בס"ד

מיני פרויקט בבסיסי נתונים

ניהול ספריה באוניברסיטה

מגיש: עדן ישי כהן

בס"ד

תוכן עניינים:

שלב שני

הקדמה:

בשלב זה, התמקדתי בתשאול בסיס הנתונים לצורך הפקת מידע רלוונטי ומעמיק על נתונים בטבלאות המערכת. כתיבת השאילתות תוכננה כך שתספק מידע עשיר ושימושי, תוך שימוש בפקודות מתקדמות, חיתוכים, איחודים, ותתי-שאילתות.

שאילתות ללא פרמטרים:

Select Queries:

שאילתה 1: דירוג ממוצע של ספרים ומספר ביקורות

מטרת השאילתה:

השאילתה מספקת מידע מקיף על כל ספר בספרייה, כולל שם הספר, שפתו, תאריך הפרסום, קטגוריית הספר, שם המחבר המלא (כולל תואר אקדמי), הדירוג הממוצע שלו ומספר הביקורות שנכתבו עליו. המידע המתקבל מאפשר לספרייה להבין אילו ספרים זוכים להצלחה רבה מבחינת דירוגים וביקורות. בכך, הספרייה יכולה להחליט אילו ספרים לקדם או להזמין מחדש.

הסבר השאילתה:

השאילתה משלבת את טבלת הספרים Books עם טבלת הקטגוריות BookCategories, המחברים BookAuthors, והביקורות Reviews. היא עושה זאת באמצעות JOIN. הביקורות משויכות באופן שמאפשר לספור את מספר הביקורות ולחשב את הממוצע של הדירוגים לכל ספר באמצעות הפונקציות האגרגטיביות COUNT ו-AVG. כדי להבטיח שלא יהיו ערכים ריקים בדירוגים, נעשה שימוש בפונקציה NVL. לבסוף, התוצאות מסודרות לפי הדירוג הממוצע הגבוה ביותר, ובמקרה של דירוגים זהים – לפי מספר הביקורות.

```sql
SELECT
    B.Book_id,
    B.Book_name,
    B.Language,
    B.Publication_date,
    BC.Category,
    BA.author_academic_title || ' ' || BA.Author_first_name || ' ' || BA.Author_last_name AS Author,
    ROUND(NVL(AVG(R.Rating), 0), 2) AS Avg_Rating,
    COUNT(R.Reviewer_id) AS Total_Ratings
FROM
    Books B
JOIN
    BookCategories BC ON B.Category_id = BC.Category_id
JOIN
    BookAuthors BA ON B.Author_id = BA.Author_id
LEFT JOIN
    Reviews R ON B.Book_id = R.Book_id
GROUP BY
    B.Book_id,
    B.Book_name,
    B.Language,
    B.Publication_date,
    BC.Category,
    BA.author_academic_title,
    BA.Author_first_name,
    BA.Author_last_name
ORDER BY
    Avg_Rating DESC, Total_Ratings DESC;
```

	BOOK_ID	BOOK_NAME	LANGUAGE	PUBLICATION_DATE	CATEGORY	AUTHOR	AVG_RATING	TOTAL_RATINGS
1	3626	SQL for Data Management	Spanish	28-03-2021 00:00:00	Programming	Dr. Megan Taylor	3.77	31
2	3280	Introduction to Swift	Japanese	10-06-2019 00:00:00	Programming	Dr. Laura Green	3.76	29
3	3317	Introduction to HTML and CSS	Hindi	25-08-2018 00:00:00	Programming	Dr. Jessica Brown	3.75	28
4	3061	Time Series Analysis	Hindi	15-09-2020 00:00:00	Statistics	Prof. Linda Cohen	3.74	35
5	3063	Multivariate Analysis	Spanish	20-05-2018 00:00:00	Statistics	Prof. Linda Cohen	3.72	39
6	3566	Perl Programming	English	12-03-2019 00:00:00	Programming	Prof. William Taylor	3.7	20
7	3674	Scala for Developers	German	22-03-2021 00:00:00	Programming	Prof. Sarah Brown	3.69	29
8	3729	Dart for Mobile Development	Italian	10-02-2021 00:00:00	Programming	Dr. Brian Harris	3.67	18
9	3024	Introduction to Economics	English	12-01-2021 00:00:00	Economics	Prof. Rachel Green	3.64	28
10	3150	Introduction to Psychology	English	15-09-2020 00:00:00	Psychology	Dr. Emma Davis	3.63	30
11	3314	Introduction to Linguistics	Russian	22-11-2018 00:00:00	Linguistics	Dr. Patricia White	3.57	37

בס"ד

שאילתה 2: זמינות חדרי לימוד

מטרת השאילתה:
השאילתה מספקת מידע על חדרי הלימוד בספרייה, כולל מיקומם, קיבולתם, וזמן הזמינות הקרוב ביותר עבור חדרים שאינם תפוסים כעת. המידע מאפשר לצוות הספרייה לנהל את זמינות החדרים בצורה יעילה ולתכנן את השימוש בהם בצורה מיטבית.

הסבר השאילתה:
השאילתה משתמשת בטבלת StudyRooms לחדרי לימוד, ומחברת אותה עם טבלת RequestsRooms המכילה מידע על בקשות לשימוש בחדרים. באמצעות שימוש ב-LEFT JOIN, השאילתה בודקת אילו חדרים זמינים כעת, תוך שימוש בחיסור כדי לסנן חדרים שתפוסים. פונקציית CASE מספקת מידע האם החדר מצויד בלוח או מסך, ו-MIN מחשבת את זמן הזמינות הקרוב. תוצאות השאילתה כוללות גם את הזמן בדקות עד הזמינות הבאה.

```sql
SELECT
    SR.Room_number,
    SR.Library_floor,
    SR.Library_section,
    SR.Room_capacity,
    CASE
        WHEN SR.Board = 'Y' THEN 'Has Board'
        ELSE 'No Board'
    END AS Board_Status,
    CASE
        WHEN SR.Screen = 'Y' THEN 'Has Screen'
        ELSE 'No Screen'
    END AS Screen_Status,
    TO_CHAR(
        NVL(MIN(RR.Start_time), SYSDATE + 1),
        'YYYY-MM-DD HH24:MI:SS'
    ) AS Available_Until,
    ROUND(
        NVL(
            (MIN(RR.Start_time) - SYSDATE) * 24 * 60,
            24 * 60
        )
    ) AS Available_Minutes
FROM
    StudyRooms SR
LEFT JOIN
    RequestsRooms RR
    ON SR.Room_number = RR.Assigned_room
    AND RR.Start_time > SYSDATE
    AND RR.Status = 'approved'
WHERE
    SR.Room_number IN (
        SELECT
            SR.Room_number
        FROM
            StudyRooms SR
        MINUS
        SELECT
            RR.Assigned_room
        FROM
            RequestsRooms RR
        WHERE
            SYSDATE BETWEEN RR.Start_time AND RR.Start_time + (RR.Duration_hours / 24)
              AND RR.Status = 'approved'
    )
GROUP BY
    SR.Room_number, SR.Library_floor, SR.Library_section, SR.Room_capacity, SR.Board, SR.Screen
ORDER BY
    SR.Room_number;
```

	ROOM_NUMBER	LIBRARY_FLOOR	LIBRARY_SECTION	ROOM_CAPACITY	BOARD_STATUS	SCREEN_STATUS	AVAILABLE_UNTIL	AVAILABLE_MINUTES
1	4	2	R	6	Has Board	Has Screen	2024-11-26 11:00:00	16
2	6	1	Z	8	No Board	No Screen	2024-11-26 11:00:00	16
3	12	2	H	3	No Board	No Screen	2024-11-26 12:00:00	76
4	14	1	I	15	No Board	No Screen	2024-11-26 12:00:00	76
5	21	1	C	3	No Board	No Screen	2024-11-26 11:00:00	16
6	23	1	S	7	No Board	No Screen	2024-11-26 11:00:00	16
7	34	2	W	11	No Board	Has Screen	2024-11-26 11:00:00	16
8	42	1	Y	6	No Board	Has Screen	2024-11-26 11:00:00	16
9	49	1	M	15	No Board	No Screen	2024-11-26 12:00:00	76

הורץ בשעה הזאת:

10:44
26/11/2024

שאילתה 3: סטודנטים עם חובות וכמות השאלות

מטרת השאילתה:
שאילתה זו מיועדת להפיק רשימה של סטודנטים שיש להם חובות בספרייה, כולל מידע על מספר הספרים שהשאילו והסכום הכולל של החובות שהם חייבים. מידע זה עוזר לצוות הספרייה לנהל מעקב אחר חובות ולהתמקד בגבייה.

הסבר השאילתה:
השאילתה מבוססת על שילוב של טבלת Students עם טבלת Lent, באמצעות LEFT JOIN כדי להבטיח שהנתונים כוללים גם סטודנטים ללא השאלות פעילות. היא משתמשת בפונקציות אגרגטיביות כמו SUM ו-COUNT לצורך חישוב הסכום הכולל של החובות ומספר ההשאלות לכל סטודנט. תנאי ה-HAVING מבטיחים שהתוצאה תכלול רק סטודנטים עם לפחות השאלה אחת וחוב חיובי.

```sql
SELECT
    S.Student_id,
    S.Student_first_name || ' ' || S.Student_last_name AS Student_Name,
    S.Phone AS Phone_Number,
    S.Email AS Email_Address,
    COUNT(L.Book_id) AS Total_Books_Borrowed,
    SUM(L.Fine_amount) AS Total_Fines
FROM
    Students S
LEFT JOIN
    Lent L ON S.Student_id = L.Student_id
GROUP BY
    S.Student_id, S.Student_first_name, S.Student_last_name, S.Phone, S.Email
HAVING
    SUM(L.Fine_amount) > 0  -- Only students who have paid fines
    AND COUNT(L.Book_id) >= 1  -- Only students who have borrowed at least one book
ORDER BY
    Total_Fines DESC, Total_Books_Borrowed DESC;
```

	STUDENT_ID	STUDENT_NAME	PHONE_NUMBER	EMAIL_ADDRESS	TOTAL_BOOKS_BORROWED	TOTAL_FINES
1	392543765	Ofir Peretz	056-5492996	ofir_peretz01@jct.ac.il	31	555
2	356852699	Mor Tzur	053-5539612	mor91tzur@gmail.com	27	481
3	253051841	Alon Gross	050-7577848	alon_gross@jct.ac.il	30	472
4	359581477	Tal Zohar	057-0257474	tal_zohar@gmail.com	29	468
5	352136873	David Kramer	050-0511247	david_kramer@gmail.com	31	464
6	305552788	Roni Berkowitz	057-7617134	roni_berkowitz90@jct.ac.il	30	464
7	263562407	Dean Ben-Shitrit	056-6903390	dean96ben-shitrit@jct.ac.il	29	457
8	222457913	Niv Tayeb	056-2555041	niv_tayeb97@gmail.com	22	452
9	335121067	Michael Shamir	059-9198077	michael_shamir@jct.ac.il	31	451
0	327028676	Michael Kramer	059-5927219	michael_kramer@gmail.com	29	442

שאילתה 4: מחברים וספרים

מטרת השאילתה:
השאילתה נועדה להפיק רשימה של מחברים, כולל המידע על מספר הספרים שכתבו, הדירוג הממוצע של ספריהם, וסך כל הביקורות שקיבלו. מידע זה עוזר לספרייה לזהות מחברים פופולריים ולנהל את ההיצע שלהם.

הסבר השאילתה:
השאילתה משלבת את טבלת BookAuthors עם הטבלאות Books ו-Reviews. היא משתמשת בתת-שאילתה מחושבת שמסכמת את הביקורות והדירוגים לכל ספר. הפונקציות האגרגטיביות COUNT ו-SUM מחשבות את סך הביקורות ומספר הספרים שכתב כל מחבר. לאחר מכן, התוצאה ממוין לפי מספר הספרים, הדירוג הממוצע, ושם המחבר.

```sql
SELECT
    BA.Author_id,
    BA.Author_academic_title || ' ' || BA.Author_first_name || ' ' || BA.Author_last_name AS Author_Name,
    BA.Nationality,
    TO_CHAR(BA.Author_birth_date, 'YYYY-MM-DD') AS Birth_Date,
    COUNT(B.Book_id) AS Total_Books_Written,
    NVL(SUM(R.Total_Ratings), 0) AS Total_Ratings_Received,
    ROUND(NVL(SUM(R.Total_Score) / SUM(R.Total_Ratings), 0), 2) AS Avg_Rating
FROM
    BookAuthors BA
LEFT JOIN
    Books B ON BA.Author_id = B.Author_id
LEFT JOIN
    (
        SELECT
            Book_id,
            COUNT(*) AS Total_Ratings,
            SUM(Rating) AS Total_Score
        FROM
            Reviews
        GROUP BY
            Book_id
    ) R ON B.Book_id = R.Book_id
GROUP BY
    BA.Author_id,
    BA.Author_academic_title,
    BA.Author_first_name,
    BA.Author_last_name,
    BA.Nationality,
    BA.Author_birth_date
ORDER BY
    Total_Books_Written DESC, Avg_Rating DESC, Author_Name ASC;
```

	AUTHOR_ID	AUTHOR_NAME	NATIONALITY	BIRTH_DATE	TOTAL_BOOKS_WRITTEN	TOTAL_RATINGS_RECEIVED	AVG_RATING
1	1008	Prof. Linda Cohen	Spain	2007-11-23	8	260	3.26
2	1007	Dr. Alan Berger	Israel	1975-06-26	8	252	3.13
3	1001	Dr. Jacob Levi	Germany	1948-04-14	8	223	3.07
4	1005	Dr. David Katz	Canada	1980-01-06	8	253	3.06
5	1004	Prof. Rachel Green	Canada	1949-01-23	8	249	3.06
6	1006	Prof. Sarah Levy	France	1980-09-09	8	248	3.03
7	1003	Dr. Michael Stern	Israel	1987-05-15	8	230	2.98
8	1009	Dr. Robert Stein	India	2001-05-26	8	241	2.92
9	1002	Prof. Dana Cohen	Korea	1945-01-04	8	250	2.82
0	1023	Dr. Steven Miller	Japan	1984-07-14	5	150	3.2
1	1012	Dr. Laura Green	China	1941-06-11	5	153	3.14
2	1017	Dr. Paul Adams	USA	1940-06-28	5	151	3.07
3	1014	Prof. Anna Lewis	Spain	1948-03-02	5	167	3.06
4	1015	Dr. Mark Wilson	Russia	1985-04-11	5	150	3.05

שאילתה 5: דירוגים והשאלות הגבוהים ביותר

מטרת השאילתה:
שאילתה זו מספקת מידע על 5 הספרים עם הדירוגים הממוצעים הגבוהים ביותר, ו-5 הספרים שהושאלו הכי הרבה. המידע מאפשר לצוות הספרייה להבחין בין ספרים פופולריים מבחינת השאלות לבין ספרים שמוערכים מבחינת איכות.

הסבר השאילתה:
השאילתה מחולקת לשני חלקים, המאוחדים באמצעות UNION ALL. החלק הראשון מחלץ את 5 הספרים עם הדירוג הממוצע הגבוה ביותר, ומציג את פרטי הספר כולל הקטגוריה ותת-הקטגוריה. החלק השני מחלץ את 5 הספרים עם מספר ההשאלות הגבוה ביותר, תוך שימוש בפונקציה COUNT וביצוע LEFT JOIN לטבלת Lent. התוצאות כוללות שם קטגוריה ותת-קטגוריה, כדי לספק הקשר רחב על הספרים שנבחרו.

```sql
SELECT *
FROM (
    SELECT
        B.Book_id,
        B.Book_name,
        B.Language,
        B.Publication_date,
        BC.Category AS Category_Name,
        BC.Subcategory AS Subcategory_Name,
        ROUND(AVG(R.Rating), 2) AS Avg_Rating,
        NULL AS Total_Borrows
    FROM
        Books B
    LEFT JOIN
        Reviews R ON B.Book_id = R.Book_id
    LEFT JOIN
        BookCategories BC ON B.Category_id = BC.Category_id
    GROUP BY
        B.Book_id, B.Book_name, B.Language, B.Publication_date, BC.Category, BC.Subcategory
    ORDER BY
        Avg_Rating DESC
) WHERE ROWNUM <= 5

UNION ALL

SELECT *
FROM (
    SELECT
        B.Book_id,
        B.Book_name,
        B.Language,
        B.Publication_date,
        BC.Category AS Category_Name,
        BC.Subcategory AS Subcategory_Name,
        NULL AS Avg_Rating,
        COUNT(L.Book_id) AS Total_Borrows
    FROM
        Books B
    LEFT JOIN
        Lent L ON B.Book_id = L.Book_id
    LEFT JOIN
        BookCategories BC ON B.Category_id = BC.Category_id
    GROUP BY
        B.Book_id, B.Book_name, B.Language, B.Publication_date, BC.Category, BC.Subcategory
    ORDER BY
        Total_Borrows DESC
) WHERE ROWNUM <= 5;
```

	BOOK_ID	BOOK_NAME	LANGUAGE	PUBLICATION_DATE	CATEGORY_NAME	SUBCATEGORY_NAME	AVG_RATING	TOTAL_BORROWS
1	3626	SQL for Data Management	Spanish	28-03-2021 00:00:00	Programming	SQL	3.77	(null)
2	3280	Introduction to Swift	Japanese	10-06-2019 00:00:00	Programming	Swift	3.76	(null)
3	3317	Introduction to HTML and CSS	Hindi	25-08-2018 00:00:00	Programming	Web Development	3.75	(null)
4	3061	Time Series Analysis	Hindi	15-09-2020 00:00:00	Statistics	Time Series	3.74	(null)
5	3063	Multivariate Analysis	Spanish	20-05-2018 00:00:00	Statistics	Multivariate Analysis	3.72	(null)
6	3657	Dart for Mobile Development	English	10-02-2021 00:00:00	Programming	Dart	(null)	99
7	3306	Rust for Systems Programming	Spanish	12-09-2021 00:00:00	Programming	Rust	(null)	95
8	3378	JavaScript Basics	French	30-10-2018 00:00:00	Programming	JavaScript	(null)	95
9	3389	Kotlin for Mobile Development	Japanese	05-11-2020 00:00:00	Programming	Kotlin	(null)	93
10	3510	Ruby Programming	Italian	20-03-2019 00:00:00	Programming	Ruby	(null)	92

בס"ד

שאילתה 6: ספרים עם דירוג נמוך והשאלות רבות

מטרת השאילתה:
השאילתה מזהה ספרים עם דירוג ממוצע נמוך מהממוצע הכולל של הספרים, אך עם מספר השאלות הגבוה מהממוצע הכולל. המידע מאפשר להבין אילו ספרים פופולריים למרות הדירוג הנמוך שלהם.

הסבר השאילתה:
השאילתה מבוססת על חיתוך (INTERSECT) בין שתי קבוצות ספרים. הקבוצה הראשונה כוללת ספרים עם דירוג ממוצע נמוך מהממוצע, והקבוצה השנייה כוללת ספרים עם מספר השאלות מעל הממוצע. התוצאה מציגה פרטי ספר מלאים, כולל קטגוריה, מחבר, דירוג ממוצע ומספר השאלות.
חישוב הממוצעים מתבצע בתתי-שאילתות נפרדות.

```sql
SELECT
    B.Book_id,
    B.Book_name,
    B.Language,
    B.Publication_date,
    BC.Category AS Book_Category,
    BA.Author_academic_title || ' ' || BA.Author_first_name || ' ' || BA.Author_last_name AS Author_Name,
    AVG_R.Avg_Rating,
    BORROWS.Total_Borrows
FROM (
    SELECT
        Book_id
    FROM (
        SELECT
            B.Book_id
        FROM
            Books B
        LEFT JOIN
            Reviews R ON B.Book_id = R.Book_id
        GROUP BY
            B.Book_id
        HAVING
            ROUND(AVG(R.Rating), 2) < (
                SELECT ROUND(AVG(Avg_Rating), 2)
                FROM (
                    SELECT
                        B.Book_id,
                        ROUND(AVG(R.Rating), 2) AS Avg_Rating
                    FROM
                        Books B
                    LEFT JOIN
                        Reviews R ON B.Book_id = R.Book_id
                    GROUP BY
                        B.Book_id
                )
            )
    )
    INTERSECT
```

```sql
    SELECT
        Book_id
    FROM (
        SELECT
            B.Book_id
        FROM
            Books B
        LEFT JOIN
            Lent L ON B.Book_id = L.Book_id
        GROUP BY
            B.Book_id
        HAVING
            COUNT(L.Book_id) > (
                SELECT ROUND(AVG(Total_Borrows), 2)
                FROM (
                    SELECT
                        B.Book_id,
                        COUNT(L.Book_id) AS Total_Borrows
                    FROM
                        Books B
                    LEFT JOIN
                        Lent L ON B.Book_id = L.Book_id
                    GROUP BY
                        B.Book_id
                )
            )
    )
) INTERSECT_BOOKS
JOIN Books B ON INTERSECT_BOOKS.Book_id = B.Book_id
JOIN BookCategories BC ON B.Category_id = BC.Category_id
JOIN BookAuthors BA ON B.Author_id = BA.Author_id
LEFT JOIN (
    SELECT
        B.Book_id,
        ROUND(AVG(R.Rating), 2) AS Avg_Rating
    FROM
        Books B
    LEFT JOIN
        Reviews R ON B.Book_id = R.Book_id
    GROUP BY
        B.Book_id
) AVG_R ON B.Book_id = AVG_R.Book_id
LEFT JOIN (
    SELECT
        B.Book_id,
        COUNT(L.Book_id) AS Total_Borrows
    FROM
        Books B
    LEFT JOIN
        Lent L ON B.Book_id = L.Book_id
    GROUP BY
        B.Book_id
) BORROWS ON B.Book_id = BORROWS.Book_id
ORDER BY B.Book_name;
```

BOOK_ID	BOOK_NAME	LANGUAGE	PUBLICATION_DATE	BOOK_CATEGORY	AUTHOR_NAME	AVG_RATING	TOTAL_BORROWS
3551	Advanced Kotlin Programming	Hindi	12-08-2021	Programming	Prof. Brian Harris	3	73
3009	Advanced Organic Chemistry	Korean	15-01-2020	Chemistry	Prof. Dana Cohen	2.97	74
3361	Advanced Perl Programming	English	22-11-2020	Programming	Dr. Sarah Roberts	2.95	79
3311	Advanced Perl Programming	Russian	05-10-2020	Programming	Prof. Steven White	2.93	75
3633	Advanced PHP Programming	English	25-09-2020	Programming	Prof. Brian Clark	2.83	84
3687	Advanced PHP Programming	Spanish	22-05-2021	Programming	Dr. Jessica Martinez	2.81	76
3300	Advanced Scala Programming	Hindi	05-10-2020	Programming	Prof. Thomas Green	2.84	80
3491	Advanced SQL Programming	Spanish	10-02-2021	Programming	Prof. Robert Miller	2.58	77
3316	Advanced SQL Programming	Korean	18-10-2020	Programming	Prof. Robert Taylor	2.95	81
3447	Advanced SQL Queries	French	30-11-2020	Programming	Prof. Jessica Lewis	2.91	77
3565	Advanced TypeScript Programming	Chinese	18-12-2020	Programming	Dr. Megan Brown	2.88	72
3441	Advanced TypeScript Programming	English	22-05-2020	Programming	Dr. Charles White	2.77	82
3114	Art Criticism	Spanish	30-09-2018	Art History	Prof. Lisa Brown	2.91	77
3038	Artificial Intelligence	French	20-02-2022	Computer Science	Dr. David Katz	2.81	71
[illegible]	[illegible]	English	28-02-2022	Chemistry	Dr. Steven Miller	2.8	73

בס"ד

Update Queries:

שאילתה 1: עדכון מספר העותקים הזמינים לפי מצב הספר

מטרת השאילתה:
מטרת השאילתה היא להוסיף ספרים לספרייה (לעדכן את מספר העותקים במצב "זמין" של כל ספר בטבלת BookStatuses). העדכון מבוסס על הסכום הכולל של העותקים בכל המצבים האפשריים של הספר: "זמין", "פגום", ו"אבוד" כלומר, להוסיף כמות עותקים כמספר הספרים שניזקו ונאבדו עבור ספר ספציפי. עם זאת, העדכון מתבצע רק עבור ספרים שעמדו בקריטריון של מספר השאלות העולה על ממוצע מספר ההשאלות של כל הספרים, בתוספת 20.

הסבר השאילתה:
השאילתה מעדכנת את השדה Number_of_copies בספרים שנמצאים במצב "זמין". לשם כך, היא מבצעת חישוב פנימי באמצעות שאילתה מקוננת שמסכמת את העותקים בכל המצבים הרלוונטיים עבור כל ספר. התנאי בחלק WHERE מבטיח שהעדכון יתבצע רק עבור ספרים שעומדים בקריטריון של מספר השאלות. חישוב קריטריון זה מתבצע באמצעות שאילתה פנימית נוספת, שבה נמדד ממוצע כלל מספר ההשאלות, ואליו מתווספת תוספת קבועה של 20. בצורה זו, השאילתה מוודאת שרק ספרים פופולריים שעמדו בקריטריון זוכים לעדכון מספר העותקים.

```
UPDATE BookStatuses BS
SET
    BS.Number_of_copies = (
        SELECT
            SUM(CASE WHEN InnerBS.Status = 'available' THEN InnerBS.Number_of_copies ELSE 0 END) +
            SUM(CASE WHEN InnerBS.Status = 'damaged' THEN InnerBS.Number_of_copies ELSE 0 END) +
            SUM(CASE WHEN InnerBS.Status = 'lost' THEN InnerBS.Number_of_copies ELSE 0 END)
                FROM
            BookStatuses InnerBS
        WHERE
            InnerBS.Book_id = BS.Book_id
    )
WHERE
    BS.Status = 'available'
    AND BS.Book_id IN (
        SELECT
            L.Book_id
        FROM
            Lent L
        GROUP BY
            L.Book_id
        HAVING
            COUNT(*) >= (
                SELECT
                    ROUND(AVG(Total_Borrows), 2) + 20
                FROM (
                    SELECT
                        COUNT(*) AS Total_Borrows
                    FROM
                        Lent
                    GROUP BY
                        Book_id
                )
            )
    );
```

הערכים שצריכים להתעדכן לפני ההרצה

	BOOK_ID	BOOK_NAME	AVAILABLE_COPIES	LOST_COPIES	DAMAGED_COPIES
1	3086	Introduction to Psychology	13	1	4
2	3306	Rust for Systems Programming	13	3	5
3	3378	JavaScript Basics	14	2	4
4	3380	Java Programming Basics	11	1	3
5	3389	Kotlin for Mobile Development	9	1	1
6	3510	Ruby Programming	11	2	2
7	3623	TypeScript for Angular	9	0	3
8	3657	Dart for Mobile Development	10	1	1
9	3705	Advanced PHP Programming	13	3	2

הרצת השאילתה

Query Result | Script Output | Task completed in

9 rows updated.

הערכים אחרי עדכון

	BOOK_ID	BOOK_NAME	AVAILABLE_COPIES	LOST_COPIES	DAMAGED_COPIES
1	3086	Introduction to Psychology	18	1	4
2	3306	Rust for Systems Programming	21	3	5
3	3378	JavaScript Basics	20	2	4
4	3380	Java Programming Basics	15	1	3
5	3389	Kotlin for Mobile Development	11	1	1
6	3510	Ruby Programming	15	2	2
7	3623	TypeScript for Angular	12	0	3
8	3657	Dart for Mobile Development	12	1	1
9	3705	Advanced PHP Programming	18	3	2

אפשר לראות שהסכום החדר של הספרים הזמינים הוא תוספת של הספרים שנאבדו וניזקו לספרים הזמינים.

שאילתה 2: שינוי סטטוס בקשות חדרים לסטטוס מותנה בתשלום

מטרת השאילתה:
מטרת השאילתה היא לעדכן את הסטטוס של בקשות לחדרים בטבלת RequestsRooms. הבקשות המעודכנות הן אלו שמאושרות, אבל התנאי לעדכון הוא שהסטודנט שביצע את הבקשה חייב סכום חוב של לפחות 200 שקלים או שהבקשה מתייחסת לתקופה עתידית. במקרה כזה, הסטטוס מתעדכן ל-"approved only if paid".

הסבר השאילתה:
שאילתה זו מעדכנת את הסטטוס של בקשות בטבלת RequestsRooms ל "approved only if paid"- עבור בקשות עתידיות שבהן הסטודנט החייב סכום כולל של מעל 200 שקלים. סינון הבקשות מבוצע לפי זמן ההתחלה (Start_time > SYSDATE) ולפי חוב הסטודנט, המחושב באמצעות SUM(Fine_amount) לאחר קיבוץ לפי מזהה הסטודנט. רק בקשות עם סטטוס "approved" מתעדכנות לסטטוס החדש.

```
UPDATE RequestsRooms
SET Status = 'approved only if paid'
WHERE Start_time > SYSDATE
AND Student_id IN (
    SELECT Student_id
    FROM Lent
    GROUP BY Student_id
    HAVING SUM(Fine_amount) > 200
) AND Status = 'approved';
```

הסטטוסים השונים של בקשות הספרייה

```
SELECT count(*), status
FROM RequestsRooms
WHERE Start_time > SYSDATE
group by status;
```

Script Output | Query Result | SQL | All Rows Fetched: 2 in 0.003 sec

	COUNT(*)	STATUS
1	908	approved
2	567	rejected

הרצת השאילתה

```
467 rows updated.
```

הערכים אחרי עדכון

```
SELECT count(*), status
FROM RequestsRooms
WHERE Start_time > SYSDATE
group by status;
```

Script Output | Query Result | SQL | All Rows Fetched: 3 in 0.002 seconds

	COUNT(*)	STATUS
1	441	approved
2	467	approved only if paid
3	567	rejected

בס"ד

Delete Queries:

שאילתה 1: מחיקת ספרנים עם פחות משימות שהושלמו

מטרת השאילתה:
מטרת השאילתה היא למחוק עד חמישה ספרנים שיש להם את מספר המשימות שהושלמו הנמוך ביותר, ובמקרה של תיקו במספר המשימות, סדר המחיקה יהיה לפי תאריך תחילת העבודה המוקדם ביותר.

הסבר השאילתה:
השאילתה עושה שימוש בפקודת DELETE למחיקת ספרנים מטבלת Librarians. כדי לזהות את הספרנים למחיקה, השאילתה משתמשת בפקודות SELECT פנימיות.
תחילה, מתבצע חישוב של מספר המשימות שהושלמו על ידי כל ספרן באמצעות COUNT. לאחר מכן, נבחרים הספרנים עם מספר המשימות הנמוך ביותר. במידה ויש מספר ספרנים עם אותו מספר משימות, הסינון נעשה לפי תאריך תחילת העבודה המוקדם ביותר. מגבלת ROWNUM <= 5 מוודאת כי עד חמישה ספרנים יימחקו.

```sql
DELETE FROM Librarians
WHERE Librarian_id IN (
    SELECT Librarian_id
    FROM (
        SELECT
            L.Librarian_id,
            COUNT(RA.Request_id) AS Completed_Tasks,
            L.Hire_date
        FROM
            Librarians L
        LEFT JOIN
            RequestAssignments RA ON L.Librarian_id = RA.Assigned_librarian
        GROUP BY
            L.Librarian_id, L.Hire_date
        HAVING
            COUNT(RA.Request_id) = (
                SELECT MIN(TaskCount)
                FROM (
                    SELECT
                        COUNT(RA2.Request_id) AS TaskCount
                    FROM
                        Librarians L2
                    LEFT JOIN
                        RequestAssignments RA2 ON L2.Librarian_id = RA2.Assigned_librarian
                    GROUP BY
                        L2.Librarian_id
                    ORDER BY
                        TaskCount ASC, L2.Hire_date ASC
                )
            )
        ORDER BY
            Completed_Tasks ASC, Hire_date ASC
    )
    WHERE ROWNUM <= 5
);
```

	COUNT_LIBRARIANS
1	400

	LIBRARIAN_ID	COMPLETED_TASKS	HIRE_DATE
1	397795675	0	10-02-1982 00:00:00
2	385533765	0	28-11-1985 00:00:00
3	356568279	0	19-08-1988 00:00:00
4	231245507	0	05-04-1989 00:00:00
5	282181411	0	18-11-2002 00:00:00
6	284819760	0	07-11-2006 00:00:00
7	279782528	0	28-02-2013 00:00:00
8	209347681	0	04-05-2014 00:00:00
9	336757406	0	27-12-2014 00:00:00
10	392214409	0	02-01-2015 00:00:00
11	263340143	0	21-06-2015 00:00:00

5 rows deleted.

	COUNT_LIBRARIANS
1	395

	LIBRARIAN_ID	COMPLETED_TASKS	HIRE_DATE
1	284819760	0	07-11-2006 00:00:00
2	279782528	0	28-02-2013 00:00:00
3	209347681	0	04-05-2014 00:00:00
4	336757406	0	27-12-2014 00:00:00
5	392214409	0	02-01-2015 00:00:00
6	263340143	0	21-06-2015 00:00:00
7	209926450	0	06-06-2017 00:00:00
8	250628864	0	29-06-2018 00:00:00
9	206257180	0	17-09-2018 00:00:00
10	202213385	0	08-04-2019 00:00:00

צירפתי צילומי מסך של כמות הספרנים לפני המחיקה וספרנים עם המשימות שהושלמו הנמוך ביותר לפי הסדר

של המחיקה

ושל התוצאות של אחרי המחיקה שבו רואים שיש רק 395 ספרנים וגם ה-5 ספרנים הראשונים ממקודם כבר לא מופיעים בטבלה

בס"ד

שאילתה 2: מחיקת בקשות חדרים עם שעות מצטברות מעל 10

מטרת השאילתה:
מטרת השאילתה היא למחוק בקשות לחדרים שבהן סטודנטים עברו את המגבלה של 10 שעות מצטברות שהוגשו להחל מתאריך מחר והלאה.

הסבר השאילתה:
השאילתה משתמשת בפקודת DELETE למחיקת בקשות מתוך טבלת RequestsRooms. הבקשות נבחרות אם הסטודנט עבר את מגבלת 10 השעות המצטברות באמצעות שאילתה פנימית הכוללת חיבור עצמי לטבלה (LEFT JOIN).
על מנת לוודא שהבקשות נלקחות מהתאריך של מחר והלאה, נוסף סינון על שדה Start_time באמצעות הפונקציה TRUNC(SYSDATE + 1). התנאים בפקודת HAVING משמשים כדי לוודא שרק הבקשות המתאימות למחיקה נבחרות.

```sql
DELETE FROM RequestsRooms
WHERE (Student_id, Request_time) IN (
    SELECT
        RR.Student_id,
        RR.Request_time
    FROM
        RequestsRooms RR
    LEFT JOIN
        RequestsRooms PreviousRR
    ON
        RR.Student_id = PreviousRR.Student_id
        AND PreviousRR.Start_time <= RR.Start_time
        AND PreviousRR.Status = 'approved'
    WHERE
        RR.Status = 'approved'
    GROUP BY
        RR.Student_id,
        RR.Request_time,
        RR.Start_time,
        RR.Duration_hours,
        RR.Number_of_people,
        RR.Board,
        RR.Screen,
        RR.Status
    HAVING
        RR.Start_time >= TRUNC(SYSDATE + 1)
        AND NVL(SUM(PreviousRR.Duration_hours), 0) > 10
);
```

```sql
SELECT COUNT(*) AS total_requests_before_delete
FROM RequestsRooms;
```

pt Output × | Query Result ×

SQL | All Rows Fetched: 1 in 0.002 seconds

	TOTAL_REQUESTS_BEFORE_DELETE
1	15086

	STUDENT_ID	REQUEST_TIME	START_TIME	DURATION_HOURS	STATUS	TOTAL_PREVIOUS_HOURS
145	221045545	02-10-2024 21:12:06	29-11-2024 14:00:00	1.5	approved	13.5
146	204829154	02-10-2024 23:03:39	30-11-2024 18:00:00	1.5	approved	12.5
147	205187552	28-10-2024 16:06:39	30-11-2024 21:00:00	1	approved	13.5
148	333881050	18-10-2024 20:03:56	29-11-2024 08:00:00	2.5	approved	14.5
149	329552665	01-10-2024 11:44:53	30-11-2024 13:00:00	1.5	approved	13
150	346146194	29-10-2024 15:33:33	30-11-2024 09:00:00	2.5	approved	14.5
151	297471526	02-11-2024 22:19:13	30-11-2024 20:00:00	1.5	approved	13.5
152	268437571	02-10-2024 04:24:43	29-11-2024 07:00:00	1.5	approved	15
153	365657014	12-10-2024 22:52:53	30-11-2024 09:00:00	2.5	approved	14
154	338160435	25-10-2024 19:14:20	29-11-2024 12:00:00	3	approved	11.5
155	206229643	31-10-2024 00:41:26	29-11-2024 12:00:00	1	approved	10.5
156	279388201	10-10-2024 07:37:01	30-11-2024 15:00:00	2.5	approved	15
157	258047844	09-10-2024 02:06:07	29-11-2024 08:00:00	3.5	approved	12.5
158	297626657	05-10-2024 05:49:38	30-11-2024 21:00:00	1	approved	13.5
159	320326697	02-11-2024 15:52:55	29-11-2024 10:00:00	3	approved	14
160	252362025	12-10-2024 22:51:16	30-11-2024 17:00:00	1.5	approved	12
161	241526706	17-10-2024 09:42:21	29-11-2024 17:00:00	1	approved	11.5
162	360152433	07-10-2024 19:24:04	30-11-2024 21:00:00	1	approved	11
163	299387167	17-10-2024 13:58:23	30-11-2024 07:00:00	1	approved	11.5
164	348755422	12-10-2024 01:31:44	29-11-2024 19:00:00	1	approved	10.5
165	214414633	14-10-2024 12:48:02	30-11-2024 19:00:00	1	approved	13
166	375886579	11-10-2024 09:15:11	30-11-2024 08:00:00	4	approved	12

166 rows deleted.

הוספתי צילומי מסך של: כמות בקשות החדרי לימוד לפני המחיקה וגם של הבקשות הצפויות להימחק.

צירפתי אח"כ תמונה של ההדפסה של 166 שורות נמחקו

ואז הוספתי צילומי מסך של: כמות בקשות החדרי לימוד אחרי המחיקה.

	TOTAL_REQUESTS_AFTER_DELETE
1	14920

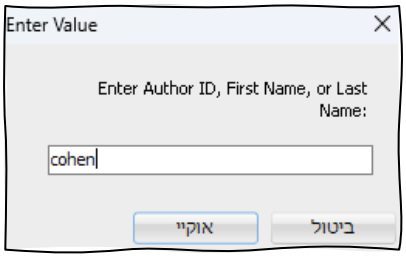
שאילתות עם פרמטרים:

שאילתה 1: שליפת ספרים לפי מזהה מחבר או שם מחבר

מטרת השאילתה:
לספק רשימה של ספרים הקשורים למחבר מסוים, בהתבסס על מזהה מחבר או שם פרטי/משפחה. השאילתה נועדה להציג מידע מקיף כמו שם הספר, תאריך פרסום, קטגוריה, תת-קטגוריה, שם מלא של המחבר ודירוג ממוצע של הספרים. פעולה זו מסייעת לספרייה לזהות ספרים פופולריים או משמעותיים שנכתבו על ידי המחבר.

הסבר השאילתה:
השאילתה מחברת את טבלאות הספרים, המחברים, הקטגוריות והביקורות. היא משתמשת בפרמטר לחיפוש גמיש על פי מזהה מחבר, שם פרטי או שם משפחה. פונקציות אגרגטיביות משמשות לחישוב דירוג ממוצע, עם שימוש ב-NVL לטיפול בערכים חסרים. הנתונים מסודרים לפי מזהה מחבר, דירוג ממוצע ושם הספר.

```oracle_sql
-##############################################################
- Accept parameter for author ID or name
-##############################################################
ACCEPT author_identifier CHAR PROMPT 'Enter Author ID, First Name, or Last Name: ';

-##############################################################
- Define a reusable variable for the parameter
-##############################################################
DEFINE author_param = '&author_identifier';

-##############################################################
- Query to retrieve books by author
-##############################################################
SELECT
    B.Book_id,
    B.Book_name,
    TO_CHAR(B.Publication_date, 'YYYY-MM-DD') AS Publication_Date,
    BA.Author_id AS Author_ID,
    BA.author_academic_title || ' ' || BA.Author_first_name || ' ' || BA.Author_last_name AS Author_Name,
    BC.Category AS Book_Category,
    BC.Subcategory AS Book_Subcategory,
    NVL(ROUND(AVG(R.Rating), 2), 0) AS Avg_Rating
FROM
    Books B
JOIN
    BookAuthors BA ON B.Author_id = BA.Author_id
JOIN
    BookCategories BC ON B.Category_id = BC.Category_id
LEFT JOIN
    Reviews R ON B.Book_id = R.Book_id
WHERE
    (
        (REGEXP_LIKE('&&author_param', '^[0-9]+$') AND TO_CHAR(BA.Author_id) = '&&author_param')
        OR
        (LOWER(BA.Author_first_name) LIKE LOWER('&&author_param') OR
            LOWER(BA.Author_last_name) LIKE LOWER('&&author_param'))
    )
GROUP BY
    B.Book_id, B.Book_name, B.Publication_date,
    BA.Author_id, BA.author_academic_title, BA.Author_first_name, BA.Author_last_name,
    BC.Category, BC.Subcategory
ORDER BY
    BA.Author_id ASC, Avg_Rating DESC, B.Book_name ASC;
```



	BOOK_ID	BOOK_NAME	PUBLICATION_DATE	AUTHOR_ID	AUTHOR_NAME	BOOK_CATEGORY	BOOK_SUBCATEGORY	AVG_RATING
1	3012	Spectroscopy in Organic Chemistry	2017-03-25	1002	Prof. Dana Cohen	Chemistry	Organic Chemistry	3.29
2	3014	Medicinal Chemistry	2020-10-18	1002	Prof. Dana Cohen	Chemistry	Medicinal Chemistry	3.12
3	3009	Advanced Organic Chemistry	2020-01-15	1002	Prof. Dana Cohen	Chemistry	Organic Chemistry	2.97
4	3011	Organic Synthesis	2021-11-30	1002	Prof. Dana Cohen	Chemistry	Organic Chemistry	2.91
5	3013	Organic Reaction Mechanisms	2018-12-12	1002	Prof. Dana Cohen	Chemistry	Organic Chemistry	2.8
6	3008	Organic Chemistry for Beginners	2018-05-10	1002	Prof. Dana Cohen	Chemistry	Organic Chemistry	2.68
7	3015	Green Chemistry	2022-05-05	1002	Prof. Dana Cohen	Chemistry	Sustainable Chemistry	2.55
8	3010	Laboratory Techniques in Organic Chemistry	2019-08-20	1002	Prof. Dana Cohen	Chemistry	Organic Chemistry	2.24
9	3061	Time Series Analysis	2020-09-15	1008	Prof. Linda Cohen	Statistics	Time Series	3.74
10	3063	Multivariate Analysis	2018-05-20	1008	Prof. Linda Cohen	Statistics	Multivariate Analysis	3.72
11	3060	Biostatistics	2019-07-10	1008	Prof. Linda Cohen	Statistics	Biostatistics	3.29
12	3056	Introduction to Statistics	2019-08-18	1008	Prof. Linda Cohen	Statistics	Descriptive Statistics	3.29
13	3058	Statistical Methods	2018-02-28	1008	Prof. Linda Cohen	Statistics	Statistical Methods	3.18
14	3057	Probability and Statistics	2020-11-05	1008	Prof. Linda Cohen	Statistics	Probability	3
15	3062	Experimental Design	2021-11-30	1008	Prof. Linda Cohen	Statistics	Experimental Design	2.93
16	3059	Data Analysis	2021-03-25	1008	Prof. Linda Cohen	Statistics	Data Analysis	2.74

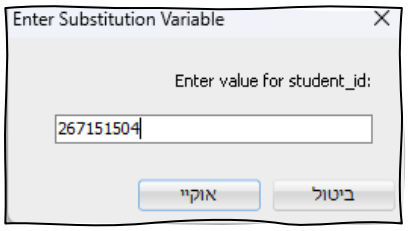
בס"ד

שאילתה 2: שליפת ביקורות של סטודנט עבור ספרים

מטרת השאילתה:
להחזיר רשימת ספרים שביקורותיהם נכתבו על ידי סטודנט מסוים, תוך הצגת פרטים כמו ממוצע דירוגים כללי, דירוג אישי של הסטודנט וביקורת עד 4000 תווים. פעולה זו מסייעת לנתח את דפוסי הביקורות של הסטודנט.

הסבר השאילתה:
השאילתה מחברת את טבלאות הספרים והביקורות, תוך סינון לפי מזהה הסטודנט. היא מחשבת את ממוצע הדירוגים הכללי של הספרים ומשלבת את הביקורת האישית שנכתבה על ידי הסטודנט.
הנתונים מסודרים לפי ממוצע הדירוגים ושם הספר.

```oracle_plsql
ACCEPT student_id NUMBER PROMPT 'Enter Student ID: ';

SELECT
    B.Book_id,
    B.Book_name,
    ROUND(AVG(R.Rating), 2) AS Avg_Rating,
    SR.Rating AS Student_Rating,
    DBMS_LOB.SUBSTR(SR.Review_text, 4000, 1) AS Student_Review
FROM
    Books B
JOIN
    Reviews SR ON B.Book_id = SR.Book_id
LEFT JOIN
    Reviews R ON B.Book_id = R.Book_id
WHERE
    SR.Reviewer_id = &student_id
GROUP BY
    B.Book_id, B.Book_name, SR.Rating, DBMS_LOB.SUBSTR(SR.Review_text, 4000, 1)
ORDER BY
    Avg_Rating DESC, B.Book_name ASC;
```



	BOOK_ID	BOOK_NAME	AVG_RATING	STUDENT_RATING	STUDENT_REVIEW
1	3467	Ruby on Rails for Web Development	3.25	4	This was an excellent read. I really liked it.
2	3005	Probability Theory	3.22	4	The characters were great and very well-developed.
3	3618	Go Programming for Web	3.13	1	The story was predictable and didn't offer any surprises.
4	3045	Molecular Biology	3.03	1	I couldn't relate to the characters or the story. It just wasn't my cup of tea.
5	3155	Digital Marketing	2.9	4	The book was fantastic! I highly recommend it.
6	3016	Principles of Physics	2.85	5	An unforgettable story. Highly recommend!
7	3441	Advanced TypeScript Programming	2.77	3	The story was solid and kept me entertained.
8	3501	Advanced C++ Programming	2.53	3	The characters were good, but the plot was just okay.
9	3319	Responsive Web Design with HTML and CSS	2.37	3	The book was entertaining and enjoyable.

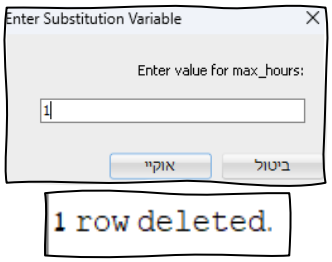
שאילתה 3: מחיקת בקשות חדרים חורגות

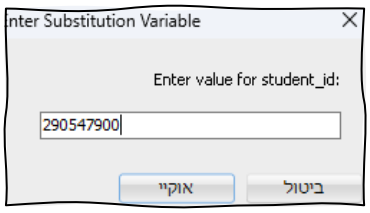
מטרת השאילתה:
למחוק בקשות לחדרים של סטודנט מסוים שהזמן המצטבר של הבקשות המאושרות שלו חורג מהמגבלה המוגדרת. המטרה היא לוודא שהשימוש במשאבים של הספרייה, כמו חדרי לימוד, יהיה הוגן ומבוקר.

הסבר השאילתה:
השאילתה מקבלת פרמטרים של מזהה סטודנט ומספר שעות מאושרות מרביות היא מזהה בקשות מאושרות של הסטודנט שהזמן הכולל שלהן חורג מהמגבלה המוגדרת באמצעות חיבור תתי-שאילתות שמזהות את הבקשות והזמן המצטבר שלהן. הבקשות מתאריכים עתידיים נבדקות לפי תנאים המוגדרים ב-HAVING. לאחר מכן, הבקשות המתאימות נמחקות מטבלת RequestsRooms.

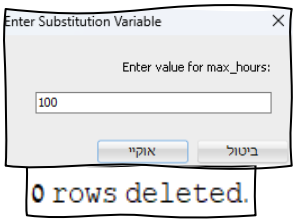
```oracle_plsql
ACCEPT student_id NUMBER PROMPT 'Enter Student ID: ';
ACCEPT max_hours NUMBER PROMPT 'Enter Maximum Approved Hours: ';

DELETE FROM RequestsRooms
WHERE (Student_id, Request_time) IN (
    SELECT
        RR.Student_id,
        RR.Request_time
    FROM
        RequestsRooms RR
    LEFT JOIN
        RequestsRooms PreviousRR
    ON
        RR.Student_id = PreviousRR.Student_id
        AND PreviousRR.Start_time < RR.Start_time
        AND PreviousRR.Status = 'approved'
    WHERE
        RR.Status = 'approved'
        AND RR.Start_time >= TRUNC(SYSDATE + 1)
        AND RR.Student_id = &student_id
    GROUP BY
        RR.Student_id,
        RR.Request_time,
        RR.Start_time,
        RR.Duration_hours
    HAVING
        NVL(SUM(PreviousRR.Duration_hours), 0) + RR.Duration_hours > &max_hours
);
```

אפשר לראות בצילומי מסך של הטסט שיש לאותו סטודנט רק בקשה אחת עתידית לחדר לימוד. כאשר הגבלתי לשעה אחת אז היא נמחקה וכאשר הגבלתי ל-100 שעות במצטבר אז היא לא נמחקה





	FUTURE_APPROVED_REQUESTS
1	1



בס"ד

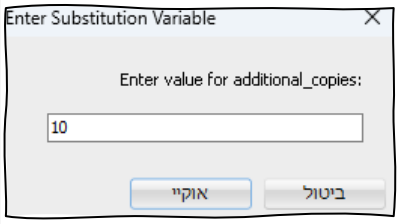
שאילתה 4: עדכון כמות ספרים זמינים

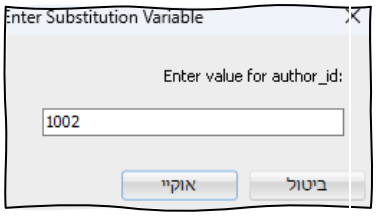
מטרת השאילתה:
לעדכן את כמות העותקים של ספרים לפי מספר שנוסף באמצעות פרמטר. השאילתה מתמקדת בספרים של מחבר מסוים עם דירוג מעל ממוצע הקטגוריה שלהם. פעולה זו מבטיחה התאמת מלאי הספרים לביקוש המשתמשים.

הסבר השאילתה:
השאילתה מזהה ספרים העונים לקריטריונים של מחבר ודירוג מעל ממוצע הקטגוריה. לאחר זיהוי הספרים, מספר העותקים הזמינים מתעדכן על ידי הוספת הערך שצוין בפרמטר. הפעולה מתמקדת בספרים פופולריים כדי להתאים את המלאי לדרישות המשתמשים.

```oracle_plsql
ACCEPT author_id NUMBER PROMPT 'Enter Author ID: ';
ACCEPT additional_copies NUMBER PROMPT 'Enter Number of Copies to Add: ';

UPDATE BookStatuses BS
SET
    BS.Number_of_copies = BS.Number_of_copies + &additional_copies
WHERE
    BS.Status = 'available'
    AND BS.Book_id IN (
        SELECT
            B.Book_id
        FROM
            Books B
        JOIN
            BookAuthors BA ON B.Author_id = BA.Author_id
        JOIN
            BookCategories BC ON B.Category_id = BC.Category_id
        LEFT JOIN
            Reviews R ON B.Book_id = R.Book_id
        WHERE
            BA.Author_id = &author_id
        GROUP BY
            B.Book_id, B.Book_name, BC.Category_id, BC.Category, BC.Subcategory
        HAVING
            NVL(ROUND(AVG(R.Rating), 2), 0) > (
                SELECT
                    NVL(ROUND(AVG(R2.Rating), 2), 0)
                FROM
                    Books B2
                JOIN
                    BookCategories BC2 ON B2.Category_id = BC2.Category_id
                LEFT JOIN
                    Reviews R2 ON B2.Book_id = R2.Book_id
                WHERE
                    BC2.Category_id = BC.Category_id
            )
    );
```

	BOOK_ID	STATUS	NUMBER_OF_COPIES
1	3009	available	10
2	3011	available	10
3	3012	available	11





3 rows updated.

בטסט אפשר את כמות הספרים הזמינים בספרייה של הסופר עם המזהה 1002 בקטגוריות שהספרים שלו פופולריים בהן

	BOOK_ID	STATUS	NUMBER_OF_COPIES
1	3009	available	20
2	3011	available	20
3	3012	available	21

אחרי העדכון הודפס ש-3 שורות עודכנו ואפשר לראות שכמות העותקים הזמינים בספרייה של הספרים הנ"ל גדלו ב-10 כל אחד.

אילוצים:

אילוץ 1: פורמט תקין לטלפונים בטבלת Students ובטבלת Librarians

האילוץ check_phone_format מבטיח שדה הטלפון יעמוד בפורמט תקין. הפורמט הנדרש הוא מספר שמתחיל ב-"05", אחריו ספרה נוספת, מקף, ושבע ספרות נוספות (לדוגמה: 050-1234567). אילוץ זה מחזק את נכונות הנתונים ומבטיח אחידות בשדה הטלפון בשתי הטבלאות.

```
ALTER TABLE Students
ADD CONSTRAINT check_student_phone_format CHECK (
    REGEXP_LIKE(Phone, '^05[0-9]-[0-9]{7}$')
);
```

```
ALTER TABLE Librarians
ADD CONSTRAINT check_librarian_phone_format CHECK (
    REGEXP_LIKE(Phone, '^05[0-9]-[0-9]{7}$')
);
```

```
Error starting at line : 1 in command -
INSERT INTO Students (Student_id, Student_first_name, Student_last_name, Email, Phone, Birth_date,
Degree, Start_date) VALUES (278377995, 'Maya', 'Menahem', 'maya_menahem06jct.ac.il', '055-8855072',
TO_DATE('2006-01-07', 'YYYY-MM-DD'), 'MA in art', TO_DATE('2022-10-13', 'YYYY-MM-DD'))
Error report -
ORA-02290: אילוץ בדיקה (NEW_LIBRARY.CHECK_STUDENT_EMAIL_VALIDITY) הופר
```

```
Error starting at line : 1 in command -
INSERT INTO Librarians (Librarian_id, Librarian_first_name, Librarian_last_name, Email,
Phone, Librarian_birth_date, Hire_date) VALUES (200885937, 'Ron', 'Friedman', 'ron_friedman@gmail.co', '051-3613182',
TO_DATE('1984-05-07', 'YYYY-MM-DD'), TO_DATE('2010-02-27', 'YYYY-MM-DD'))
Error report -
ORA-02290: אילוץ בדיקה (NEW_LIBRARY.CHECK_LIBRARIAN_EMAIL_VALIDITY) הופר
```

אפשר לראות כאן שגם אצל הסטודנטים וגם אצל הספרנים לא התאפשר להכניס את הנתונים הנ"ל בגלל האילוצים.

אילוץ 2: פורמט תקין למיילים בטבלת Students ובטבלת Librarians

האילוץ check_valid_email מבטיח שדה המייל יעמוד בפורמט תקין. המייל חייב להתחיל באותיות או מספרים, יכול לכלול נקודות, מקפים ותווים מיוחדים מסוימים, ולאחר מכן סימן "@", שם דומיין חוקי, וסיומת המתאימה ל-co.il או com (לדוגמה: example@mail.com). אילוץ זה מבטיח שהמיילים הנכנסים תקינים ושמישים לצורכי הספרייה.

```
ALTER TABLE Students
ADD CONSTRAINT check_student_email_validity CHECK (
    REGEXP_LIKE(
        Email,
        '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+(\.ac\.il|\.co\.il|\.com)$'
    )
);
```

```
ALTER TABLE Librarians
ADD CONSTRAINT check_librarian_email_validity CHECK (
    REGEXP_LIKE(
        Email,
        '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+(\.ac\.il|\.co\.il|\.com)$'
    )
);
```

```
Error starting at line : 1 in command -
INSERT INTO Students (Student_id, Student_first_name, Student_last_name, Email, Phone, Birth_date,
Degree, Start_date) VALUES (278377995, 'Maya', 'Menahem', 'maya_menahem06@jct.ac.il', '100-8855072',
TO_DATE('2006-01-07', 'YYYY-MM-DD'), 'MA in art', TO_DATE('2022-10-13', 'YYYY-MM-DD'))
Error report -
ORA-02290: אילוץ בדיקה (NEW_LIBRARY.CHECK_STUDENT_PHONE_FORMAT) הופר
```

```
Error starting at line : 1 in command -
INSERT INTO Librarians (Librarian_id, Librarian_first_name, Librarian_last_name, Email,
Phone, Librarian_birth_date, Hire_date) VALUES (200885938, 'Ron', 'Friedman', 'ron_friedman@gmail.com',
'051-102030', TO_DATE('1984-05-07', 'YYYY-MM-DD'), TO_DATE('2010-02-27', 'YYYY-MM-DD'))
Error report -
ORA-02290: אילוץ בדיקה (NEW_LIBRARY.CHECK_LIBRARIAN_PHONE_FORMAT) הופר
```

אפשר לראות כאן שגם אצל הסטודנטים וגם אצל הספרנים לא התאפשר להכניס את הנתונים הנ"ל בגלל האילוצים.

אילוץ 3: דירוג תקין בטבלת Reviews

האילוץ check_valid_rating מוודא שהדירוגים בטבלת הביקורות יהיו מספרים שלמים בין 1 ל-5. דירוגים אלה משמשים למדידת איכות הספרים ונכונותם מבטיחה ניתוח נתונים איכותי ומדויק.

```
ALTER TABLE Reviews
ADD CONSTRAINT check_valid_rating CHECK (
    Rating BETWEEN 1 AND 5 AND MOD(Rating, 1) = 0
);
```

```
Error starting at line : 1 in command -
INSERT INTO Reviews (Book_id, Reviewer_id, Review_text, Rating, Review_date, Review_title) VALUES
(3001, 379023898, 'The author made a good effort with this book.',
6, TO_DATE('2024-02-27', 'YYYY-MM-DD'),'Good Effort')
Error report -
ORA-02290: אילוץ בדיקה (NEW_LIBRARY.CHECK_VALID_RATING) הופר
```

אפשר לראות כאן שלא התאפשר להכניס את הנתונים הנ"ל בגלל האילוץ של Rating מ-1 עד 5.

אילוץ 4: מספר האנשים בבקשות חדרים

האילוץ check_valid_people_count מוודא שמספר האנשים בבקשות חדרים יעמוד בין 1 ל-15 בלבד. אילוץ זה נועד להבטיח שנעשה שימוש נכון ומדויק במשאבי חדרי הלימוד בספרייה.

```
ALTER TABLE RequestsRooms
ADD CONSTRAINT check_valid_people_count CHECK (
    Number_of_people BETWEEN 1 AND 15
);
```

```
Error starting at line : 1 in command -
INSERT INTO RequestsRooms (Student_id, Request_time, Start_time, Duration_hours,
Number_of_people, Board, Screen, Handled_Librarian, Assigned_room, Status) VALUES
(334338522, TO_DATE('2024-10-01 00:05:11', 'YYYY-MM-DD HH24:MI:SS'),
TO_DATE('2024-11-02 09:00:00', 'YYYY-MM-DD HH24:MI:SS'), 1.0, 16, 'N', 'N', 228565115, 3, 'approved')
Error report -
ORA-02290: אילוץ בדיקה (NEW_LIBRARY.CHECK_VALID_PEOPLE_COUNT) הופר
```

אפשר לראות כאן שלא התאפשר להכניס את הנתונים הנ"ל בגלל באילוץ של ההגבלה עד 15 אנשים בחדר בבקשה לחדרי לימוד.

אילוץ 5: ערך ברירת מחדל לעמודת סטטוס בבקשות חדרים

האילוץ הדיפולטיבי על עמודת Status בטבלת RequestsRooms מגדיר שערך ברירת המחדל עבור סטטוס הבקשה יהיה waiting, במידה ולא הוזן ערך אחר בזמן ההכנסה לטבלה. הגדרה זו נועדה להבטיח שכל בקשה תתחיל במצב ראשוני ברור כברירת מחדל, מה שמקל על ניהול תהליך הטיפול בבקשות ומצמצם סיכוי לטעויות או ערכים חסרים במערכת.

```
ALTER TABLE RequestsRooms
MODIFY Status DEFAULT 'waiting';
```

```
INSERT INTO RequestsRooms (
  Student_id, Request_time, Start_time, Duration_hours,
  Number_of_people, Board, Screen, Handled_Librarian, Assigned_room
)
VALUES (
  278377999, TO_DATE('20/11/2024', 'DD/MM/YYYY'), TO_DATE('21/11/2024', 'DD/MM/YYYY'),
  2.5, 5, 'Y', 'N', NULL, NULL
);

SELECT * FROM RequestsRooms WHERE Student_id = 278377999 and Request_time=TO_DATE('20/11/2024', 'DD/MM/YYYY');
```

Script Output | Query Result

SQL | All Rows Fetched: 1 in 0.004 seconds

	STUDENT_ID	REQUEST_TIME	START_TIME	DURATION_HOURS	NUMBER_OF_PEOPLE	BOARD	SCREEN	HANDLED_LIBRARIAN	ASSIGNED_ROOM	STATUS
1	278377999	20/11/2024 00:00	21/11/2024 00:00	2.5	5	Y	N	(null)	(null)	waiting

אפשר לראות כאן שעל אף שלא הוכנס בשאילתה ערך לסטטוס, נוצרה רשומה עם ערך דפולטיבי waiting.

אילוץ 6: חיוב ביוגרפיה של סופר

האילוץ על עמודת Biography בטבלת BookAuthors מוודא שלא ניתן להכניס או לעדכן רשומה כך שעמודה זו תכיל ערך NULL. אילוץ זה נועד להבטיח שכל מחבר בטבלה יכלול ביוגרפיה מלאה כחלק ממידע חיוני המאפשר שימוש במערכת בצורה תקינה. כך נשמרת איכות המידע ומונעת חוסר עקביות בנתונים.

```
ALTER TABLE BookAuthors
MODIFY Biography  NOT NULL;
```

```
Error starting at line : 1 in command -
INSERT INTO BookAuthors (
  Author_id, Author_academic_title, Author_first_name,
  Author_last_name, Author_birth_date, Nationality, Biography

VALUES (
  1, 'Dr.', 'John', 'Doe', TO_DATE('1980-01-01', 'YYYY-MM-DD'), 'American', NULL
)
Error at Command Line : 6 Column : 79
Error report -
SQL Error: ORA-01400: לא ניתן להכניס ערך NULL לתוך ("NEW_LIBRARY"."BOOKAUTHORS"."BIOGRAPHY")
01400. 00000 - "cannot insert NULL into (%s)"
*Cause:    An attempt was made to insert NULL into previously listed objects.
*Action:   These objects cannot accept NULL values.
```

אפשר לראות כאן שלא התאפשר להכניס את הסופר הנ"ל מכיוון שאין לו ביוגרפיה.