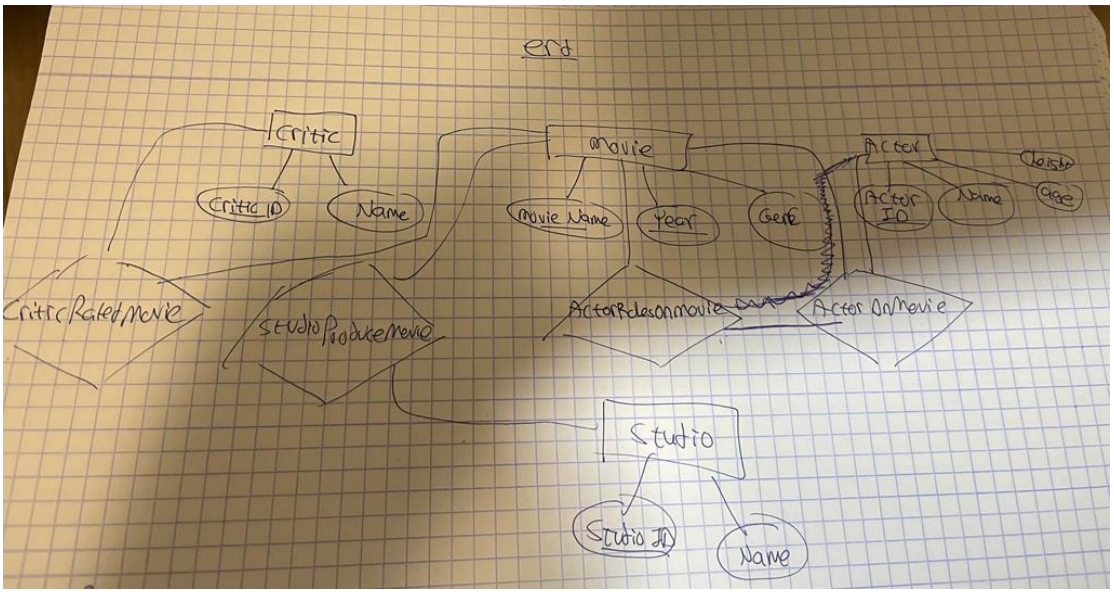
236363 - מסדי נתונים

תרגיל בית 2

תרשים ERD:



במסד הנתונים בחרנו ליצור טבלה עבור כל אובייקט שתואר בתרגיל:
$movie, actor, critic, studio$.
כל טבלה מתארת את שדות האובייקט הנתונים.

בנוסף, בחרנו להוסיף 4 טבלאות נוספות עבור תיאור היחסים בין האובייקטים:

1. $\underline{ActorOnMovie}$ – מקשרת בין סרטים ושחקנים. הגדרנו את ה-$Primary\ key$ בטבלה להיות movie name, year,actorID . היחס מתאר קשר של $many\ to\ many$ בין הישויות $movie$ ו-$actor$ כי סרט יכול להכיל מספר שחקנים ושחקן יכול להימצא במספר סרטים.
2. $\underline{ActorRolesOnMovie}$ – מקשרת בין סרטים ותפקידי השחקנים בהם. היחס מתאר קשר של $many\ to\ many$ בין הישויות $movie$ ו-$ActorOnMovie$ כי סרט יכול להכיל מספר תפקידים עבור שחקן לסרט ותפקידים של שחקן יכולים יכול להימצא במספר סרטים. הגדרנו את (MovieName,Year,ActorId) להיות

foreign key, כך שמחיקה של אחד האובייקטים תוביל להסרתו גם מטבלה זו.

3. StudioProduceMovie- מקשרת בין סטודיו והסרטים שהופקו בו. הגדרנו את ה-$Primary\ key$ בטבלה להיות movie name, year,studioID . היחס מתאר קשר של $one\ to\ many$ בין הישויות $studio$ ו-$movie$ כי סטודיו יכול להכיל מספר סרטים, בעוד כל סרט יכול להיות מופק על ידי סטודיו בודד או ללא סטודיו בכלל.

4. CriticRatedMovie - מקשרת בין סרט והביקורות שלו. הגדרנו את ה-$Primary\ key$ בטבלה להיות movie name, year,criticID . היחס מתאר קשר של $many\ to\ many$ בין הישויות $critic$ ו-$movie$ כי סרט יכול להכיל מספר ביקורות ומבקר יכול לבקר מספר סרטים שונים.

בהגדרת הטבלאות אכפנו את האילוצים על הערכים באמצעות CHECK וכדי להבטיח שהערכים עבור actor, movie, critic ו- $studio$ לא יהיו אופציונליים השתמשנו ב- $NOT\ NULL$. בטבלאות היחסים הגדרנו את המפתחות הזרים שלהם עם $ON\ DELETE\ CASCADE$ כדי להבטיח שאם למשל נסיר סרט מטבלת הסרטים אז כל הרשומות שלו בטבלאות היחסים יוסרו גם כן.

בנוסף, הגדרנו עבור המסד שלנו $VIEWS$ בהם עשינו שימוש מספר פעמים לאורך התרגיל וחסכו לנו ביצוע שאילתות מקוננות נוספות.

- MovieExists- מכיל רק את השדות MovieName, Year הרלוונטים כדי לדעת האם סרט כלשהו קיים, ללא שמירה על מידע נוסף בו לרוב לא נעשה שימוש בשאילתות השונות.
- ActorPlayInMovie- מכיל רק את השדות ActorId, MovieName, Year הרלוונטים כדי לדעת האם שחקן כלשהו שיחק בסרט מסוים, ללא שמירה על מידע נוסף בו לרוב לא נעשה שימוש בשאילתות השונות.
- IsStudioProducedMovie- מכיל רק את השדות StudioId, MovieName, Year הרלוונטים כדי לדעת האם סטודיו הפיק סרט מסוים, ללא שמירה על מידע נוסף בו לרוב לא נעשה שימוש בשאילתות השונות.
- MovieBudget- מכיל רק את השדות Budget, MovieName, Year בהם נעשה שימוש מספר פעמים בשאילתות שונות לאורך התרגיל.
- CriticFanOfMovie- מכיל רק את השדות CriticId, MovieName, Year שהם השדות הדרושים כדי לדעת האם מבקר ביקר סרט מסויים, ללא שמירה על מידע נוסף בו לרוב לא נעשה שימוש בשאילתות השונות.

פונקציות:

CRUD API:

addCritic – השתמשנו בפעולה INSERT על מנת להכניס את הביקורת שניתן לנו לטבלה Critic .
במקרה של הצלחה הוחזר OK במקרה שיש שגיאה בפרמטרים החזרנו BAD_PARAMS במקרה שכבר קיימת ביקורת עם הID הזה החזרנו ALREADY_EXISTS ו Errorבמקרה של שגיאה אחרת.

getCriticProfile– השתמשנו בפעולה SELECT על מנת לחלץ את המשתנים של critic מהטבלה Critic שמתאימים לcriticID שקיבלנו ולהחזיר אותם. את תוצאת השאילתה הכנסנו למשתנה result והחזרנו אותו. במקרה של כישלון שלחנו Badcritic.

deleteCritic-- השתמשנו בפעולה DELETE על הטבלה Critic על מנת למחוק את הביקורת עם ה id הרלוונטי. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה affected_rows -שונה מאפס החזרנו סטטוס OK .במקרה שביקורת לא קיימת החזרנו NotExist ו Errorבמקרה אחר.

addMovie– השתמשנו בפעולה INSERT על מנת להכניס את הסרט שניתן לנו לטבלה Movie .
במקרה של הצלחה הוחזר OK במקרה שיש שגיאה בפרמטרים החזרנו BAD_PARAMS במקרה שכבר קיים הסרט עם הID הזה החזרנו ALREADY_EXISTS ו Errorבמקרה של שגיאה אחרת.

getMovieProfile– השתמשנו בפעולה SELECT על מנת לחלץ את המשתנים של movie מהטבלה Movie שמתאימים לmovieID שקיבלנו ולהחזיר אותם. את תוצאת השאילתה הכנסנו למשתנה resultוהחזרנו אותו. במקרה של כישלון שלחנו Badmovie.

deleteMovie– השתמשנו בפעולה DELETE על הטבלהMovie על מנת למחוק את הסרט עם ה id הרלוונטי. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה affected_rows -שונה מאפס החזרנו סטטוס OK .במקרה שסרט לא קיים החזרנו NotExist ו Errorבמקרה אחר.

addActor– השתמשנו בפעולה INSERT על מנת להכניס את השחקן שניתן לנו לטבלה Actor .
במקרה של הצלחה הוחזר OK במקרה שיש שגיאה בפרמטרים החזרנו BAD_PARAMS במקרה שכבר קיים השחקן עם הID הזה החזרנו ALREADY_EXISTS ו Errorבמקרה של שגיאה אחרת.

getActorProfile– השתמשנו בפעולה SELECT על מנת לחלץ את המשתנים של actor מהטבלה actor שמתאימים לactorID שקיבלנו ולהחזיר אותם. את תוצאת השאילתה הכנסנו למשתנה result והחזרנו אותו. במקרה של כישלון שלחנו Badactor.

deleteActor– השתמשנו בפעולה DELETE על הטבלהActor על מנת למחוק את השחקן עם ה id הרלוונטי. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה affected_rows -שונה מאפס החזרנו סטטוס OK .במקרה ששחקן לא קיים החזרנו NotExist ו Errorבמקרה אחר.

addStudio– השתמשנו בפעולה INSERT על מנת להכניס את הסטודיו שניתן לנו לטבלה Studio .
במקרה של הצלחה הוחזר OK במקרה שיש שגיאה בפרמטרים החזרנו BAD_PARAMS במקרה שכבר קיים הסטודיו עם הID הזה החזרנו ALREADY_EXISTS ו Errorבמקרה של שגיאה אחרת.

getStudioProfile – השתמשנו בפעולה SELECT על מנת לחלץ את המשתנים של studio מהטבלה Studio שמתאימים לstudioID שקיבלנו ולהחזיר אותם. את תוצאת השאילתה הכנסנו למשתנה resultוהחזרנו אותו. במקרה של כישלון שלחנו Badstudio.

deleteStudio – השתמשנו בפעולה DELETE על הטבלהStudio על מנת למחוק את סטודיו עם ה id הרלוונטי. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה affected_rows -שונה מאפס החזרנו סטטוס OK .במקרה שסטודיו לא קיים החזרנו NotExist ו Errorבמקרה אחר.

Basic API

criticRatedMovie – השתמשנו בפעולה INSERT על הטבלהcriticRatedMovie על מנת להוסיף את הביקורת עם ה idהרלוונטי לסרט עם ה yearIname הרלוונטים. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה _ `rows_effected` -שונה מאפס החזרנו סטטוס OK במקרה של כישלון - החזרנו BAD_PARAMS במקרה שהדירוג לא בין 1-5. במקרה שכבר קיימת ביקורת הזו לסרט הזה ALREADY_EXISTS ובמקרה שהסרט הזה או הביקורת לא קיימת Not_Exist .ו Errorבמקרה של שגיאה אחרת.

criticDidntRateMovie- השתמשנו בפעולה DELETE על הטבלהcriticRatedMovie על מנת למחוק את הביקורת עם ה idהרלוונטי לסרט עם ה year Iname הרלוונטים.. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה - affected_rowsשונה מאפס החזרנו סטטוס. OK במקרה שהביקורת או הסרט לא קיים או שלא הייתה ביקורת לסרט הזה החזרנו NotExist ו Errorבמקרה אחר.

actorPlayedInMovie- השתמשנו בפעולה INSERT על מנת להכניס בשאילתה אחת מספר שורות שונות. כך הכנסנו עבור כל תפקיד של שחקן כלשהו שורה חדשה בActorRolesOnMovie המכילה את הסרט, השחקן ואותו תפקיד. בנוסף בעזרת שאילתה נוספת הכנסנו לטבלה ActorOnMovie שורה אחת עבור כל שחקן את הסרט בו שיחק והמשכורת שקיבל.

actorDidntPlayInMovie – השתמשנו בפעולה DELETE על הטבלה`ActorOnMovie` על מנת למחוק את השחקן עם ה idהרלוונטי לסרט עם ה year Iname הרלוונטים.. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה - affected_rowsשונה מאפס החזרנו סטטוס. OK במקרה שהשחקן או הסרט לא קיים או שלא היה שחקן בסרט הזה החזרנו NotExist ו Errorבמקרה אחר.

getActorsRoleInMovie – השתמשנו בפעולה SELECT על מנת לחלץ את הרשימת תפקידים של שחקן מהטבלה `ActorRolesOnMovie` שמתאימים לactorID שקיבלנו וmovie name,year של הסרט את תוצאת השאילתה הכנסנו למשתנה result בתור רשימה והחזרנו אותו. במקרה שלא קיים השחקן או הסרט או השחקן לא שיחק בסרט הזה שלח רשימה ריקה.

studioProducedMovie – השתמשנו בפעולה INSERT על הטבלהstudioProducedMovie על מנת להוסיף את הסרט עם הmovie name,year הרלוונטי לסטודיו עם הid הרלוונטי והrevenue והbudfget . את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה _ `rows_effected` -שונה מאפס החזרנו סטטוס OK במקרה של כישלון - החזרנו BAD_PARAMS במקרה budget or revenue are negative במקרה

שכבר קיים סטודיו לסרט הזה- ALREADY_EXISTS ובמקרה שהסרט הזה או הסטודיו לא קיימים Not_Exist. ו Errorבמקרה של שגיאה אחרת.

studioDidntProduceMovie – השתמשנו בפעולה DELETE על הטבלהstudioProducedMovie על מנת למחוק את הסטודיו עם ה idהרלוונטי לסרט עם movie name ו year הרלוונטים.. את תוצאת השאילתה שמרנו במשתנה`rows_effected` על מנת לבדוק כמה שורות הושפעו מהפעולה. במקרה של הצלחה שבו ה affected_rows -שונה מאפס החזרנו סטטוס. OK במקרה שהסטודיו או הסרט לא קיים או שלא היה סרט בסטודיו הזה החזרנו NotExist ו Errorבמקרה אחר.

averageRating – השתמשנו בפעולה SELECT על מנת להוציא מה`CriticRatedMovie` - את הממוצע של הדירוגים על הסרט עם הmovie name ו-year הרלוונטים .את תוצאת השאילתה הכנסנו למשתנה result ובאמצעותו החזרנו את הממוצע .במקרה של כישלון כללי החזרנו 0.

averageActorRating – בפונקציה זו השתמשנו בשאילתות מקוננות. בשאילתה הראשונה השתמשנו בפעולה SELECT על מנת לבחור את כלל הסרטים שבהם השחקן עם הactorID שקיבלנו מהטבלה `ActorPlayInMovie` ובשאילתה הראשית השתמשנו בפעולה SELECT על `CriticRatedMovie` על מנת להחזיר את הממוצע של הדירוגים עבור כל סרט שנמצא בטבלה שקיבלנו מהשאילתה הראשונה .את תוצאת השאילתה הכנסנו למשתנה result ובאמצעותו החזרנו את הממוצע של כל הסרטים בהם השחקן השתתף. במקרה והשחקן לא שיחק באף סרט או אם אין דירוג עבור הסרטים או אם השחקן לא קיים החזרנו 0.

bestPerformance – בשלב הראשון בצענו LEFT JOIN בין טבלה המכילה את כל הסרטים שהשחקן שקיבלנו כקלט משחק בהם (בנינו טבלה זו על ידי שאילתה מקוננת) לבין CriticRatedMovie כדי לקבל טבלה שמכילה עבור כל סרט בו הוא שיחק את הדירוגים השונים שהמבקרים נתנו לו, ביצענו LEFT JOIN כדי לקבל NULL עבור סרטים שלא דורגו, אותו החלפנו ל-0 בעזרת COALESCE. בשלב הבא בעזרת GROUP BY על תוצאת השלב הקודם לפי MovieName, Year חישבנו לכל סרט של השחקן את הדירוג הממוצע שלו, דאגנו למיין את התוצאות כפי שנדרש בתרגיל ולבחור רק את התוצאה הראשונה לאחר המיון. בשלב האחרון בחרנו מהטבלה Movies המכילה את כל המידע על סרט מסוים את אותו הסרט שהחזרנו בשלב הקודם, כדי לקבל את כל הפרטים עליו על מנת להחזיר אובייקט כפי שנדרש בתרגיל.

- stageCrewBudget – בשאילתה הראשונה השתמשנו בפעולה SELECT על מנת לסכום את כל הsalary ששחקנים שהשתתפו בסרט עם ה movie nameוyear קיבלו מהטבלה `ActorOnMovie`. בשאילתה השנייה השתמשנו בפעולת SELECT על מנת לקבל מהview את התקציב של הסרט עם הmovie name והyear .ובשאילתה השלישית פעולת SELECT על `MovieExists` על מנת לקבל את הסרט עם הmovie name והyear .
לאחר מכן השתמשנו בשאילתה $LEFT\ JOIN$ בין הטבלאות `buildAllMovies`אל `buildBudget` תחת התנאי שחזרנו 0 אם התקציב שווה לnull. כמו כן השתמשנו בשאילתהSelect כדי לחסר בין התקציב לסכום המשכורות מתוך ה- $LEFT\ JOIN$ בין הטבלאות `joinMovieBudget` אל `buildSalaryAmount` .
במידה והסרט לא קיים החזרנו -1.סרטים שלא הופקו ע"י אף אחד מהסטודיו החזרנו 0.

- overlyInvestedInMovie- בשאילתה הראשונה השתמשנו בפעולה SELECT על מנת לסכום את כמות התפקידים שהיה לשחקן עם הactorID בסרט הזה מהטבלה `ActorRolesOnMovie` .
בשאילתה השנייה השתמשנו בפעולת SELECT על מנת לסכום את כמות התפקידים שהיו בסך הכל בסרט עם ה movie name ו year .

לאחר מכן השתמשנו בשאילתה $INNER\ JOIN$ בין הטבלאות buildEachMovieRolesNumber אל buildActorRolesNumber תחת התנאי שחזרנו 0 אם התקציב שווה לnull. כמו כן השתמשנו בשאילתהSelect כדי לחסר בין התקציב לסכום המשכורות מתוך ה- $LEFT\ \ JOIN$ בין הטבלאות joinMovieBudget אל buildSalaryAmount .ומחזיר true אם לשחקן יש מספר תפקידים כפול 2 ממספר התפקידים שיש בסרט.מחזיר false אחרת אואם לא קיים כזה שחקן או סרט או השחקן לא שיחק בסרט.

Advanced API

- franchiseRevenue- עשינו RIGHT OUTER JOIN בין טבלה המכילה את MovieName, Year, Revenue (אותה בנינו בעזרת שאילתה מקוננת) לבין טבלה עם כלל הסרטים, זאת כדי לתפוס גם סרטים שלא הופקו על ידי סטודיו מסויים. על תוצאת ה JOIN עשינו GROUP BY לפי MovieName כדי לסכום את הרווח של כל סרט ודאגנו להחשיב ערך NULL כ-0 בחישוב ולבסוף למיין את התוצאות כפי שנדרש בתרגיל.
- studioRevenueByYear – באמצעות GROUP BY של StudioId, Year על הטבלה שמכילה את הסרטים השונים שהופקו לכל סטודיו, הצלחנו לחשב את סך כל הרווח שכל סטודיו הפיק בשנה. בשלב הבא מיינו את התוצאות לפי הגדרת התרגיל.
- getFanCritics- בשלב הראשון חישבנו בעזרת GROUP BY של StudioId על IsStudioProducedMovie את מספר הסרטים שכל סטודיו הפיק באופן כללי. בשלב הבא באמצעות ביצוע INNER JOIN בין טבלה המכילה את מספר הסרטים שכל מבקר דירג עבור סטודיו מסויים עם IsStudioProducedMovie וביצוע GROUP BY של CriticId, StudioId על התוצאה קיבלנו טבלה שמכילה עבור כל מבקר את מספר הסרטים שהוא ביקר עבוד סטודיו מסויים. בשלב האחרון בצענו INNER JOIN בין תוצאות שני השלבים (אותם ביצענו כשאילתות מקוננות) וקיבלנו טבלה שמכילה עבור כל מבקר את מספר הסרטים שביקר עבור סטודיו מסוים וכמה סרטים אותו סטודיו הפיק, עשינו סינון כך שיוחזרו צירופים של CriticId, StudioId רק עבור שורות בהן המבקר ביקר את כל הסרטים שהופקו בסטודיו והוא ביקר לפחות סרט אחד כזה. את התוצאה מיינו כפי שנדרש בתרגיל.
- averageAgeByGenre- ביצענו INNER JOIN בין טבלה המכילה לכל שחקן את הגיל שלו לבין ActorPlayInMovie וכך קיבלנו טבלה שבה לכל שחקן יש גיל ואת הסרטים שבו שיחק. בשלב הבא עשינו INNER JOIN בין התוצאה לבין טבלה המכילה את כל הסרטים יחד עם הז'אנר שלהם ולבסוף בעזרת GROUP BY על התוצאה לפי ז'אנר חישבנו את הממוצע על הגילאים לכל ז'אנר ומיינו לפי דרישות התרגיל.
- getExclusiveActors- בשלב הראשון ביצענו LEFT JOIN בין ActorPlayInMovie ל IsStudioProducedMovie כדי לקבל טבלה בה לכל שחקן יש שורה עבור כל סטודיו שהוא שיחק בסרט שאותו סטודיו הפיק, עשינו LEFT JOIN כדי לקבל NULL עבור סרטים שלא הופקו על ידי אף סטודיו. בשלב הבא בעזרת GROUP BY על תוצאת השלב הקודם לפי ActorID הצלחנו לבחור רק שחקנים ששיחקו בסרטים שהופקו על ידי סטודיו אחד בדיוק (שחקנים ששיחקו בסרטי אינדי סוננו בשלב זה). לבסוף החזרנו שורות של ActorID, StudioID מתוצאת השלב הראשון רק עבור שחקנים שנמצאים בתוצאת השלב השני ומיינו כפי שנדרש בתרגיל.

Database Functions

הפונקציה $createTables$ יוצרת את הטבלאות וה-$VIEWS$ של מסד הנתונים, תוך שימוש בשאילתה$CREATE\ TABLE$, , $CREATE\ VIEW$.

הפונקציה $clearTables$ מנקה את תוכן הטבלאות באמצעות שימוש בשאילתה $DELETE$.

הפונקציה $dropTables$ מוחקת את הטבלאות באמצעות $DROP\ TABLE$, $DROP\ VIEW$.