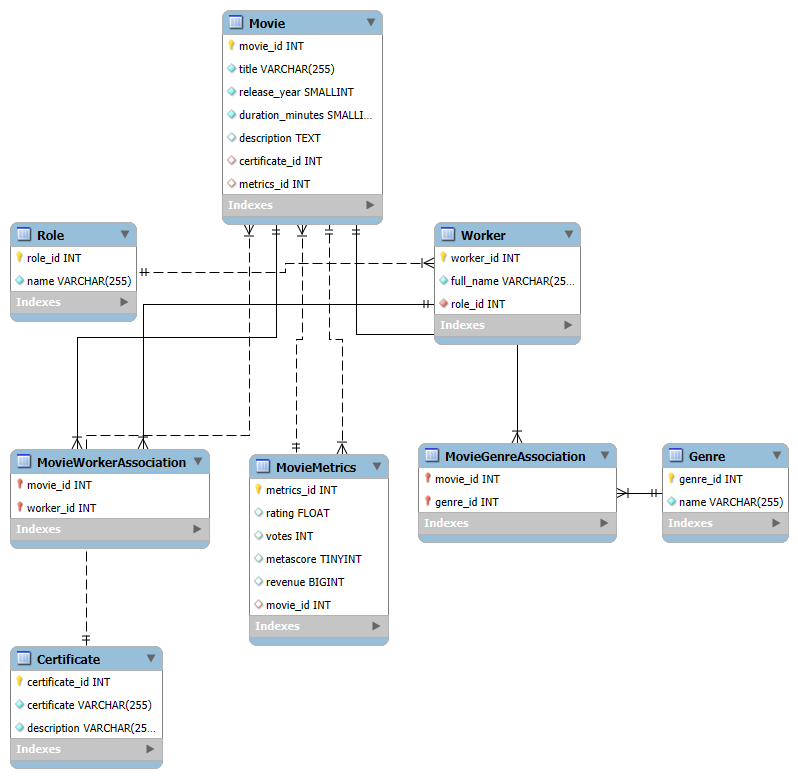
System Documentation

תיאור ה- Data Scheme Structure

* תיאור גרפי
* 

**פירוט**

מדובר על Database של סרטים, ולכן ה- Movie Table מהווה את הטבלה המרכזית

(כפי שמעידים מספר הקשרים לתוכה ומיקומה באיור).

לסרט יש:

* רשומה ב- Movie Table שרלוונטית עבורו
* מזהים של העובדים שהשתתפו בו ניתנים להשגה דרך MovieWorkerAssociation

ומשם דרך Worker Table לפרטיהם המדויקים ואף דרך Role Table לתפקידם

* רשומה ב- Certificate שרלוונטית עבורו
* רשומות Genre שרלוונטיות עבור דרך MovieGenreAssociation ומשם דרך Genre לפרטיהם המדויקים

הסבר לבחירת העיצוב הנוכחי של הטבלאות

בנינו לפי העקרונות שנלמדו:

* התחשבות בקשרים מובנים: one-to-one, one-to-many עם forgien key לחיסכון,  
  many-to-many בטבלה (בשביל Normalization)
* התחשבות ב- Normalization
* שמות משמעותיים
* אינדקסים בשביל השאילתות

**על הבחירה ב- MovieMetrics**  
בחרנו ליצור MovieMetrics ולא לשים את המידע הזה גם בטבלה של Movies כי מדובר על מטריקות יותר מספריות וסטטיסטיות פחות מהותיות כחלק מהסרט עצמו.  
  
חסרון קיים: עוד טבלה שצריך לתחזק, ולהתחשב בה ב- Queries.  
יתרון קיים: החלוקה לעוד טבלה עושה סדר לוגי, ההפרדה בין מטריקות למידע החיוני.  
  
בצורה דומה גם על הבחירה בטבלה Certificate  
(אם כי בה הייתה פחות התלבטות, וזה ממילא היה יותר מתבקש להפריד)

בנוסף, נציין תיקון שעשינו והוא שיש one-to-one בין metrics ל- movie ולכן בחרנו לשים foreign key בשניהם במקום טבלה נוספת MovieMetricsAssociation,  
מה שמתאים אל many-to-many.   
  
  
**על הבחירה ב- Worker וכן Role**בחרנו לא לעשות את ה- Director וה- Actors בירושה מה- Worker.  
אלא לעשות טבלה נוספת של Role שאומרת על התפקיד.  
  
חסרון קיים: אם היו מאפיינים מפרידים של Director ו- Actor (מעבר להיותם מה שהם),  
לעשות את זה בירושה היה נוח מאוד.  
  
יתרון קיים: המצב האמיתי של ה- Database הוא שאין מאפיינים מפרידים.  
ולא ייאסף עוד מידע (בהתאם ל- user manual).  
לכן העיצוב של טבלת Role ולא טבלאות בירושה מ- Worker עם Foreign Key ובלי שדות נוספים, הוא פחות בזבזני בטבלאות ויותר הגיוני לנו עיצובית.

**על הבחירה ב- Association Tables**הן משקפות יחסים של many-to-many ואלטרנטיבות אחרות לא היו עומדות ב- Normalization.  
כאשר היה אפשר לעשות את היחס דרך forgien key בטבלה  
(one-to-one, one-to-many) כך נעשה ממקום של חסכון בטבלאות.  
  
  
**אופטימיזציות של ה- Database**

אופטימיזציה של Queries

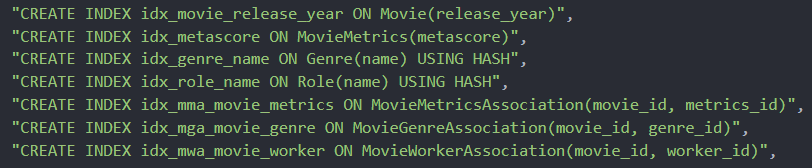
* ביצוע של Select רק על העמודות הרלוונטיות
* הימנעות מטבלאות זמניות שלא לצורך קריאות
* שכתוב שאילתות לצורה יעילה יותר

אופטימיזציה של Scheme

* שימוש ב- Data Types קטנים ככל האפשר, שימוש ב- UNSIGNED כשניתן
* עקרונות Normalization שעוזרות ליעילות
* הוספת בדיקות CHECK בתוך ה- Scheme כדי להימנע מבדיקות כאלה  
  ב- Queries שכתבנו (שאינן DML)
* **שימוש באינדקסים** (מעבר ל- defaults, גם custom ones)

**שימוש באינדקסים Custom**  
הוספנו אינדקסים מותאמים אישית לשאילתות שכתבנו.

* idx\_movie\_release\_year הוא B-Tree כי release\_year משמש ל- Order By וגם GROUP BY ב- query\_1, וכן WHERE לפי =< ב- query\_2 וגם ORDER BY בה.
* idx\_metascore הוא B-Tree כי metascore משמש ל- Order By ב- query\_4  
  וכן ל- ORDER BY ב- query\_3
* idx\_genre\_name הוא Hash כי name של Genre מושווה בתנאי WHERE של query\_2 (כדי להציג רק את ה- genre שבו המשתמש בחר)
* idx\_role\_name הוא Hash כי name של Role מושווה בתנאי WHERE של query\_3 (להבדיל ‘director’ לעומת ‘actor’)



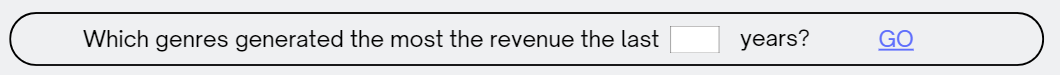
אינדקס Full-Text עבור query\_4 וכן query\_5



מעבר ל- custom indices:

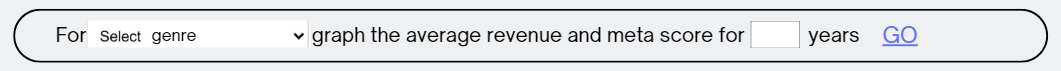
* אינדקסים של Primary Key נוצרים ב- default.
* בנוסף, עשינו 2 טבלאות של Association Tables בשביל many-to-many  
  ועשינו 2 Composite Indices כי לכל טבלה יצרנו Primary Key משתי ה- attributes  
  (על מנת לייעל את השאילתות JOIN עם הטבלה)

**פירוט על השאילתות**

Query 1:  
המטרה: לאפשר למשתמש לראות בצורה נוחה את הז'אנרים הכי מוצלחים  
מבחינת הכנסה כלכלית, בכל אחת מהשנים לאחור שבחר.  
  
מחזיר: לכל שנה לפי מה שבחר המשתמש מחזיר את  
השנה, הז'אנר הכי מצליח מבחינה כלכלית, ההכנסה הכלכלית שלו.  
  
נתמך בעיצוב ה- :Database  
הטבלאות שבהם משתמשים הן: Movie, MovieMetrics, Genre.  
הן כוללות את כל המידע הנדרש לשאילתא.  
בנוסף, כדי להשתמש בהם עם JOINs ולאפשר יחס many-to-many  
עשינו 2 טבלאות Association.  
מעבר לזה, עשינו אינדקס על release\_year בתור אופטימיזציה על השאילתא.

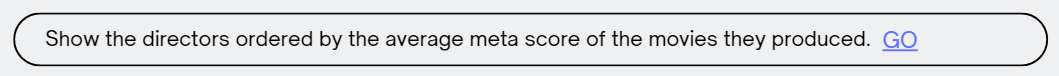
תיאור מילולי כללי של השאילתא:

* מחשבת את ההכנסות הכוללות לכל סרט ומסווגת את ההכנסות לפי שנה וז'אנר
* בוחרת את הז'אנר עם ההכנסה המקסימלית לכל שנה, משאירה רק אותם
* ומוסיפה את השדות הרלוונטיים, ומתקבלת טבלת הפלט

****Query 2:  
המטרה: לאפשר למשתמש לראות בצורה נוחה את ממוצע ההכנסה וה- metascore  
של הז'אנר שבחר, בטווח השנים שבחר.  
  
מחזיר: לכל שנה לפי מה שבחר המשתמש מחזיר את  
השנה, ההכנסה, דירוג metascore.  
  
נתמך בעיצוב ה- :Database  
הטבלאות שבהם משתמשים הן: Movie, MovieMetrics, Genre.  
הן כוללות את כל המידע הנדרש לשאילתא.  
בנוסף, כדי להשתמש בהם עם JOINs ולאפשר יחס many-to-many  
עשינו 2 טבלאות Association.  
מעבר לזה, עשינו אינדקס על release\_year וכן אינדקס על name של Genre  
בתור אופטימיזציות על השאילתא

תיאור מילולי כללי של השאילתא:

* מפלטרת לפי ז'אנר ושנה שהמשתמש בחר את המידע הרלוונטי  
  (שנה, רווחים, דירוג)
* בוחרת את הז'אנר עם ההכנסה המקסימלית לכל שנה, משאירה רק אותם

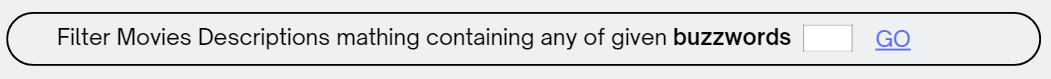
****Query 3:

המטרה: לאפשר למשתמש להבין איזה במאים מוצלחים ולאיזה שחקנים שיתוף פעולה מוצלח עם במאי. מדד ההצלחה הוא ממוצע על ה- metascore.  
  
מחזיר: מחזירה טבלה המדורגת לפי ה- metascore של הבמאים  
לכל במאי, בשורה שלו השחקנים מסודרים לפי ההצלחה של כל אחד מהם עם הבמאי עצמו.  
מדד ההצלחה של שיתוף פעולה של במאי ושחקן הוא ממוצע metascore של הסרטים שעבדו עליהם ביחד.  
  
נתמך בעיצוב ה- :Database  
הטבלאות שבהם משתמשים הן: Movie, MovieMetrics, Worker, Role  
הן כוללות את כל המידע הנדרש לשאילתא.  
בנוסף, כדי להשתמש בהם עם JOINs ולאפשר יחס many-to-many  
עשינו טבלת MovieWorkerAssociation  
מעבר לזה, עשינו אינדקס name של Role בתור אופטימיזציה על השאילתא.

תיאור מילולי כללי של השאילתא:

* מוצאת את שם הבמאי לכל סרט ואת רשימת השחקנים שהיו בסרטים שלו
* לכל במאי, מסדרת אותם לפי ממוצע metascore של הסרטים
* רשימת הבמאים מסודרת לפי metascore של ממוצע הסרטים שלו

נעשה שימוש ב- nested-query בשביל לעשות average metascore לכל שחקן עם במאי

****Query 4:

המטרה: לאפשר למשתמש לראות תיאור של הסרטים  
שכוללים בתיאור שלהם את אחת מה- buzzwords שהוא בחר  
(למשל, כדי לשאוב השראה מסרטים אחרים שעוסקים בנושאים דומים)  
  
מחזיר: לכל סרט שכולל בתיאור שלו אחד ה- buzzwords הוא מחזיר  
שם סרט, תיאור, metascore  
  
נתמך בעיצוב ה- :Database  
הטבלאות שבהם משתמשים הן: Movie, MovieMetrics.  
הן כוללות את כל המידע הנדרש לשאילתא.  
מעבר לזה, עשינו אינדקס Full Text על description של Movie כדי לאפשר חיפוש בפנים הטקסט.  
תיאור מילולי כללי של השאילתא:

* עושה פלטור לפי הסרטים שכוללים בתיאור לפחות buzzword אחד
* עושה JOINs בשביל metascore לכל סרט
* מיון לפי metascore, עד 20 סרטים

****Query 5:

המטרה: לאפשר למשתמש לראות את רשימת הבמאים של הסרטים המוצלחים (מעל הממוצע revenue) שבהם התיאור כולל את מילת ה- buzzword שלו  
(למשל, כדי שהמשתתף יבין עם איזה במאי לעבוד, מבחינת ניסיון שלו בתחום.  
או ממי לשאוב השראה)  
  
מחזיר: לכל סרט שכולל בתיאור שלו אחד ה- buzzwords הוא מחזיר  
שם סרט, במאים, הכנסה, ממוצע הכנסה  
  
נתמך בעיצוב ה- :Database  
הטבלאות שבהם משתמשים הן:   
Movie, MovieMetrics, Worker, MovieWorkerAssociation  
הן כוללות את כל המידע הנדרש לשאילתא.  
בנוסף, כדי להשתמש בהם עם JOINs ולאפשר יחס many-to-many  
עשינו טבלה.MovieWorkerAssociation   
מעבר לזה, עשינו אינדקס Full Text על description של Movie כדי לאפשר חיפוש בפנים הטקסט.  
תיאור מילולי כללי של השאילתא:

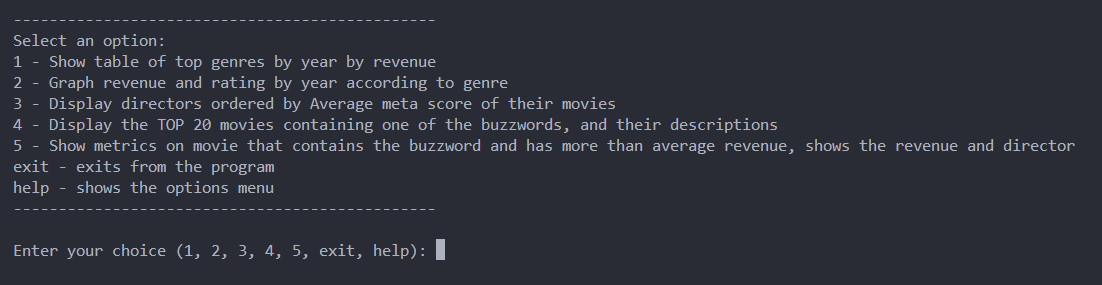
* עושה פלטור לפי הסרטים שכוללים בתיאור את ה- buzzword
* מחשבת את ה- average revenue ומשאירה את שהכניסו מעל הממוצע
* מחזירה ביחד עם שמות הסרטים האלה, את שמות הבמאים, את הכנסותיהם,  
  ואת ההכנסה הממוצעת

דוגמות הרצה

1. ליצירת ה- DB עם Tables ו- indexes לפי ה- schema יש להריץ  
   תוך תיקיית src/ בצורה הבאה: **python create\_db\_script.py**
2. למילוי את ה- Tables במידע מתוך ה- Dataset יש להריץ

מתוך תיקיית src/ בצורה הבאה: **python api\_data\_retrieve.py**

1. על מנת להריץ את ה- UI הטקסטואלי של ה- Queries יש להריץ  
   מתוך תיקיית src/ בצורה הבאה: **python queries\_execution.py**מומלץ לעשות **pip install -r requirements.txt** לפני תחילת ההרצות.

נרחיב הדגמה של ה- queries\_execution.  
  
כך נראה הממשק, לאחר ההרצה:

* בחירה של help מראה שוב את צג זה
* בחירה של exit מסיימת את ריצת הסקריפט
* בחירה באפשרויות שתואמות כל query, מודגמות להלן

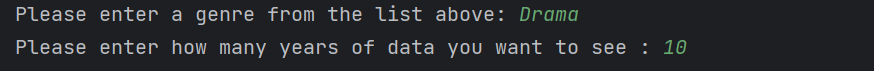
A screenshot of a computer

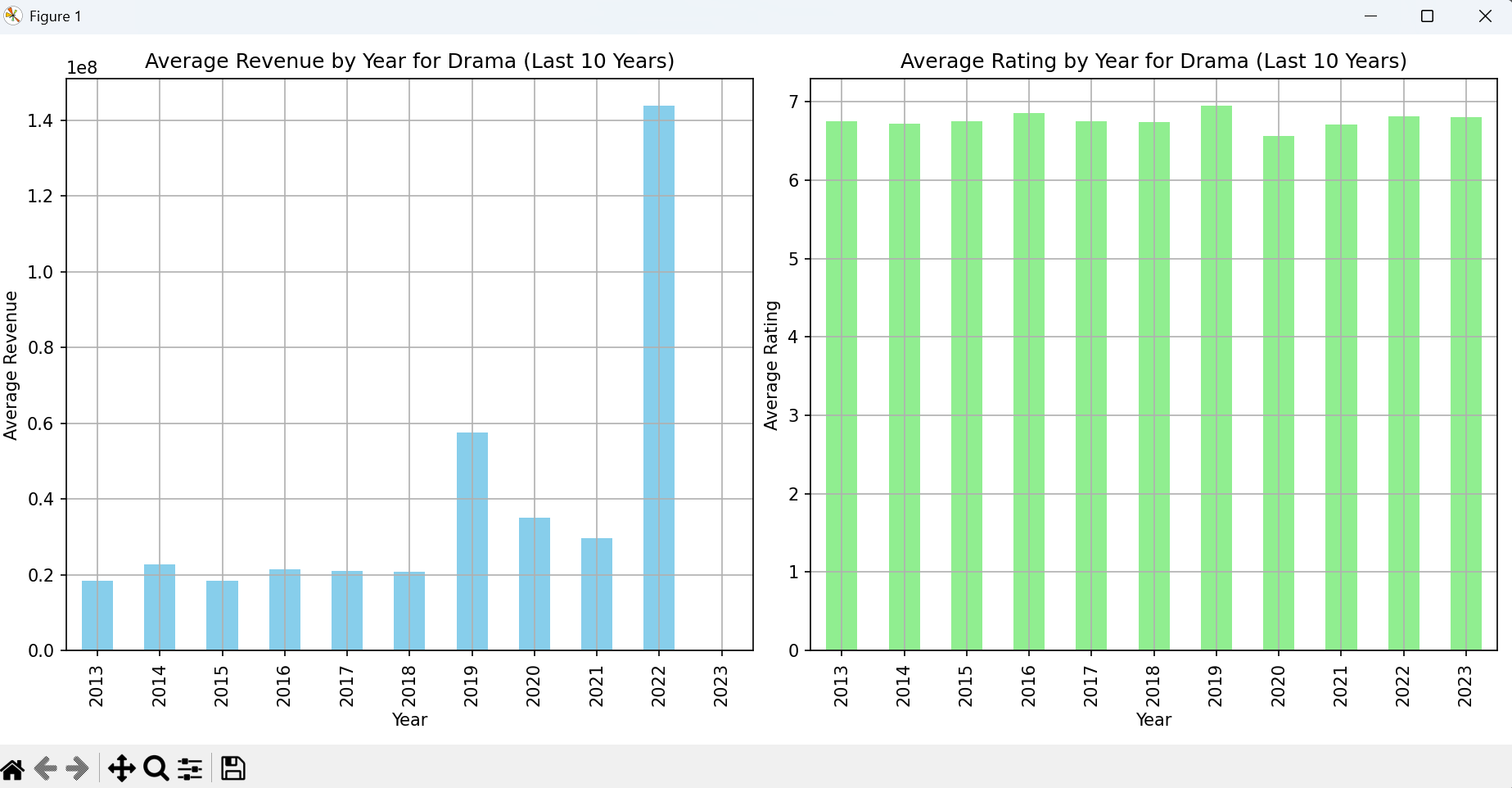
Description automatically generatedדוגמה להרצה של **query1** – Show table of top genres by year by revenue

דוגמה להרצה של **query2** – Graph revenue and rating by year according to genre

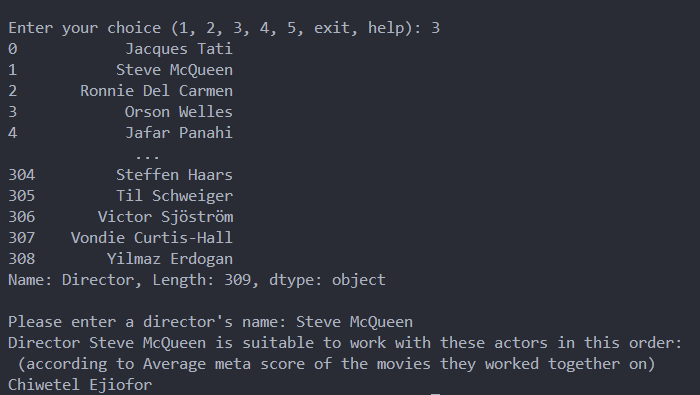
A screenshot of a computer

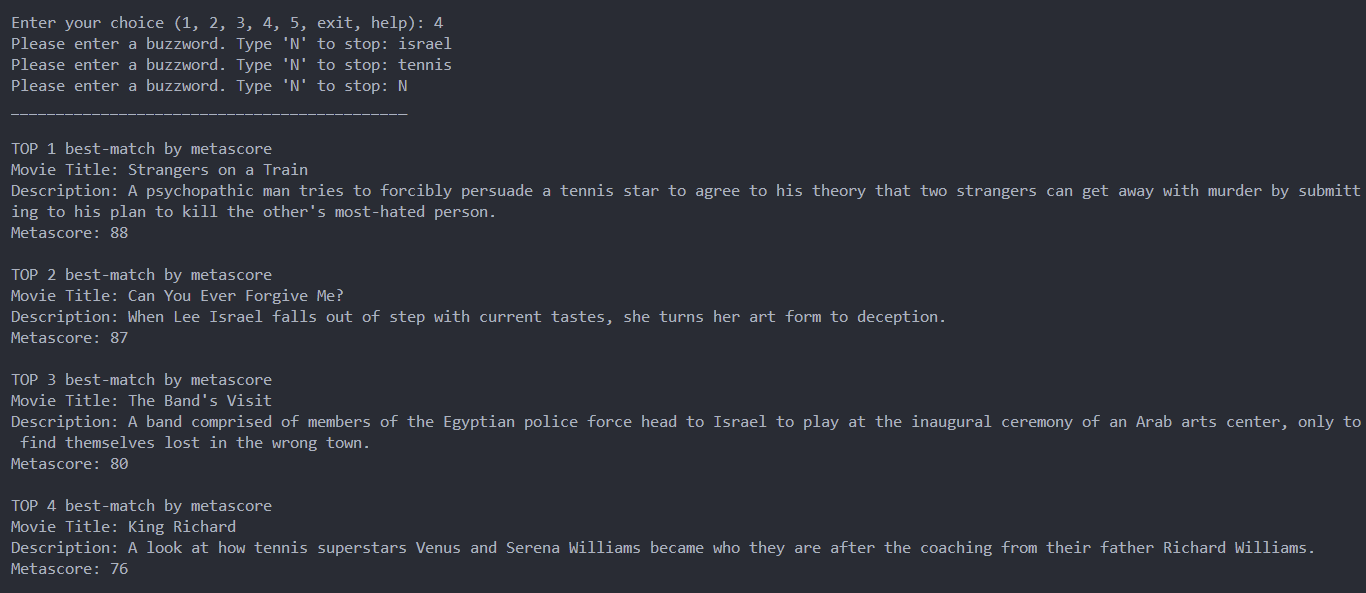
Description automatically generated

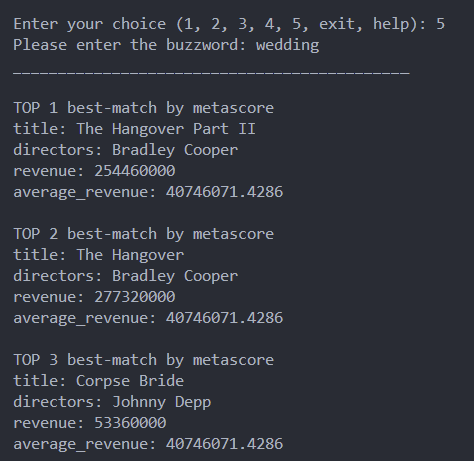




דוגמה להרצה של **query3** –   
Display directors ordered by Average meta score of their moves



דוגמה להרצה של **query4** -   
Display the TOP 20 movies containing one of the buzzwords, and their descriptions

דוגמה להרצה של **query5** -   
Show metrics on movie that contains the buzzword and has more than average revenue, shows the revenue and director