_imdb_link)      /* Primary Key */
);
```

DDL Script for table movie_info:

```
CREATE TABLE dbimdb.movie_info
(
      movie_title                 VARCHAR(100)           NOT NULL,
      title_year                  INT                    NOT NULL,
      color                       VARCHAR(20),
      aspect_ratio                NUMERIC(22, 4),
      duration_in_mins            INT                    NOT NULL,
      language                    VARCHAR(20),
      country                     VARCHAR(20)            NOT NULL,
      budget                      NUMERIC(22, 4),
      gross                       NUMERIC(22, 4),
      movie_facebook_likes        INT                    NOT NULL,
      cast_total_facebook_likes   INT                    NOT NULL,
      movie_imdb_link             VARCHAR(100)           NOT NULL,
      director_name               VARCHAR(40)            NOT NULL,
      actor_1_name                VARCHAR(40)            NOT NULL,
      actor_2_name                VARCHAR(40),
      actor_3_name                VARCHAR(40),
      CONSTRAINT PK_movie_info
                 PRIMARY KEY (movie_title, title_year),  /* Primary Key */
      CONSTRAINT FK_movie_info_imdb_movie_page
                 FOREIGN KEY (movie_imdb_link)
                 REFERENCES dbimdb.imdb_movie_page (movie_imdb_link),
      CONSTRAINT FK_movie_info_director_info
                 FOREIGN KEY (director_name)
                 REFERENCES dbimdb.director_info (director_name),
      CONSTRAINT FK_movie_info_actor1_info
                 FOREIGN KEY (actor_1_name)
                 REFERENCES dbimdb.actor_info (actor_name),
      CONSTRAINT FK_movie_info_actor2_info
                 FOREIGN KEY (actor_2_name)
                 REFERENCES dbimdb.actor_info (actor_name),
      CONSTRAINT FK_movie_info_actor3_info
                 FOREIGN KEY (actor_3_name)
                 REFERENCES dbimdb.actor_info (actor_name),
      CONSTRAINT CHK_Title_Year
                 CHECK (title_year>= 1900    AND   title_year <=2100)
);
```

DDL Script for table movie_genres:

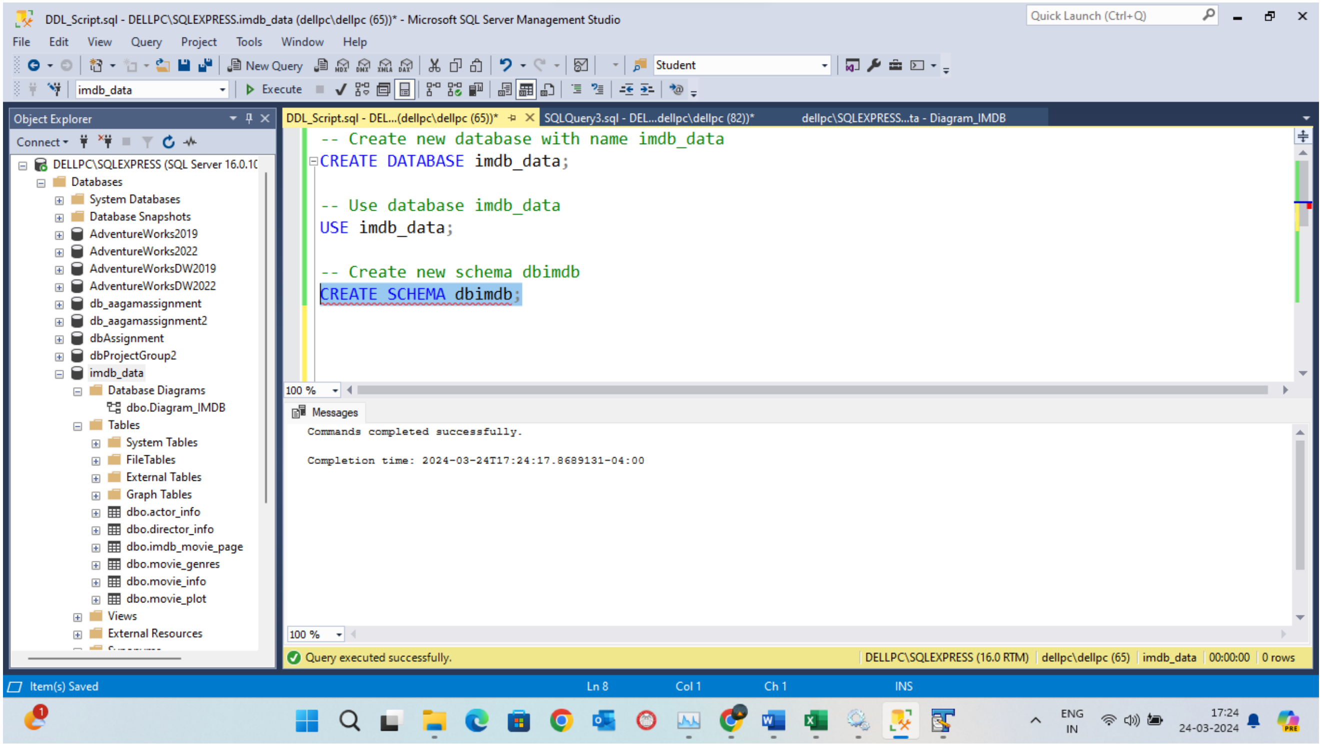
```
CREATE TABLE dbimdb.movie_genres
(
      movie_title                       VARCHAR(100)     NOT NULL,
      title_year                        INT              NOT NULL,
      genre                             VARCHAR(80)      NOT NULL,
      CONSTRAINT PK_movie_genres
                 PRIMARY KEY (movie_title, title_year, genre),/* PrimKey */
      CONSTRAINT FK_movie_genres_movie_info
                 FOREIGN KEY (movie_title, title_year)
                 REFERENCES dbimdb.movie_info (movie_title, title_year)
);
```

DDL Script for table movie_plot:

```
CREATE TABLE dbimdb.movie_plot
(
      movie_title             VARCHAR(100)     NOT NULL,
      title_year              INT              NOT NULL,
      plot_keywords           VARCHAR(150)     NOT NULL,
      CONSTRAINT  PK_movie_plot
                  PRIMARY KEY (movie_title, title_year, plot_keywords),/* PK */
      CONSTRAINT  FK_movie_plot_movie_info
                  FOREIGN KEY (movie_title, title_year)
                  REFERENCES dbimdb.movie_info (movie_title, title_year)
);
```

Deliverable 1.6. Create physical tables in the SQL Server database

Screenshot for Create Schema

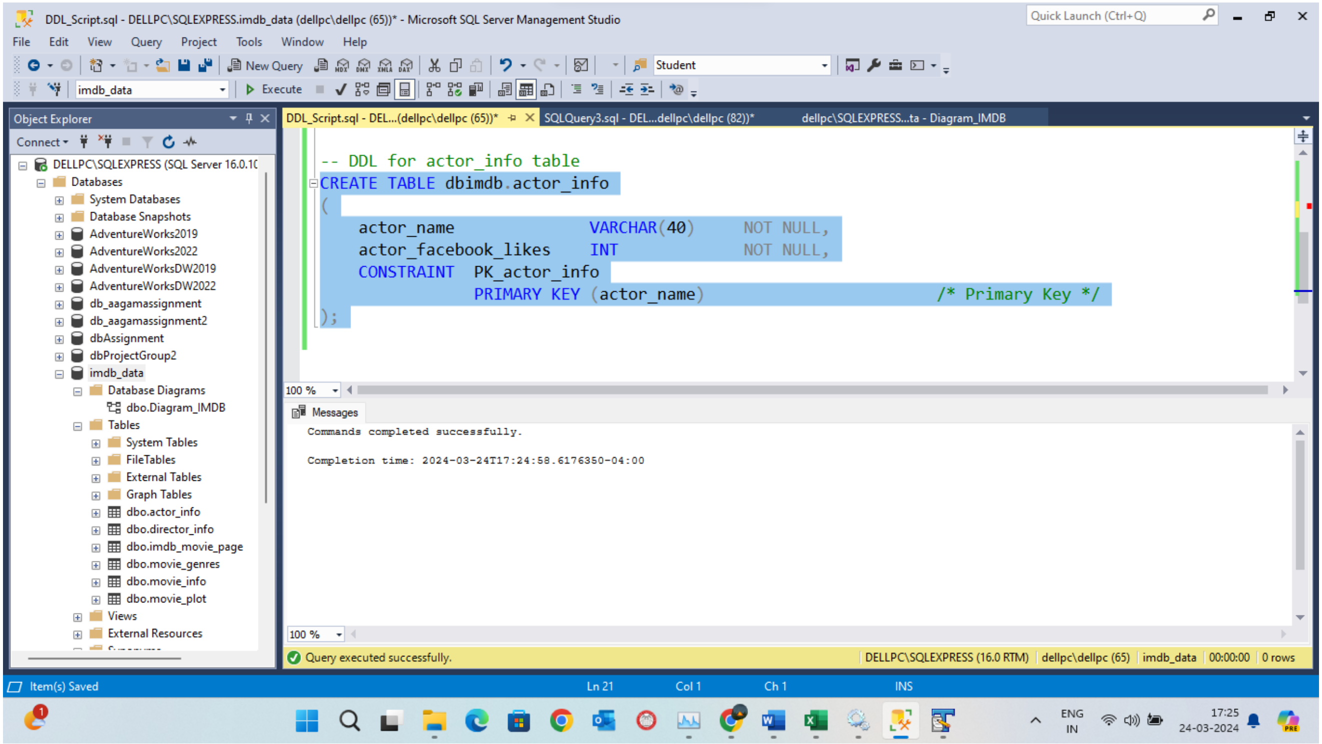


```
-- Create new database with name imdb_data
CREATE DATABASE imdb_data;

-- Use database imdb_data
USE imdb_data;

-- Create new schema dbimdb
CREATE SCHEMA dbimdb;
```

```
Commands completed successfully.

Completion time: 2024-03-24T17:24:17.8689131-04:00
```

Query executed successfully. | DELLPC\SQLEXPRESS (16.0 RTM) | dellpc\dellpc (65) | imdb_data | 00:00:00 | 0 rows

Screenshot for Create table actor_info:



DDL_Script.sql - DELLPC\SQLEXPRESS.imdb_data (dellpc\dellpc (65))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
imdb_data
Student
Object Explorer
DDL_Script.sql - DEL...(dellpc\dellpc (65))*
SQLQuery3.sql - DEL...dellpc\dellpc (82))*
dellpc\SQLEXPRESS...ta - Diagram_IMDB
Item(s) Saved
Ln 21
Col 1
Ch 1
INS

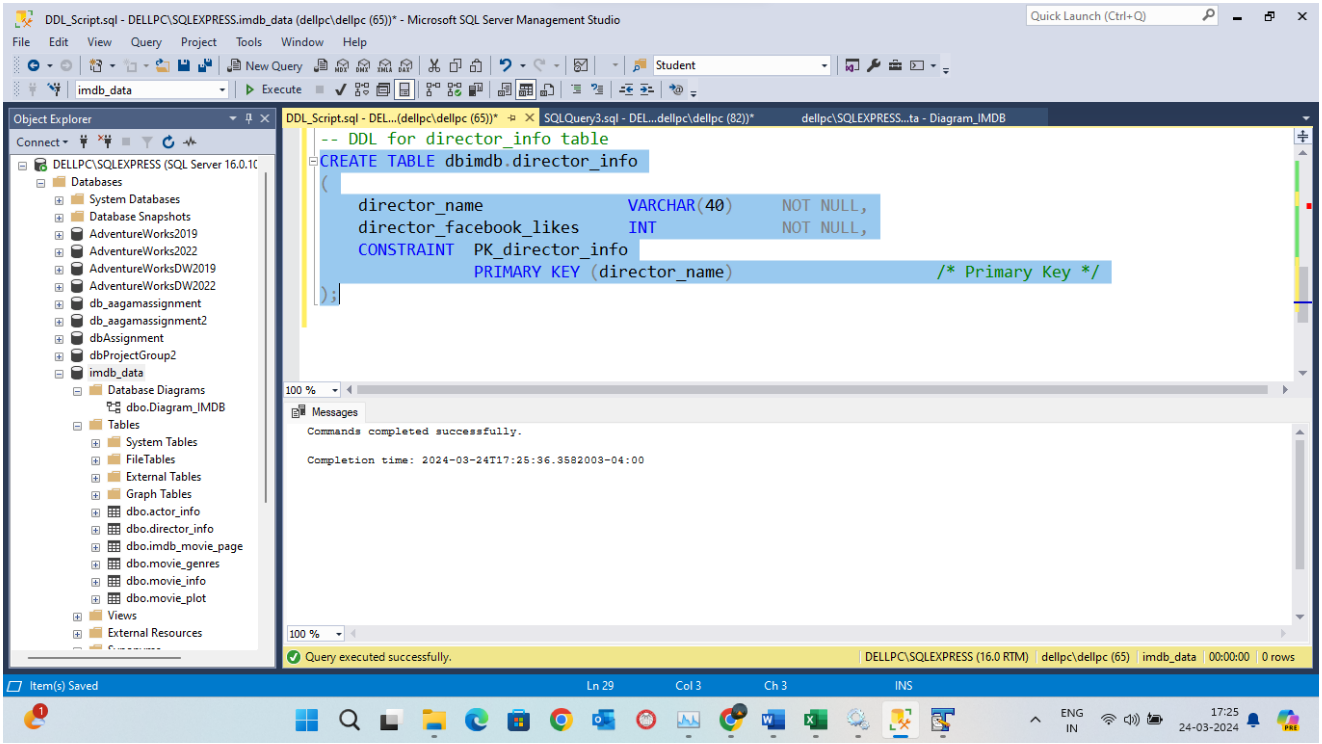
```
-- DDL for actor_info table
CREATE TABLE dbimdb.actor_info
(
    actor_name              VARCHAR(40)     NOT NULL,
    actor_facebook_likes    INT             NOT NULL,
    CONSTRAINT  PK_actor_info
                PRIMARY KEY (actor_name)                    /* Primary Key */
);
```

Messages

```
Commands completed successfully.

Completion time: 2024-03-24T17:24:58.6176350-04:00
```

Query executed successfully. | DELLPC\SQLEXPRESS (16.0 RTM) | dellpc\dellpc (65) | imdb_data | 00:00:00 | 0 rows

Screenshot for Create table director_info:



DDL_Script.sql - DELLPC\SQLEXPRESS.imdb_data (dellpc\dellpc (65))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Execute
imdb_data
Student
Object Explorer
DDL_Script.sql - DEL...(dellpc\dellpc (65))*
SQLQuery3.sql - DEL...dellpc\dellpc (82))*
dellpc\SQLEXPRESS...ta - Diagram_IMDB
Item(s) Saved
Ln 29
Col 3
Ch 3
INS

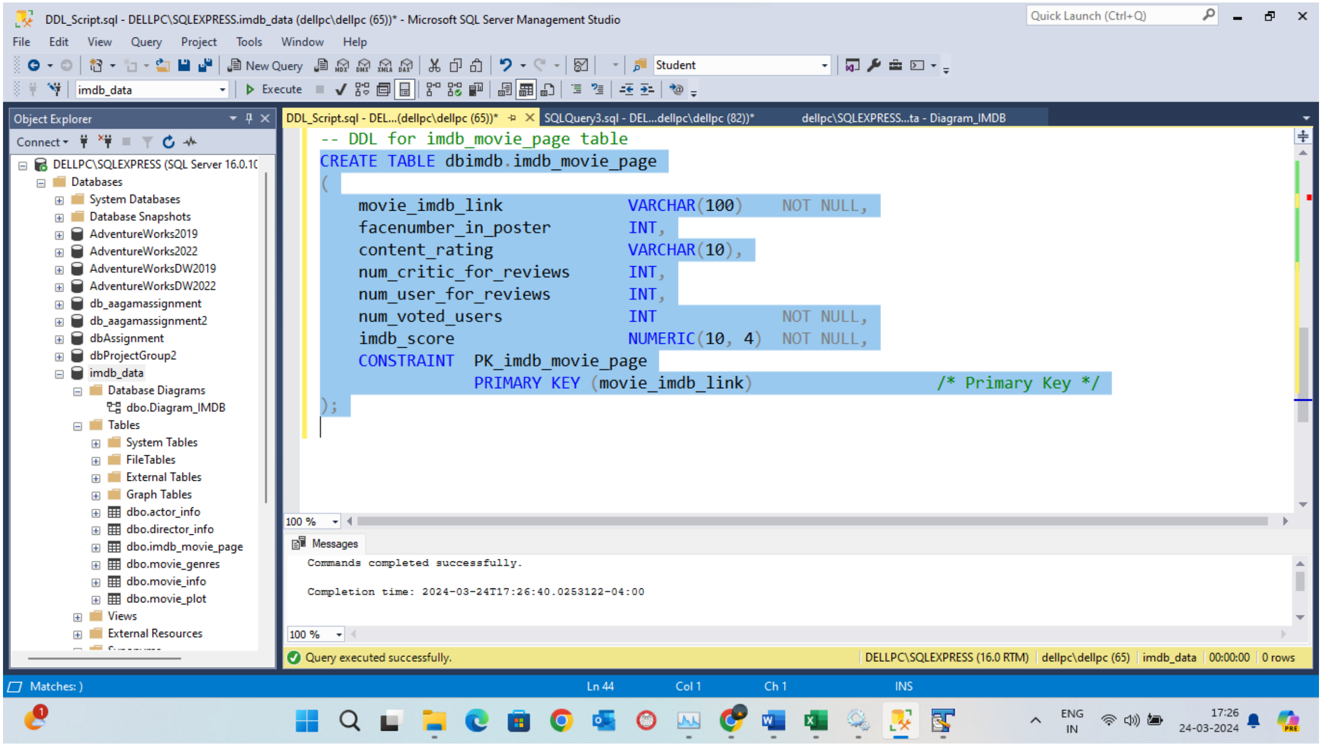
```
-- DDL for director_info table
CREATE TABLE dbimdb.director_info
(
    director_name              VARCHAR(40)     NOT NULL,
    director_facebook_likes    INT             NOT NULL,
    CONSTRAINT  PK_director_info
                PRIMARY KEY (director_name)                 /* Primary Key */
);
```

Messages

```
Commands completed successfully.

Completion time: 2024-03-24T17:25:36.3582003-04:00
```

Query executed successfully. | DELLPC\SQLEXPRESS (16.0 RTM) | dellpc\dellpc (65) | imdb_data | 00:00:00 | 0 rows

Screenshot for Create table imdb_movie_page:

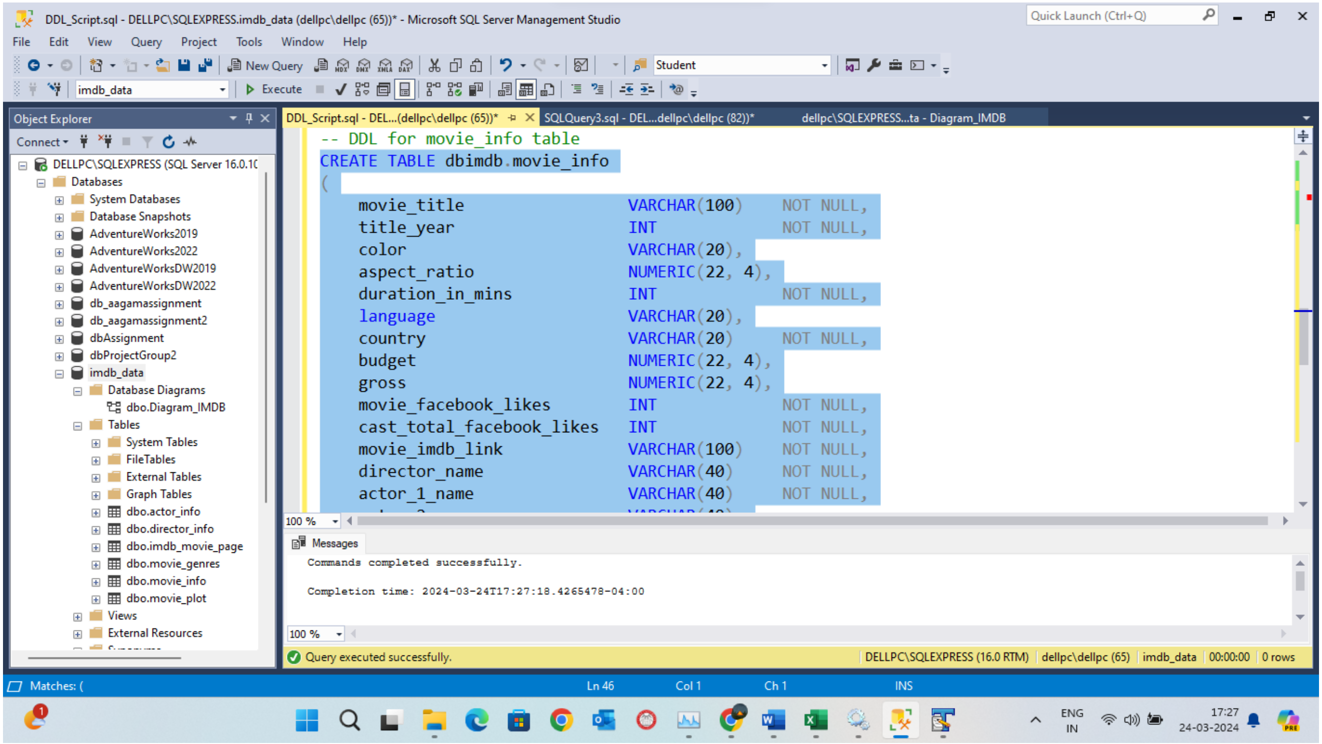


DDL_Script.sql - DELLPC\SQLEXPRESS.imdb_data (dellpc\dellpc (65))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Student
imdb_data
Execute
Object Explorer
DDL_Script.sql - DEL...(dellpc\dellpc (65))*
SQLQuery3.sql - DEL...dellpc\dellpc (82))*
dellpc\SQLEXPRESS...ta - Diagram_IMDB
Messages
Query executed successfully.
DELLPC\SQLEXPRESS (16.0 RTM) dellpc\dellpc (65) imdb_data 00:00:00 0 rows
Ln 44 Col 1 Ch 1 INS

```
-- DDL for imdb_movie_page table
CREATE TABLE dbimdb.imdb_movie_page
(
    movie_imdb_link             VARCHAR(100)   NOT NULL,
    facenumber_in_poster        INT,
    content_rating              VARCHAR(10),
    num_critic_for_reviews      INT,
    num_user_for_reviews        INT,
    num_voted_users             INT            NOT NULL,
    imdb_score                  NUMERIC(10, 4)  NOT NULL,
    CONSTRAINT  PK_imdb_movie_page
                PRIMARY KEY (movie_imdb_link)                    /* Primary Key */
);
```

```
Commands completed successfully.

Completion time: 2024-03-24T17:26:40.0253122-04:00
```

Screenshot for Create table movie_info:

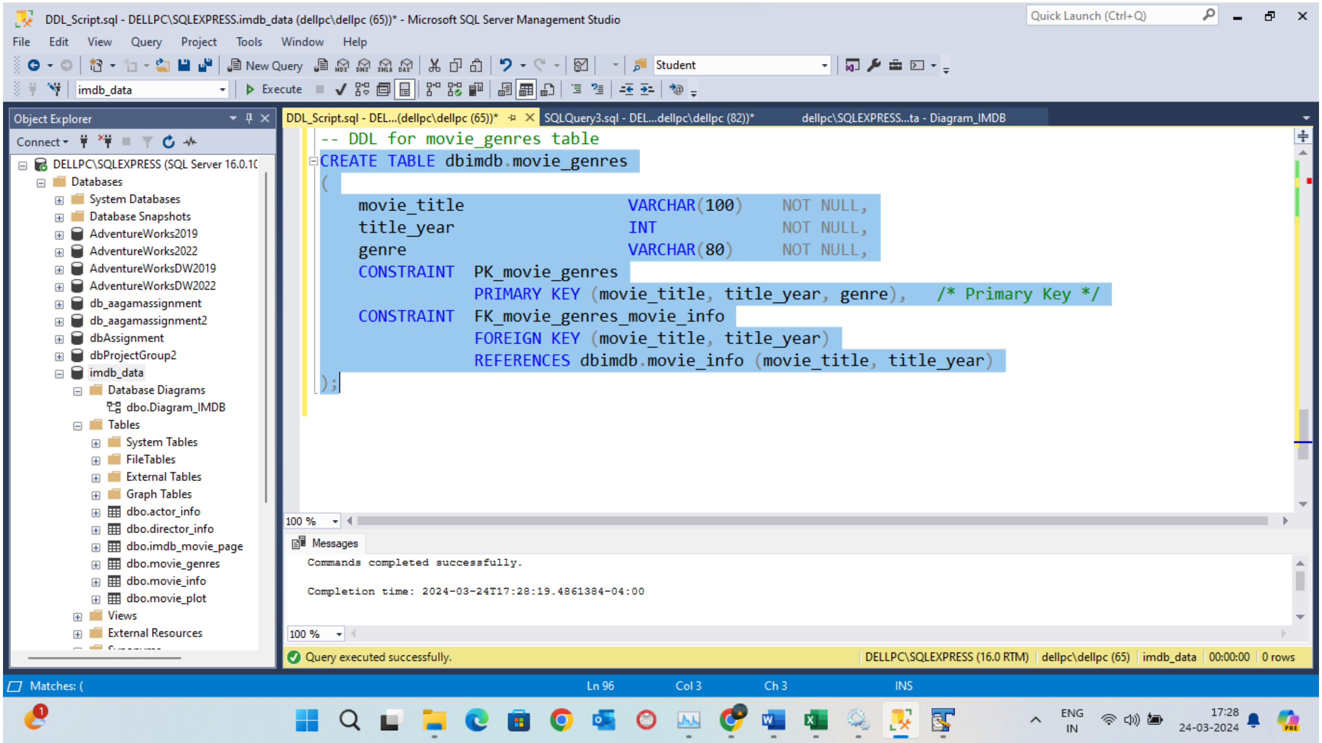


DDL_Script.sql - DELLPC\SQLEXPRESS.imdb_data (dellpc\dellpc (65))* - Microsoft SQL Server Management Studio
File Edit View Query Project Tools Window Help
New Query
Student
imdb_data
Execute
Object Explorer
DDL_Script.sql - DEL...(dellpc\dellpc (65))*
SQLQuery3.sql - DEL...dellpc\dellpc (82))*
dellpc\SQLEXPRESS...ta - Diagram_IMDB
Messages
Query executed successfully.
DELLPC\SQLEXPRESS (16.0 RTM) dellpc\dellpc (65) imdb_data 00:00:00 0 rows
Ln 46 Col 1 Ch 1 INS

```
-- DDL for movie_info table
CREATE TABLE dbimdb.movie_info
(
    movie_title                 VARCHAR(100)   NOT NULL,
    title_year                  INT            NOT NULL,
    color                       VARCHAR(20),
    aspect_ratio                NUMERIC(22, 4),
    duration_in_mins            INT            NOT NULL,
    language                    VARCHAR(20),
    country                     VARCHAR(20)    NOT NULL,
    budget                      NUMERIC(22, 4),
    gross                       NUMERIC(22, 4),
    movie_facebook_likes        INT            NOT NULL,
    cast_total_facebook_likes   INT            NOT NULL,
    movie_imdb_link             VARCHAR(100)   NOT NULL,
    director_name               VARCHAR(40)    NOT NULL,
    actor_1_name                VARCHAR(40)    NOT NULL,
```

```
Commands completed successfully.

Completion time: 2024-03-24T17:27:18.4265478-04:00
```

Screenshot for Create table movie_genres:

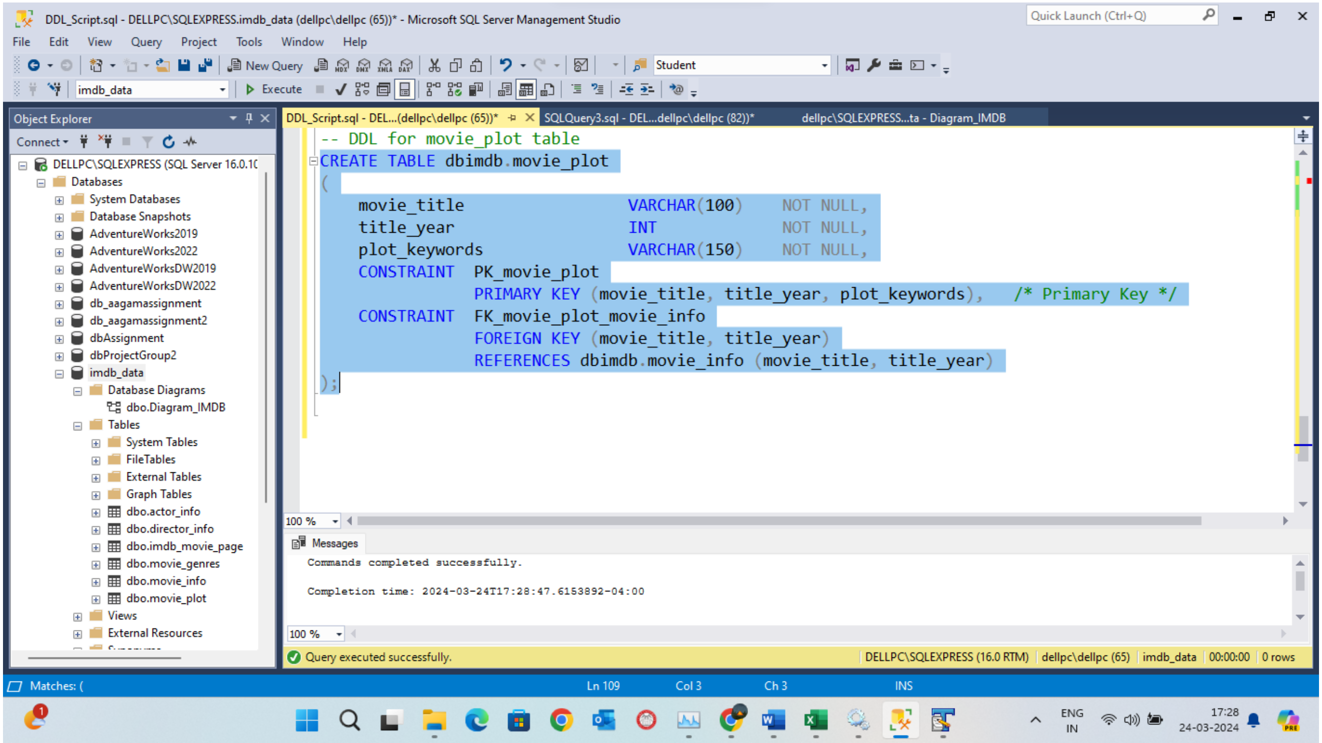

DDL_Script.sql - DELLPC\SQLEXPRESS.imdb_data (dellpc\dellpc (65))* - Microsoft SQL Server Management Studio
Object Explorer
imdb_data
Execute
Messages
DELLPC\SQLEXPRESS (16.0 RTM) | dellpc\dellpc (65) | imdb_data | 00:00:00 | 0 rows

```
-- DDL for movie_genres table
CREATE TABLE dbimdb.movie_genres
(
    movie_title                 VARCHAR(100)    NOT NULL,
    title_year                  INT             NOT NULL,
    genre                       VARCHAR(80)     NOT NULL,
    CONSTRAINT  PK_movie_genres
                PRIMARY KEY (movie_title, title_year, genre),    /* Primary Key */
    CONSTRAINT  FK_movie_genres_movie_info
                FOREIGN KEY (movie_title, title_year)
                REFERENCES dbimdb.movie_info (movie_title, title_year)
);
```

```
Commands completed successfully.

Completion time: 2024-03-24T17:28:19.4861384-04:00
```

Query executed successfully.

Screenshot for Create table movie_plot:


DDL_Script.sql - DELLPC\SQLEXPRESS.imdb_data (dellpc\dellpc (65))* - Microsoft SQL Server Management Studio
Object Explorer
imdb_data
Execute
Messages
DELLPC\SQLEXPRESS (16.0 RTM) | dellpc\dellpc (65) | imdb_data | 00:00:00 | 0 rows

```
-- DDL for movie_plot table
CREATE TABLE dbimdb.movie_plot
(
    movie_title                 VARCHAR(100)    NOT NULL,
    title_year                  INT             NOT NULL,
    plot_keywords               VARCHAR(150)    NOT NULL,
    CONSTRAINT  PK_movie_plot
                PRIMARY KEY (movie_title, title_year, plot_keywords),    /* Primary Key */
    CONSTRAINT  FK_movie_plot_movie_info
                FOREIGN KEY (movie_title, title_year)
                REFERENCES dbimdb.movie_info (movie_title, title_year)
);
```

```
Commands completed successfully.

Completion time: 2024-03-24T17:28:47.6153892-04:00
```

Query executed successfully.