תרגיל 4

תאריך הגשה : 55 :23, 03.07.24.

הוראות הגשה:
בתרגיל זה אתם נדרשים להגיש קובץ zip בודד שיכלול את הקבצים הבאים :

- **ex4.pdf** עם התשובות לשאלות. בכל הסעיפים יש לכתוב הסבר לדרך הפתרון, ולהדגיש את התוצאה הסופית של כל חישוב!
- **improved.sql** עבור התשובה לשאלה 4 סעיף א חלק 1.
- **README** שמכיל שורה בודדת ובו ה-login של הסטודנט שמגיש את התרגיל. אם התרגיל מוגש בזוגות, על שורה זאת להכיל את שני ה-login מופרדים בפסיק.

תזכורת: יש להגיש תרגיל מוקלד בלבד.

בתרגיל זה נשתמש באותה סכמה של טבלאות מתרגיל 2 ובהנחות להלן : (בכל חלק א ובחלק ב שאלה 1)

Movies (movieId, title, rating, year, duration, genre)

Actors (actorId, name, byear)

PlaysIn (movieId, actorId, character)

הנחות :

- גודל בלוק הוא 1,000 בייטים.
- השדות הנומריים : movieId, rating, year, duration, actorId, byear, תופסים כל אחד 8 בייט.
- השדות הטקסטואליים : title, genre, name, character תופסים כל אחד 10 בייט.
- מצביע תופס 8 בייט.
- הערכים בduration בטבלה Movies מתפלגים אחיד בטווח [1,200]
- הערכים בgenre בטבלה מחולקים ל4 קטגוריות באופן אחיד.
- בטבלה Movies יש 10,000 שורות.
- בטבלה Actors יש 50,000 שורות.
- בטבלה PlaysIn יש 100,000 שורות.
- גודל החוצץ (buffer) הוא 52 בלוקים.

חלק א -אינדקסים

א. נתונה השאילתה הבאה :

```
SELECT avg (duration)
FROM Movies
WHERE duration > 100
```

1. מה עלות חישוב השאילתה בהנחה שאין אינדקסים על הטבלה?

כל שורה בטבלה תופסת $4 * 8 + 2 * 10 = 52$ בייט.

בכל בלוק נכנסות $\lfloor 1{,}000 / 52 \rfloor = 19$ שורות.

הטבלה תופסת $\lceil 10{,}000/19 \rceil = 527$ בלוקים.

עלות קריאת הטבלה כולה הוא I/O 527.

כעת, נתון האינדקס הבא על הטבלה:

```
CREATE index on movies(duration)
```

2. מה תהיה דרגת הפיצול האופטימלית של האינדקס?

כל ערך באינדקס תופס 8 בייט, וכל מצביע תופס גם 8 בייט, גודל בלוק הוא 1000 בייט.

לפי הנוסחה: $d \leq \frac{b+v}{v+p} = \frac{1000+8}{8+8} \rightarrow d = 63$

3. מה תהיה עלות חישוב השאילתה באמצעות האינדקס, בהנחה שדרגת הפיצול היא זו שחושבה בסעיף הקודם?

INDEX RANGE SCAN

גובה העץ $h = \left\lceil log_{\lceil \frac{d}{2} \rceil}(T(Movies)) \right\rceil = \left\lceil log_{\lceil \frac{63}{2} \rceil}(10{,}000) \right\rceil = 3$.

כמה ערכים עם duration > 100: $\frac{200-100}{200} \times 10{,}000 = 5{,}000$

בכל עלה נכנסים לכל הפחות $\left\lceil \frac{63}{2} \right\rceil - 1 = 31$ ערכים, וסהכ צריך לעבור על $\left\lceil \frac{5{,}000}{31} \right\rceil = 162$ עלים.

סה"כ עלות חישוב עם אינקדס $\underline{3 + 162 = \mathbf{165}}$

ב. נתונה השאילתה הבאה:

```
SELECT avg(duration)
FROM Movies
WHERE genre = 'Drama'
```

ונתון האינדקס הבא על הטבלה:

```
create index on movies(genre)
```

1. מה תהיה דרגת הפיצול האופטימלית של האינדקס?

כל ערך באינדקס תופס 10 בייט, וכל מצביע תופס גם 8 בייט, גודל בלוק הוא 1000 בייט.

לפי הנוסחה: $d \leq \frac{b+v}{v+p} = \frac{1000+10}{10+8} \rightarrow d = 56$

2. מה תהיה עלות חישוב השאילתה באמצעות האינדקס, בהנחה שדרגת הפיצול היא זו שחושבה בסעיף הקודם?

INDEX RANGE SCAN+TABLE ACCESS BY ROWID

גובה העץ $h=\left\lceil log_{\left\lceil \frac{56}{2} \right\rceil}(10{,}000) \right\rceil = 3$.

מספר הערכים שמתאימים ל genre = 'Drama' הוא $\frac{10{,}000}{4} = 2500$

בכל עלה נכנסים לכל הפחות $\left\lceil \frac{56}{2} \right\rceil - 1 = 27$ ערכים, וסהכ צריך לעבור על $\left\lceil \frac{2{,}500}{27} \right\rceil = 93$ עלים.

צריך לקרוא מהטבלה 2500 ערכים, שנמצאים בלכל היותר 527 בלוקים, כמספר הבלוקים בטבלה

סה"כ עלות חישוב עם אינקדס $3 + 93 + 527 = \mathbf{623}$

ג. נתונה אותה שאילתה כמו בסעיף ב, אבל כעת, נתון האינדקס הבא על הטבלה:

```
create index on movies(genre, duration)
```

1. מה תהיה דרגת הפיצול האופטימלית של האינדקס?

כל ערך באינדקס תופס $10 + 8 = 18$ בייט, וכל מצביע תופס גם 8 בייט, גודל בלוק הוא 1000 בייט.

לפי הנוסחה: $d \leq \frac{b+v}{v+p} = \frac{1000+18}{18+8} \rightarrow d = 39$

2. מה תהיה עלות חישוב השאילתה באמצעות האינדקס, בהנחה שדרגת הפיצול היא זו שחושבה בסעיף הקודם?

INDEX RANGE SCAN

גובה העץ $\left\lceil log_{\left\lceil \frac{39}{2} \right\rceil}(10{,}000) \right\rceil = 4$ =h.

מספר הערכים שמתאימים ל genre = 'Drama' הוא $\frac{10{,}000}{4} = 2500$

בכל עלה נכנסים לכל הפחות $\left\lceil \frac{39}{2} \right\rceil - 1 = 19$ ערכים, וסהכ צריך לעבור על $\left\lceil \frac{2{,}500}{19} \right\rceil = 132$ עלים.

לא צריך לקרוא מהטבלה

סה"כ עלות חישוב עם אינקדס $\underline{4 + 132 = \mathbf{136}}$

חלק ב – Query Plans

שאלה 1 :

בשאלה זו נשתמש שוב בסכמת היחסים ובהנחות המופיעות במלבן הכחול בתחילת התרגיל.

נרצה לחשב עלות של צירוף (join) של הטבלאות $Movies \bowtie PlaysIn$.

א. מה תהיה עלות החישוב של הביטוי בעזרת אלגוריתם *Block-nested-loops*?

כל שורה של הטבלה movies תופסת 52 בייט.
בכל בלוק נכנסות $\lfloor 1000/52 \rfloor = 19$ שורות.
הטבלה תופסת סה"כ $B(Movies) = \left\lceil \frac{10,000}{20} \right\rceil = 527$ בלוקים.

כל שורה של הטבלה PlaysIn תופסת 26 בייט.
בכל בלוק נכנסות $\lfloor 1000/26 \rfloor = 38$ שורות.
הטבלה תופסת סה"כ $B(PlaysIn) = \left\lceil \frac{100,000}{38} \right\rceil = 2632$ בלוקים.

$Movies$ היחס הקטן ולכן נשתמש בנוסחה:

$$B(Movies) + \left\lceil \frac{B(Movie)}{M-2} \right\rceil \times B(PlaysIn) = 527 + \left\lceil \frac{527}{52-2} \right\rceil \times 2632$$

$$\underline{= 29,479\ I/O}$$

ב. מה גודל החוצץ המינימלי הנדרש כדי לקבל את עלות החישוב שקיבלתם בסעיף א? 50

שאלה 2 :

רוצים לחשב את הביטוי $\pi_{A,D}\sigma_{B=20 \wedge D<5}(R(A,B) \bowtie S(A,C,D))$. ההטלה היא ללא מחיקת כפילויות. גודלי היחסים הם B(S)=1,200, B(R)=4,000 . גודל כל אחד מהאטריבוטים הוא 10 bytes וגודל בלוק הוא 2,000 bytes. ליחס R יש אינדקס על האטריבוט B עם עלות גישה זניחה. כמו כן V(S,A)=1000,V(R,B)=200 וידוע שA הוא מפתח ביחס R. בחוצץ (buffer) יש 70 בלוקים.

(הערה: הכוונה ב"עלות גישה זניחה" היא שעלות הגישה לאינדקס - הירידה בו וטיול על העלים - זניחה, ולכן עלות השימוש באינדקס הוא שליפה של בלוקים מהטבלה בלבד. זה מתאים מאד למקרה בו מסד הנתונים שומר את מבנה האינדקס בזיכרון המרכזי)

א. מה יהיה מספר השורות בתוצאה?

כל שורה של R תופסת 20 בייט, בבלוק יש 2000 בייט, ולכן יש 100 שורות בבלוק.

$$T(R) = 4000 \times 100 = 400{,}000,$$

כל שורה של S תופסת 30 בייט, בבלוק יש 2000 בייט, ולכן יש 66 שורות בבלוק.

$$T(S) = 1200 \times 66 = 79{,}200$$

בגלל שA מפתח בR $\mathrm{V(R,A) = T(R) = 400{,}000}$

הנוסחה לגודל התוצאה: $\frac{T(R)\times \mathrm{T}(S)}{\max\{V(R,A),V(S,A)\}\times V(R,B)\times 3} = \frac{400{,}000\times 79{,}200}{400{,}000\times 200\times 3} = \mathbf{132}$

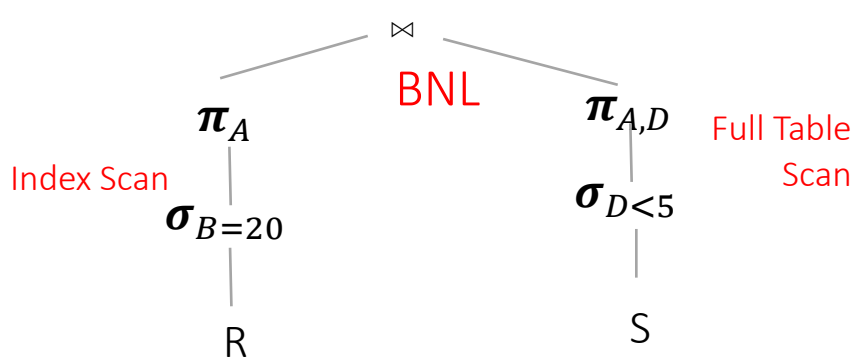
ב. מה יהיה גודל התוצאה בבלוקים?

כל שורה בתוצאה היא בגודל 20 בייט, כי יש הטלה. ולכן יש 100 שורות בבלוק.

$$\left\lceil \frac{132}{100} \right\rceil = \mathbf{2}$$

ג. מהו האלגוריתם הכי יעיל לחישוב התוצאה? **ציירו את עץ הquery plan.**

$$\pi_{A,D}\sigma_{B=20 \wedge D<5}\,(R(A,B) \bowtie S(A,C,D))$$



ד. מה עלות החישוב היעיל ביותר?

נגדיר $E_R = \pi_A \sigma_{B=20} R(A,B), E_S = \pi_{A,D} \sigma_{D<5} S(A,C,D)$

כעת,.

$$T(E_R) = \frac{T(R)}{V(R,B)} = \frac{400{,}000}{200} = 2000\,, B(E_R) = \frac{2000}{200} = 10$$

נשים לב שמספר השורות בבלוק עבור E_R הוא 200 ולא 100 מכיוון שיש הטלה ובתוצאה יש רק עמודה אחת ולא 2 ולכן בחישוב של $B(E_R)$ חילקנו ב-200

$$T(E_S) = \frac{T(S)}{3} = \frac{79{,}200}{3} = 26{,}400\,, B(E_S) = \frac{26{,}400}{100} = 264$$

נשים לב שמספר השורות בבלוק עבור E_S הוא 100 ולא 66 מכיוון שיש הטלה ובתוצאה יש רק שתי עמודות ולא 3 ולכן בחישוב של $B(E_S)$ חילקנו ב-100

נחשב את עלות הקריאה של E_R בעזרת האינדקס:

ראשית נחשב את $T(E_R) = \frac{T(R)}{V(R,B)} = \frac{400{,}000}{200} = 2{,}000$

מכיוון שיש $T(E_R) = 2000$ ועלות גישה של האינדקס זניחה, אז עבור כל שורה בתוצאת הבחירה נקרא לכל היותר בלוק 1 ועלות הקריאה ע"י האינדקס תהיה 2000.

עלות קריאה של הטבלה כולה זה $B(R) = 4000$

ולכן עדיף השימוש באינדקס ויוצא $read(E_R) = 2000$

מכיוון שליחס S אין אינדקס נקבל $read(E_S) = B(S) = 1200$,

נוסחת העלות חישוב 1:

$$read(E_S) + \left\lceil \frac{B(E_S)}{M-2} \right\rceil \times read(E_R) = 1200 + \left\lceil \frac{264}{70-2} \right\rceil \times 2000 = 9{,}200\ I/O$$

נוסחת העלות חישוב 2:

$$read(E_R) + \left\lceil \frac{B(E_R)}{M-2} \right\rceil \times read(E_S) = 2000 + \left\lceil \frac{10}{70-2} \right\rceil \times 1200 = 3{,}200\ I/O$$

שאלה 3

מטרת שאלה זו היא התנסות עם כתיבה יעילה של שאילתות ושימוש באינדקס להתייעלות.

לצורך מענה על הסעיפים הבאים, יש לטעון את הנתונים מהקובץ *moviesBig.csv* הנמצא באתר הקורס לתוך מסד הנתונים במחשב לפי ההוראות הבאות:

1. היכנסו למסד הנתונים (*psql -h dbcourse public*) והשתמשו בפקודה הבאה ליצירת הטבלה:

```
create table movies(
        movieId integer primary key,
        title varchar(150) not null,
        rating numeric check (rating>=0 and rating <=10),
        year integer check (year>0),
        duration integer check (duration>0),
        genre varchar(50)
        );
```

הערה: אם עדיין קיימת הטבלה משימוש בתרגילים קודמים, מומלץ למחוק אותה (ואת שאר הטבלאות) באמצעות הקובץ *drop.sql* וליצור מחדש.

2. צאו ממסד הנתונים, והריצו את הפקודה הבאה:

```
cat Movies-file-path/moviesBig.csv | psql -hdbcourse public -c "copy Movies FROM STDIN DELIMITER ',' CSV HEADER"
```

כאשר *Movies-file-path* הוא שם התיקייה שבה מיקמת את הקובץ *moviesBig.csv*.

נרצה לחשב את השאילתה הבאה:

```
select distinct movieId, title, duration
from Movies M1
where duration = (select min(duration)
                  from Movies M2
                  where M2.year = M1.year)
order by movieId, title, duration;
```

כאשר מריצים את השאילתה, היא רצה יותר מ-2 דקות. (מוזמנים לנסות בעצמכם...)

כאשר הרצנו את השאילתה עם פקודת *explain* (שבשונה מפקודת *explain analyse* לא מריצה את השאילתה, רק מציגה את ה-*query plan*) קיבלנו את הפלט הבא:

```
                                    QUERY PLAN
-------------------------------------------------------------------------------
 Unique  (cost=4206590.88..4206591.53 rows=37 width=480)
   →  Sort  (cost=4206590.88..4206590.98 rows=37 width=480)
         Sort Key: m1.movieid, m1.title, m1.rating, m1.year, m1.duration, m1.genre
         →  Seq Scan on movies m1  (cost=0.00..4206589.92 rows=37 width=480)
               Filter: (duration = (SubPlan 1))
               SubPlan 1
                 →  Aggregate  (cost=561.69..561.70 rows=1 width=4)
                       →  Seq Scan on movies m2  (cost=0.00..561.60 rows=37 width=4)
                             Filter: (year = m1.year)
 JIT:
   Functions: 10
   Options: Inlining true, Optimization true, Expressions true, Deforming true
(12 rows)
```

כעת ענו על השאלות הבאות:

<u>הערה</u>: כדי למדוד זמן ריצה של שאילתה, יש להריץ אותה עם פקודת *explain analyse* וזמן הריצה המבוקש הוא זמן התכנון + זמן הביצוע.

א. נסו לשפר את זמן הריצה ע"י שינוי בתחביר השאילתה.

1. כתבו את השאילתה החדשה בקובץ בשם *improved.sql*.
2. הריצו את השאילתה עם פקודת *explain analyse*, שמראה את ה-*query plan* של השאילתה החדשה, צרפו צילום מסך של התוצאה לתשובות (בדומה לצילום בתחילת השאלה), וכתבו מה זמן הריצה החדש.
3. נסו לשער מה גרם לשיפור בזמן הריצה.
(איך צורך להסביר את כל הפרטים של ה-*query plan* רק מה גרם לשיפור)

.1

```
select movieId, title, duration
from Movies M1
where (year, duration) in (select year, min(duration)
                                   from movies
                                   group by year)
order by movieId, title, duration;
```

.3 זמן ריצה: 128 מילי-שניות

```
                                                    QUERY PLAN
-----------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=2457.99..2461.00 rows=172 width=44) (actual time=128.233..128.453 rows=116 loops=1)
   →  Sort  (cost=2457.99..2458.42 rows=172 width=44) (actual time=128.231..128.297 rows=116 loops=1)
         Sort Key: m1.movieid, m1.title, m1.duration, m1.rating, m1.year, m1.genre
         Sort Method: quicksort  Memory: 36kB
         →  Hash Join  (cost=1221.08..2451.60 rows=172 width=44) (actual time=65.917..128.116 rows=116 loops=1)
               Hash Cond: ((m1.year = movies.year) AND (m1.duration = (min(movies.duration))))
               →  Seq Scan on movies m1  (cost=0.00..968.00 rows=50000 width=44) (actual time=0.011..30.162 rows=50000 loops=1)
               →  Hash  (cost=1219.76..1219.76 rows=88 width=8) (actual time=65.894..65.895 rows=88 loops=1)
                     Buckets: 1024  Batches: 1  Memory Usage: 12kB
                     →  HashAggregate  (cost=1218.00..1218.88 rows=88 width=8) (actual time=65.761..65.820 rows=90 loops=1)
                           Group Key: movies.year
                           →  Seq Scan on movies  (cost=0.00..968.00 rows=50000 width=8) (actual time=0.003..30.681 rows=50000 loops=1)
 Planning Time: 0.244 ms
 Execution Time: 128.581 ms
(14 rows)
```

.3 נראה שהשיפור בזמן ריצה נובע מכך שבשאילתה החדשה אין צורך לבצע את תת השאילתה עבור כל שורה בטבלה, אלא מחושבת פעם אחת ואז נבדקת מול כל שורה.

ב. האם אפשר לשפר את זמן הריצה של השאילתה המקורית (לפני השינוי מסעיף ב׳) ע״י הוספת אינדקס? בדקו אפשרויות שונות.

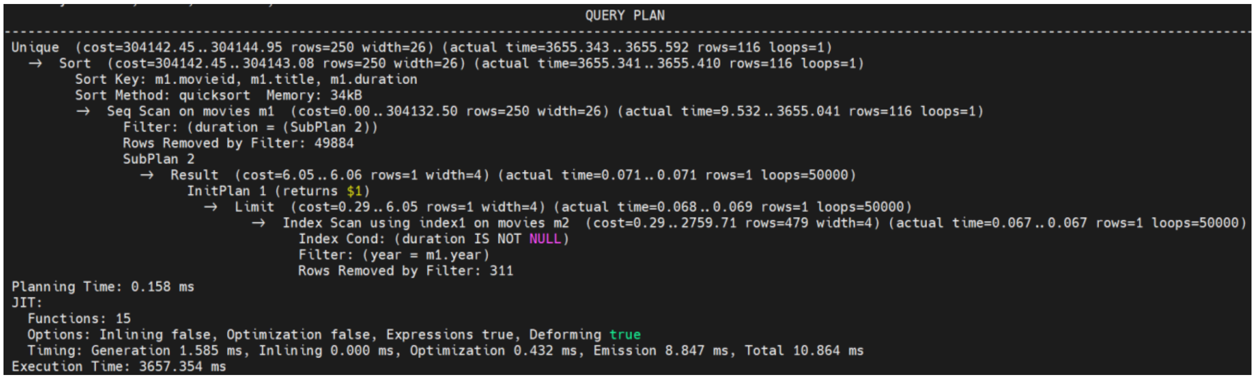
1. מצאו אינדקס אשר משפר את זמן ריצת השאילתה כך שהיא תרוץ בפחות מ30 שניות. כתבו בתשובה לסעיף זה את הפקודה לבניית האינדקס.
2. בנו את האינדקס והריצו את השאילתה עם פקודת *explain analyse*, שמראה את ה*query plan* של השאילתה, צרפו אותה לתשובות, וכתבו את זמן הריצה החדש.
3. נסו להסביר את השינוי בזמן הריצה.
   (איך צורך להסביר את כל הפרטים של ה-*query plan* רק מה גרם לשיפור)

1. יש אפשרויות שונות:

a) create index index1 on movies(duration);
b) create index index1 on movies(duration,year);
c) create index index1 on movies(year, duration);

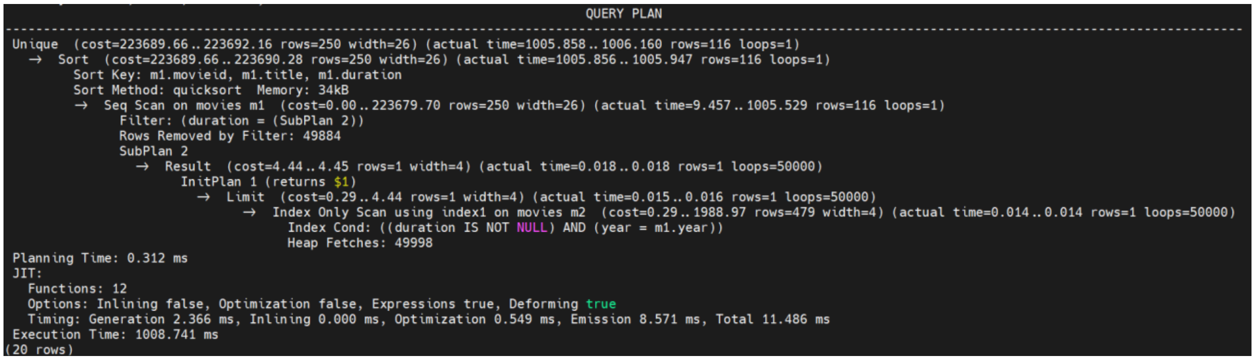
2.

a) זמן ריצה: כ 3 שניות



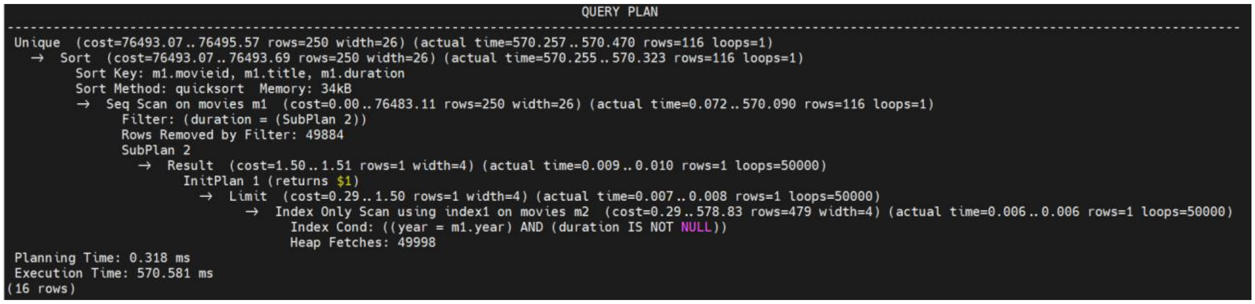
```
                                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
Unique  (cost=304142.45..304144.95 rows=250 width=26) (actual time=3655.343..3655.592 rows=116 loops=1)
  ->  Sort  (cost=304142.45..304143.08 rows=250 width=26) (actual time=3655.341..3655.410 rows=116 loops=1)
        Sort Key: m1.movieid, m1.title, m1.duration
        Sort Method: quicksort  Memory: 34kB
        ->  Seq Scan on movies m1  (cost=0.00..304132.50 rows=250 width=26) (actual time=9.532..3655.041 rows=116 loops=1)
              Filter: (duration = (SubPlan 2))
              Rows Removed by Filter: 49884
              SubPlan 2
                ->  Result  (cost=6.05..6.06 rows=1 width=4) (actual time=0.071..0.071 rows=1 loops=50000)
                      InitPlan 1 (returns $1)
                        ->  Limit  (cost=0.29..6.05 rows=1 width=4) (actual time=0.068..0.069 rows=1 loops=50000)
                              ->  Index Scan using index1 on movies m2  (cost=0.29..2759.71 rows=479 width=4) (actual time=0.067..0.067 rows=1 loops=50000)
                                    Index Cond: (duration IS NOT NULL)
                                    Filter: (year = m1.year)
                                    Rows Removed by Filter: 311
Planning Time: 0.158 ms
JIT:
  Functions: 15
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 1.585 ms, Inlining 0.000 ms, Optimization 0.432 ms, Emission 8.847 ms, Total 10.864 ms
Execution Time: 3657.354 ms
```

b) זמן ריצה כ-1.5 שניות



```
                                                                    QUERY PLAN
------------------------------------------------------------------------------------------------------------------------------------------------------------------
Unique  (cost=223689.66..223692.16 rows=250 width=26) (actual time=1005.858..1006.160 rows=116 loops=1)
  ->  Sort  (cost=223689.66..223690.28 rows=250 width=26) (actual time=1005.856..1005.947 rows=116 loops=1)
        Sort Key: m1.movieid, m1.title, m1.duration
        Sort Method: quicksort  Memory: 34kB
        ->  Seq Scan on movies m1  (cost=0.00..223679.70 rows=250 width=26) (actual time=9.457..1005.529 rows=116 loops=1)
              Filter: (duration = (SubPlan 2))
              Rows Removed by Filter: 49884
              SubPlan 2
                ->  Result  (cost=4.44..4.45 rows=1 width=4) (actual time=0.018..0.018 rows=1 loops=50000)
                      InitPlan 1 (returns $1)
                        ->  Limit  (cost=0.29..4.44 rows=1 width=4) (actual time=0.015..0.016 rows=1 loops=50000)
                              ->  Index Only Scan using index1 on movies m2  (cost=0.29..1988.97 rows=479 width=4) (actual time=0.014..0.014 rows=1 loops=50000)
                                    Index Cond: ((duration IS NOT NULL) AND (year = m1.year))
                                    Heap Fetches: 49998
Planning Time: 0.312 ms
JIT:
  Functions: 12
  Options: Inlining false, Optimization false, Expressions true, Deforming true
  Timing: Generation 2.366 ms, Inlining 0.000 ms, Optimization 0.549 ms, Emission 8.571 ms, Total 11.486 ms
Execution Time: 1008.741 ms
(20 rows)
```

c) זמן ריצה חצי שניה



```
                                                                                              QUERY PLAN
-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------
 Unique  (cost=76493.07..76495.57 rows=250 width=26) (actual time=570.257..570.470 rows=116 loops=1)
   →  Sort  (cost=76493.07..76493.69 rows=250 width=26) (actual time=570.255..570.323 rows=116 loops=1)
         Sort Key: m1.movieid, m1.title, m1.duration
         Sort Method: quicksort  Memory: 34kB
         →  Seq Scan on movies m1  (cost=0.00..76483.11 rows=250 width=26) (actual time=0.072..570.090 rows=116 loops=1)
               Filter: (duration = (SubPlan 2))
               Rows Removed by Filter: 49884
               SubPlan 2
                 →  Result  (cost=1.50..1.51 rows=1 width=4) (actual time=0.009..0.010 rows=1 loops=50000)
                       InitPlan 1 (returns $1)
                         →  Limit  (cost=0.29..1.50 rows=1 width=4) (actual time=0.007..0.008 rows=1 loops=50000)
                               →  Index Only Scan using index1 on movies m2  (cost=0.29..578.83 rows=479 width=4) (actual time=0.006..0.006 rows=1 loops=50000)
                                     Index Cond: ((year = m1.year) AND (duration IS NOT NULL))
                                     Heap Fetches: 49998
 Planning Time: 0.318 ms
 Execution Time: 570.581 ms
(16 rows)
```

3.

a. האינדקס על duration משמש לחיפוש יעיל של הערך המינימאלי עבור שנה מסוימת. עוברים על הערכים באינדקס אחד אחד מהכי קטן, עד שמוצאים ערך שהשנה שלו היא השנה המבוקשת, ואז זה הערך הרצוי.

b. כמו באינדקס הקודם, גם פה משתמשים באינדקס כדי לחפש את הערך המינימאלי, אבל במקרה הזה אין צורך לגשת לטבלה עצמה כי גם year נמצא באינדקס.

c. גם במקרה הזה מספיק לסרוק רק את האינדקס, מחפשים לפני שנה, ואז בודקים מבין כל ערכי הduration של אותה שנה (הנמצאים גם באינדקס) איזה ערך הוא המינימאלי. יוצא שהחיפוש הזה הוא הכי יעיל.

לגבי אינדקס על year בלבד, כנראה שהוא לא מספיק מייעל את השאילתה מכיוון שיש המון שורות שונות בטבלה עבור על שנה אז בפועל האינדקס עדיין צריך לגשת להמון בלוקים של הטבלה.

טיפ - כאשר אתם רוצים לנסות אינדקסים שונים מומלץ לתת שם לאינדקס בפקודת היצירה למשל:

```
CREATE INDEX <index_name> ON R (A);
```

כדי שתוכלו למחוק את האינדקס לפני שתנסו ליצור אינדקס חדש. מחיקה מתבצע ע״י הפקודה:

```
DROP INDEX <index_name>;
```

בהצלחה!