בס"ד

מיני פרויקט בבסיסי נתונים

ניהול ספריה באוניברסיטה

מגיש: עדן ישי כהן

תוכן עניינים:

הקדמה:
המערכת המוצגת בפרויקט זה היא מערכת ניהול ספרייה אוניברסיטאית, שנועדה לייעל את תהליכי הניהול והמעקב אחר משאבים, שירותים ובקשות של משתמשי הספרייה. מערכת זו כוללת ממשק לניהול קטגוריות ספרים, מחברים, לומדים, השאלות, ביקורות, חדרי לימוד, ובקשות מיוחדות.

שימוש המערכת:
המערכת מאפשרת לספרנים לנהל בצורה פשוטה ואפקטיבית את המידע הקשור למשאבים השונים של הספרייה, תוך מתן אפשרות ללומדים לבצע השאלות, ביקורות, ובקשות לחדרי לימוד. המערכת מספקת כלים למעקב אחר זמינות ספרים, טיפול בבקשות מיוחדות, והערכת משוב ממשתמשים באמצעות ביקורות. כל טבלה מקושרת לטבלאות אחרות בצורה המאפשרת קשרים מורכבים המבטיחים את עקיבות המידע.

מטרות המערכת:
המערכת נועדה לספק מענה לצרכים הבאים:

- שיפור חוויית המשתמש של הלומדים והספרנים.
- יעילות גבוהה יותר בניהול משאבי הספרייה.
- שמירה על עקיבות וניהול נכון של השאלות, חדרים, ובקשות.
- איסוף וניתוח משוב משתמשים למטרות שיפור השירות.

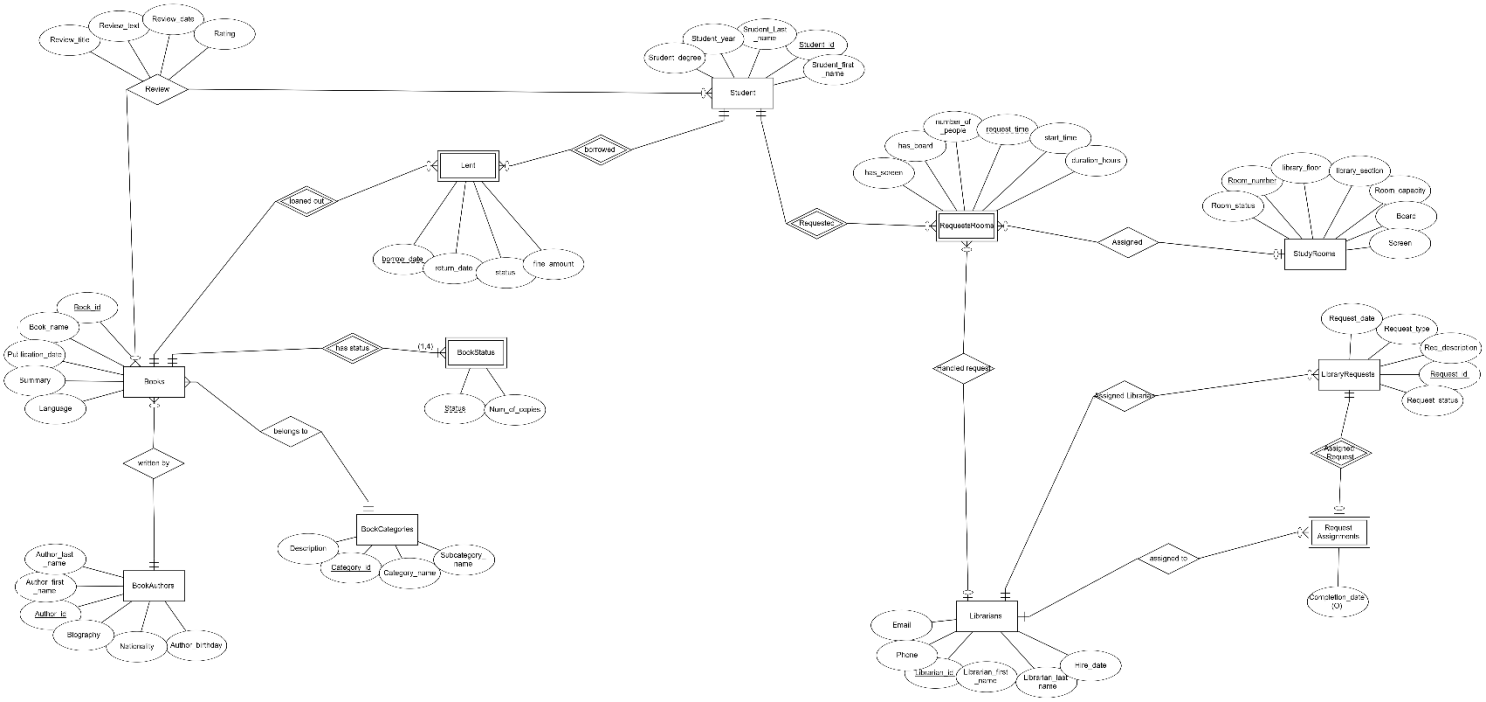
המערכת מאפשרת יצירת דוחות וניתוחים שונים עבור מנהלי הספרייה לצורך קבלת החלטות ושיפור מתמיד של תפקוד המערכת.

תיאור המערכת
המערכת כוללת מספר טבלאות עיקריות המייצגות ישויות שונות, ביניהן:

- **קטגוריות ספרים** :(**BookCategories**) טבלה המכילה את פרטי הקטגוריות והקטגוריות המשנה של הספרים בספרייה, כולל תיאור מפורט.
- **מחברים** :(**BookAuthors**) טבלה המרכזת את פרטי המחברים, כולל שם פרטי ושם משפחה, תאריך לידה, לאום, וביוגרפיה.
- **ספרים** :(**Books**) טבלה המתארת את פרטי הספרים, כולל שם הספר, תאריך פרסום, קטגוריה ומחבר, שפה, וסיכום.
- **סטטוס ספרים** :(**BookStatuses**) טבלה שמציגה את מצב הזמינות של ספרים שונים, מספר עותקים וסטטוס כללי.
- **לומדים** :(**Students**) טבלה הכוללת את פרטי הלומדים הרשומים בספרייה, כולל פרטים אישיים ומידע ליצירת קשר.
- **השאלות** :(**Lent**) טבלה שמכילה מידע על השאלת ספרים, כולל פרטי הסטודנט, הספר, ותאריכי השאלה והחזרה.
- **ביקורות** :(**Reviews**) טבלה המתעדת ביקורות של לומדים על ספרים שונים, כולל כותרת הביקורת, דירוג, ותאריך פרסום.
- **חדרי לימוד** :(**StudyRooms**) טבלה שמרכזת מידע על חדרי הלימוד בספרייה, כולל מספר החדר, קומה, אזור, ותכולת החדר.
- **בקשות לחדרים** :(**RequestsRooms**) טבלת לטיפול בבקשות לשריון חדרי לימוד עבוד סטודנט בטווח זמנים, עם אפשרות לציין ספרן שטיפל בבקשה, סטטוס, ואם בסטטוס אושר אז גם חדר מוקצה.
- **בקשות ספרייה** (**LibraryRequests**) ו**בקשות ספרים** :(**BookRequests**) טבלאות המנהלות בקשות כלליות לספרייה.

תרשימים ראשוניים:

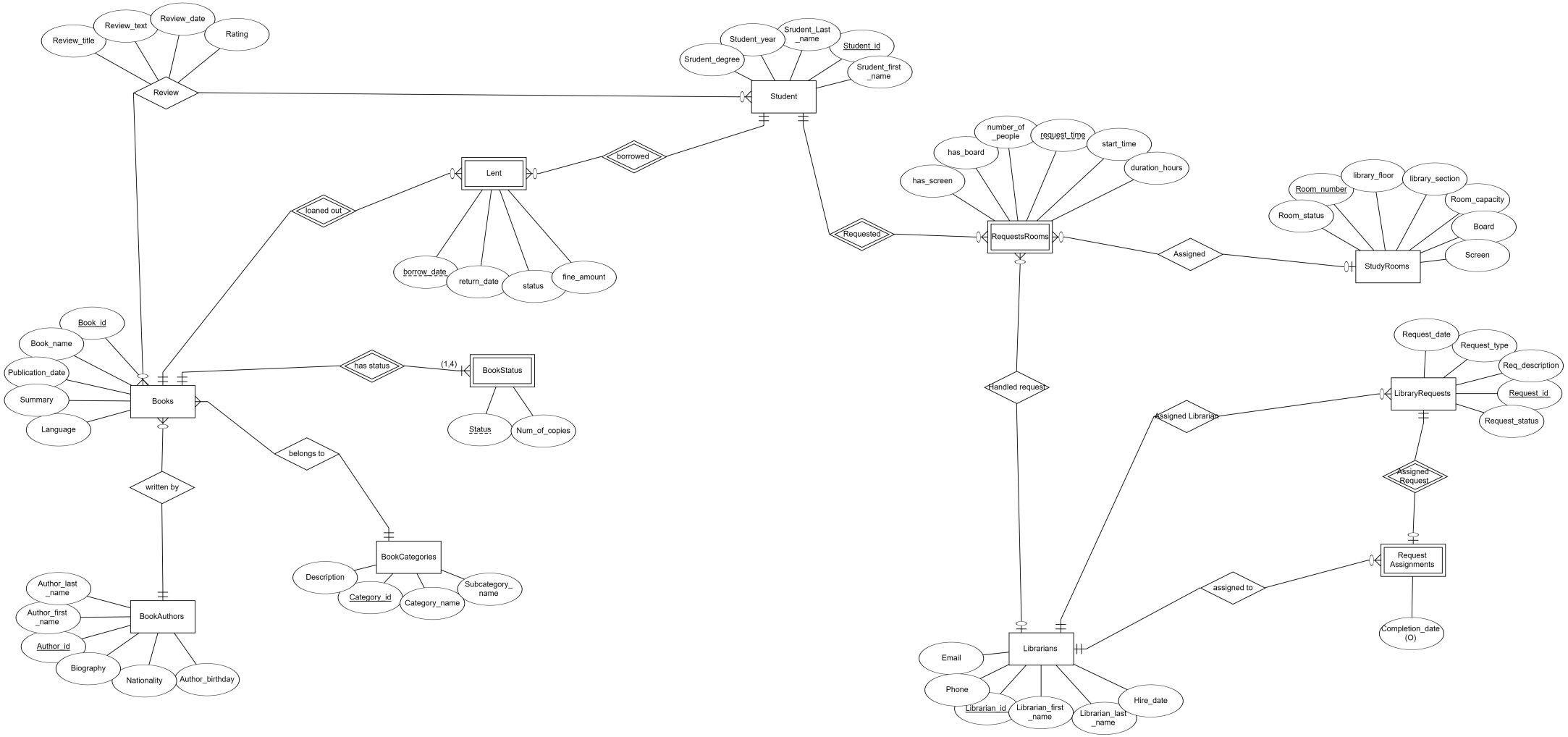
ERD: (אפשר לראות את התמונה בבירור בעמוד הבא)



הסבר תרשים ה-ERD:

התרשים שתכננתי עבור המערכת מציג את הקשרים בין הישויות המרכזיות ואת המידע השמור בכל אחת מהן:

- **ספרים (Books):** הישות המרכזית שמייצגת את הספרים בספרייה. התכונות כוללות מזהה ספר (Book_id), שם הספר, תאריך פרסום, מזהה קטגוריה, מזהה מחבר, שפה, וסיכום.
- **סופרים (BookAuthors):** ישות שמייצגת את המחברים של הספרים. התכונות כוללות מזהה מחבר (Author_id), שם פרטי, שם משפחה, תאריך לידה, לאום, וביוגרפיה.
- **קטגוריות ספרים (BookCategories):** ישות שמייצגת את הקטגוריות בהן מסווגים הספרים, כולל שם קטגוריה, תיאור וקטגוריות משנה.
- **סטטוס ספרים (BookStatuses):** ישות שמתעדת את מצב הספרים, כגון זמינות ומספר העותקים.
- **סטודנטים (Students):** ישות הכוללת מידע על הלומדים המשתמשים בשירותי הספרייה, עם תכונות כגון מזהה לומד, שם פרטי ושם משפחה, אימייל, טלפון ותאריך לידה.
- **השאלות ספרים (Lent)**: ישות שמרכזת את כל היסטוריית ההשאלות של הספרים ואת סטטוס ההשאלה, את מצב הספר שהושאל ואת הקנס אם יש על ההשאלה הנוכחית.
- **חדרי לימוד (StudyRooms):** ישות שמתארת את חדרי הלימוד בספרייה, כולל מספר חדר, קומה, אזור ותכולה.
- **בקשות לחדרים (RequestsRooms):** ישות שמרכזת בקשות לשריון חדרי לימוד, כולל פרטי הלומד המבקש, זמן הבקשה, זמן התחלה, משך, מספר המשתתפים, ותכונות כמו לוח ומסך.
- **בקשות ספרייה: (LibraryRequests):** ישות שמתארת את הבקשות שונות בספרייה, כולל סוגי הבקשות, סטטוס, ותיאור והספרן שיצר את הבקשה
- **מבצע הבקשה (RequestAssignments):** מתארת את הקשר בין בקשות הספרייה לבין הספרנים המטפלים בהן. התכונות של ישות זו כוללות מזהה בקשה (Request_id), מזהה ספרן (Librarian_id)ותאריך הטיפול בבקשה – במקרה שהבקשה הסתיימה או נסגרה.


Review_title
Review_text
Review_date
Rating
Review
Student
Srudent_degree
Student_year
Srudent_Last _name
Student_id
Srudent_first _name
Lent
borrowed
loaned out
borrow_date
return_date
status
fine_amount
Requested
RequestsRooms
has_board
number_of _people
request_time
start_time
duration_hours
has_screen
Assigned
StudyRooms
Room_number
library_floor
library_section
Room_capacity
Room_status
Board
Screen
Book_id
Book_name
Publication_date
Summary
Language
Books
has status
(1,4)
BookStatus
Status
Num_of_copies
belongs to
written by
BookCategories
Description
Category_id
Category_name
Subcategory_ name
BookAuthors
Author_last _name
Author_first _name
Author_id
Biography
Nationality
Author_birthday
Handled request
Assigned Librarian
LibraryRequests
Request_date
Request_type
Req_description
Request_id
Request_status
Assigned Request
Request Assignments
assigned to
Completion_date (O)
Librarians
Email
Phone
Librarian_id
Librarian_first _name
Librarian_last _name
Hire_date

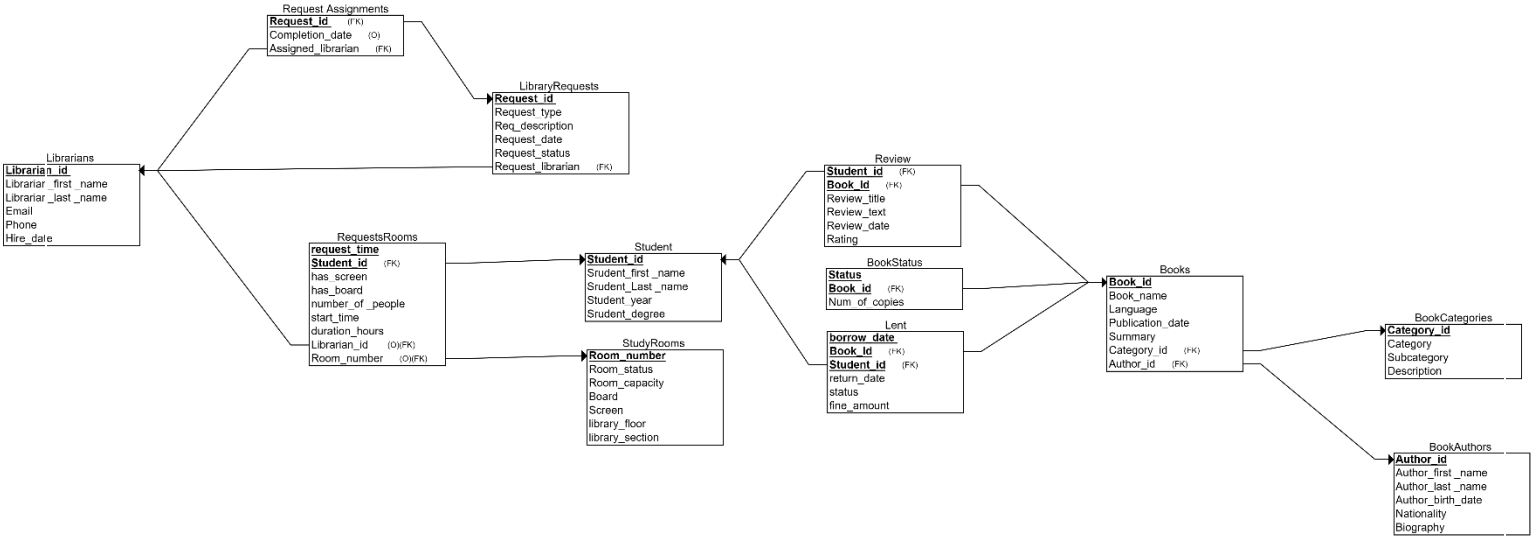
קשרים:

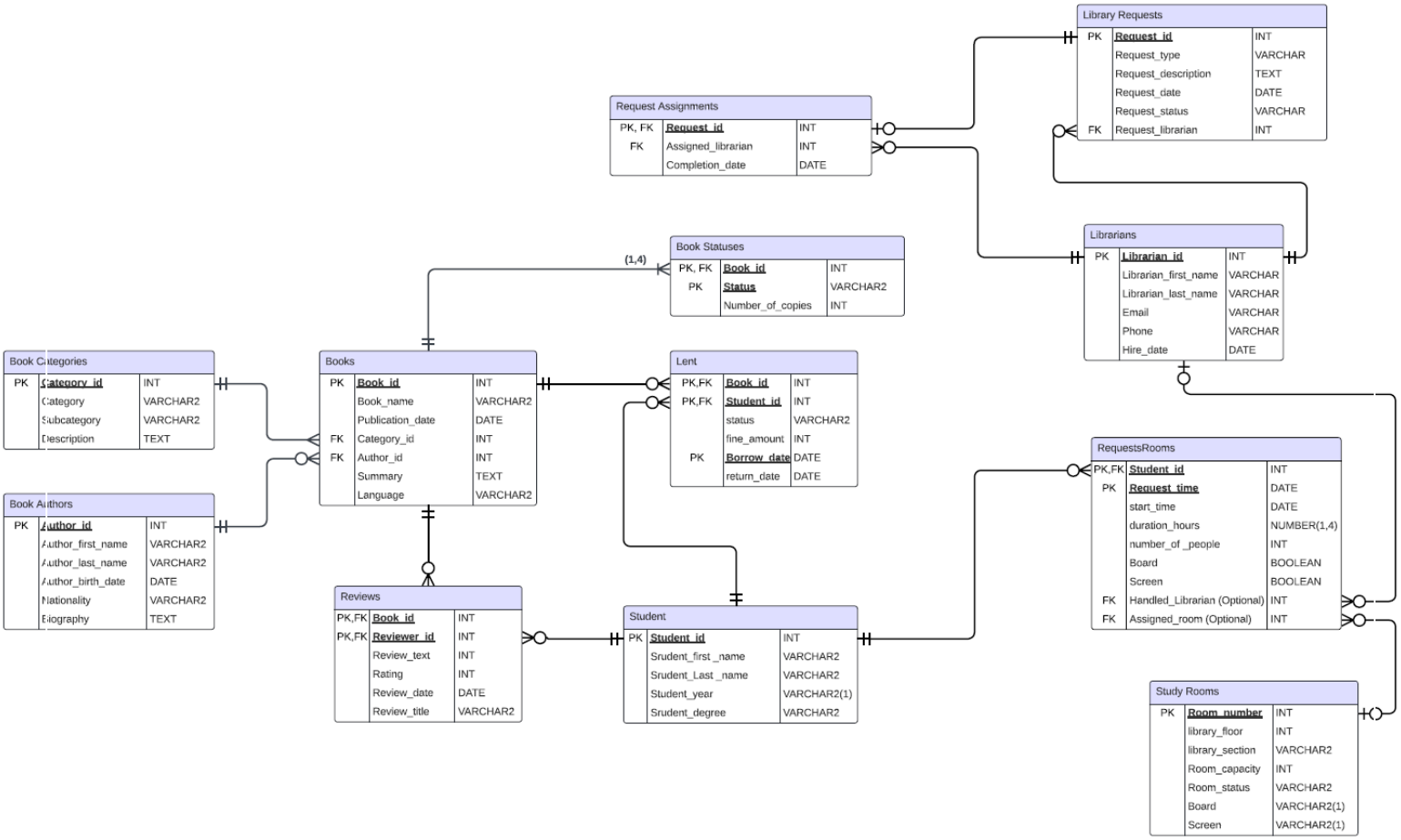
- **"Written By"**: מתאר מערכת יחסים של "אחד בלבד" ל-"אפס או יותר" בין הישות של **ספרים (Books)** לישות של **סופרים .(BookAuthors)** משמעות הקשר היא שכל ספר בספרייה חייב להיות משויך למחבר אחד בלבד, כלומר, לא ייתכן ספר ללא מחבר או עם יותר ממחבר אחד. מצד שני, עבור הישות **סופר ,**ייתכן מצב שבו אין ספרים כלל שהוא כתב הנמצאים בספרייה, או לחלופין, שהוא המחבר של מספר ספרים שונים שנמצאים במערכת הספרייה.
- "**Belong to"**: מתאר מערכת יחסים של "אחד בלבד" ל-"רבים" בין הישות של **ספרים (Books)** לישות של **קטגוריות ספרים .(BookCategories)** המשמעות היא שכל ספר בספרייה חייב להשתייך לקטגוריה אחת בלבד, אך קטגוריה יכולה לכלול מספר ספרים שונים. הקשר הזה מאפשר לסווג את הספרים לפי הקטגוריות השונות שאליהן הם שייכים, ומבטיח שכל ספר מסווג בקפידה בתוך הקטגוריה המתאימה לו.
- "**Lent**": מתאר מערכת יחסים של "אפס או יותר" ל-"אפס או יותר" בין הישות של **ספרים (Books)** לישות של **סטודנטים ,(Students)** כלומר, ספר בספרייה יכול להיות מושאל לאף סטודנט או למספר סטודנטים שונים לאורך זמן, ומנגד, סטודנט יכול לשאול אף ספר או מספר ספרים שונים מהספרייה. לקשר זה יש תכונות המאפשרות לנהל את המידע הקשור להשאלות, ביניהן :**תאריך השאלה – (Borrow_date)** התאריך שבו הספר הושאל ,**תאריך החזרה – (Return_date)** התאריך שבו הספר צריך להיות מוחזר או הוחזר בפועל ,**סטטוס – (Status)**המצב הנוכחי של הספר בזמן ההשאלה (כגון "בהשאלה", "מוחזר"), ו-**סכום קנס – (Fine_amount)** סכום קנס שניתן להחיל במקרה של איחור בהחזרת הספר. תכונות אלו מבטיחות מעקב מלא אחר תהליך ההשאלה, ניהול נכון של הספרים, ועמידה בתנאי ההשאלה.
- "**hasStatus**": מתאר מערכת יחסים של "אחד או עד ארבעה" ל-"אפס או יותר" בין הישות של **ספרים (Books)** לישות של **סטטוס ספרים .(BookStatuses)** המשמעות היא שלכל ספר בספרייה חייב להיות לפחות סטטוס אחד, ועד ארבעה סטטוסים שונים המתארים את מצבו (כגון "זמין", "בהשאלה", "נפגע", "אבוד"). מנגד, סטטוס מסוים יכול להיות משויך לאף ספר או למספר ספרים.
- **"Loaned Out":** מתאר מערכת יחסים של "אחד בלבד" ל-"אפס או יותר" בין הישות של **ספרים (Books)** לישות החלשה של **השאלות (Lent).** המשמעות היא שכל ספר בספרייה יכול להיות מושאל מספר פעמים לאורך זמן או שלא יושאל כלל, בעוד שכל רשומת השאלה בישות `Lent` חייבת להיות משויכת לספר אחד בלבד. קשר זה מאפשר מעקב מלא אחר היסטוריית ההשאלות לכל ספר, כולל פרטי תאריך השאלה (Borrow_date), תאריך החזרה (Return_date), סטטוס ההשאלה (Status), וסכום הקנס (Fine_amount) במקרה של איחור או נזק לספר.
- **"Borrowed":** מתאר מערכת יחסים של "אחד בלבד" ל-"אפס או רבים" בין הישות של **סטודנטים (Students)** לישות החלשה של **השאלות (Lent).** המשמעות היא שכל סטודנט יכול לשאול מספר ספרים לאורך זמן או שלא לשאול כלל, בעוד שכל רשומת השאלה בישות `Lent` חייבת להיות משויכת לסטודנט אחד בלבד. קשר זה מאפשר לספרייה לנהל ולעקוב אחר פעילות ההשאלות של כל סטודנט, כולל פרטים על תאריכי ההשאלה וההחזרה, מצב הספר בזמן ההשאלה, וסכום הקנס במקרים של איחורים או נזקים.
- " **Requested** ": מתאר מערכת יחסים של "אפס או יותר" ל-"אחד בלבד" בין הישות של **סטודנטים (Students)**לישות של **בקשות לחדרים .(RequestRooms)** המשמעות היא שסטודנט יכול לבצע אף בקשה או מספר בקשות להזמנת חדרי לימוד בספרייה, בעוד שכל בקשה לחדר לימוד חייבת להיות משויכת לסטודנט אחד בלבד. קשר זה מאפשר מעקב אחר בקשות חדרי הלימוד שנעשו על ידי הסטודנטים, כולל פרטי זמן הבקשה, זמן התחלה, משך זמן השימוש, ומספר המשתתפים שמבקש הסטודנט.
- "**Assigned**": מתאר מערכת יחסים של "אפס או יותר" ל-"אפס או אחד" בין הישות של **חדרי לימוד (StudyRooms)**לישות של **בקשות לחדרים .(RequestRooms)** המשמעות היא שחדר לימוד יכול להיות מוקצה לאף בקשה או למספר בקשות שונות לאורך זמן, בעוד שכל בקשה יכולה להיות משויכת לחדר לימוד אחד לכל היותר. קשר זה מאפשר מעקב אחר השיבוץ של חדרי לימוד להזמנות ומוודא ניהול תקין של הקצאת החדרים.
- "**Assigned Librarian**": מתאר מערכת יחסים של "אפס או יותר" ל-"אפס או אחד" בין הישות של **ספרנים (Librarians)**לישות של **בקשות לחדרים .(RequestRooms)** המשמעות היא שספרן יכול לטפל באף בקשה או במספר בקשות לאורך זמן, בעוד שכל בקשה יכולה להיות משויכת לספרן אחד לכל היותר. קשר זה מאפשר מעקב אחר הטיפול בבקשות חדרי לימוד ומבטיח כי ישנו ספרן אחראי על כל בקשה שטופלה, לשם מתן שירות טוב ומעקב אחר הבקשות.
- "**AssignedLibrarian**": מתאר מערכת יחסים של "אחד בלבד" ל-"אפס או אחד" בין הישות של **משימות בקשות (RequestAssignments)** לישות של **בקשות ספרייה .(LibraryRequests)** המשמעות היא שכל רשומה בטבלה **RequestAssignments**חייבת להיות משויכת לבקשת ספרייה אחת בלבד, בעוד שבקשת ספרייה יכולה להיות משויכת לאף משימה או למשימה אחת בלבד. במקרה שאין ספרן שהוקצה לבצע את המשימה, לא תיווצר רשומה בטבלה **.RequestAssignments**קשר זה מבטיח שכל משימה מוקצה מטופלת על ידי ספרן מוגדר, ותיעוד הקשר נשמר רק כאשר יש גורם אחראי לטיפול בבקשה.

בס"ד

תרשים DSD:

לאחר שסיימתי עם תרשים ה ,ERD-המשכתי ליצור את תרשים ה-DRD:
תרשים זה מתאר את הנתונים והאופן שבו הם מתקשרים בין הישויות במערכת, ומספק הבנה מעמיקה של האינטראקציות בין הטבלאות כולל סוג הנתונים וכל התכונות שבכל טבלה:





הסבר הטבלאות והנתונים:

טבלת קטגוריית ספרים(BookCategories)

שם השדה	סוג השדה	תיאור	מפתח
Category_id	INT	מספר מזהה ייחודי עבור כל קטגוריה בספרייה	PK
Category	VARCHAR2(50)	שם הקטגוריה הראשית לסיווג ספרים	
Subcategory	VARCHAR2(50)	שם תת-קטגוריה לסיווג נוסף של הספרים	
Description	CLOB	תיאור מילולי מפורט המסביר את הקטגוריה	

כדי ליצור את טבלת הקטגוריות ספרים (BookCategories) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE BookCategories (
  Category_id INT PRIMARY KEY,
  Category VARCHAR2(50) NOT NULL,
  Subcategory VARCHAR2(50) NOT NULL,
  Description CLOB
);
```

טבלת מחברי ספרים (BookAuthors)

שם השדה	סוג השדה	תיאור	מפתח
Author_id	INT	מספר מזהה ייחודי עבור כל מחבר	PK
Author_academic_title	VARCHAR2(10)	התואר האקדמי של המחבר (למשל, ד"ר או פרופ')	
Author_first_name	VARCHAR2(100)	השם הפרטי של המחבר	
Author_last_name	VARCHAR2(100)	שם המשפחה של המחבר	
Author_birth_date	DATE	תאריך הלידה של המחבר	
Nationality	VARCHAR2(50)	הלאום של המחבר	
Biography	CLOB	ביוגרפיה מפורטת של חיי המחבר ופעילותו	

כדי ליצור את טבלת הסופרים (BookAuthors) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE BookAuthors (
  Author_id INT PRIMARY KEY,
  Author_academic_title VARCHAR2(10),
  Author_first_name VARCHAR2(100) NOT NULL,
  Author_last_name VARCHAR2(100) NOT NULL,
  Author_birth_date DATE NOT NULL,
  Nationality VARCHAR2(50) NOT NULL,
  Biography CLOB
);
```

טבלת ספרנים(Librarians)

שם השדה	סוג השדה	תיאור	מפתח
Librarian_id	INT	מספר מזהה ייחודי עבור כל ספרן	PK
Librarian_first_name	VARCHAR2(100)	השם הפרטי של הספרן	
Librarian_last_name	VARCHAR2(100)	שם המשפחה של הספרן	
Librarian_birth_date	DATE	תאריך הלידה של הספרן	
Email	VARCHAR2(100)	כתובת דוא"ל ליצירת קשר עם הספרן	
Phone	VARCHAR2(40)	מספר הטלפון ליצירת קשר עם הספרן	
Hire_date	DATE	התאריך שבו הספרן החל לעבוד בספרייה	

כדי ליצור את טבלת הספרנים (Librarians) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE Librarians (
  Librarian_id INT PRIMARY KEY,
  Librarian_first_name VARCHAR2(100) NOT NULL,
  Librarian_last_name VARCHAR2(100) NOT NULL,
  Librarian_birth_date DATE NOT NULL,
  Email VARCHAR2(100) NOT NULL,
  Phone VARCHAR2(40) NOT NULL,
  Hire_date DATE NOT NULL
);
```

טבלת ספרים(Books)

שם השדה	סוג השדה	תיאור	מפתח
Book_id	INT	מספר מזהה ייחודי עבור כל ספר בספרייה	PK
Book_name	VARCHAR2(255)	שם הספר	
Publication_date	DATE	תאריך פרסום הספר	
Category_id	INT	מספר מזהה לקטגוריה של הספר (קישור לטבלת BookCategories)	FK
Author_id	INT	מספר מזהה של מחבר הספר (קישור לטבלת BookAuthors)	FK
Language	VARCHAR2(50)	השפה שבה נכתב הספר	

כדי ליצור את טבלת הספרים (Books) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE Books (
  Book_id INT PRIMARY KEY,
  Book_name VARCHAR2(255) NOT NULL,
  Publication_date DATE NOT NULL,
  Category_id INT NOT NULL,
  Author_id INT NOT NULL,
  Language VARCHAR2(50) NOT NULL,
  Book_description CLOB,
  FOREIGN KEY (Category_id) REFERENCES BookCategories(Category_id),
  FOREIGN KEY (Author_id) REFERENCES BookAuthors(Author_id)
);
```

טבלת סטטוס ספרים (BookStatuses)

שם השדה	סוג השדה	תיאור	מפתח
Book_id	INT	מספר מזהה של הספר (קישור לטבלת Books)	PK, FK
Status	VARCHAR2(50)	מצב הספר (זמין, מושאל, אבוד, פגום)	PK
Number_of_copies	INT	מספר עותקים זמינים בספרייה	

כדי ליצור את טבלת הסטטוסים של הספרים (BookStatuses) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE BookStatuses (
  Book_id INT NOT NULL,
  Status VARCHAR2(50) NOT NULL, -- available, borrowed, lost, damaged
  Number_of_copies INT NOT NULL,
  PRIMARY KEY (Book_id, Status),
  FOREIGN KEY (Book_id) REFERENCES Books(Book_id)

);
```

טבלת סטודנטים (Students)

שם השדה	סוג השדה	תיאור	מפתח
Student_id	INT	מספר מזהה ייחודי לכל סטודנט	PK
Student_first_name	VARCHAR2(100)	השם הפרטי של הסטודנט	
Student_last_name	VARCHAR2(100)	שם המשפחה של הסטודנט	
Email	VARCHAR2(100)	כתובת דוא"ל של הסטודנט	
Phone	VARCHAR2(40)	מספר טלפון ליצירת קשר עם הסטודנט	
Birth_date	DATE	תאריך הלידה של הסטודנט	
Degree	VARCHAR2(50)	מסלול הלימודים של הסטודנט	
Start_date	DATE	תאריך התחלת הלימודים	

כדי ליצור את טבלת הסטודנטים (Students) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE Students (
  Student_id INT PRIMARY KEY,
  Student_first_name VARCHAR2(100) NOT NULL,
  Student_last_name VARCHAR2(100) NOT NULL,
  Email VARCHAR2(100) NOT NULL,
  Phone VARCHAR2(40) NOT NULL,
  Birth_date DATE NOT NULL,
  Degree VARCHAR2(50) NOT NULL,
  Start_date DATE NOT NULL
);
```

טבלת השאלות ספרים (Lent)

שם השדה	סוג השדה	תיאור	מפתח
Book_id	INT	מספר מזהה של הספר (קישור לטבלת Books)	PK, FK
Student_id	INT	מספר מזהה של הסטודנט (קישור לטבלת Students)	PK, FK
Status	VARCHAR2(50)	סטטוס ההשאלה (מושאל, מוחזר, אבוד)	
Fine_amount	INT	סכום הקנס עבור איחור בהחזרת הספר	
Borrow_date	DATE	תאריך השאלת הספר	PK
Return_date	DATE	תאריך החזרת הספר	

כדי ליצור את טבלת ההשאלות (Lent) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE Lent (
  Book_id INT NOT NULL,
  Student_id INT NOT NULL,
  Status VARCHAR2(50) NOT NULL,
  Fine_amount INT NOT NULL,
  Borrow_date DATE NOT NULL,
  Return_date DATE NOT NULL,
  PRIMARY KEY (Book_id, Student_id, Borrow_date),
  FOREIGN KEY (Book_id) REFERENCES Books(Book_id),
  FOREIGN KEY (Student_id) REFERENCES Students(Student_id)
);
```

טבלת ביקורות על ספרים (Reviews)

שם השדה	סוג השדה	תיאור	מפתח
Book_id	INT	מספר מזהה של הספר (קישור לטבלת Books)	PK, FK
Reviewer_id	INT	מספר מזהה של הסטודנט שהגיש את הביקורת (קישור לטבלת Students)	PK, FK
Review_text	CLOB	תוכן הביקורת	
Rating	INT	דירוג הספר (למשל, מ-1 עד 5)	
Review_date	DATE	תאריך הביקורת	
Review_title	VARCHAR2(255)	כותרת הביקורת	

כדי ליצור את טבלת הביקורות (Reviews) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE Reviews (
  Book_id INT NOT NULL,
  Reviewer_id INT NOT NULL,
  Review_text CLOB NOT NULL,
  Rating INT NOT NULL,
  Review_date DATE NOT NULL,
  Review_title VARCHAR2(255) NOT NULL,
  PRIMARY KEY (Book_id, Reviewer_id),
  FOREIGN KEY (Book_id) REFERENCES Books(Book_id),
  FOREIGN KEY (Reviewer_id) REFERENCES Students(Student_id)
);
```

טבלת חדרי לימוד (StudyRooms)

שם השדה	סוג השדה	תיאור	מפתח
Room_number	INT	מספר מזהה ייחודי לכל חדר לימוד	PK
Library_floor	INT	קומת הספרייה שבה ממוקם החדר	
Library_section	VARCHAR2(50)	החלק בספרייה שבו ממוקם החדר	
Room_capacity	INT	מספר האנשים שהחדר יכול להכיל	
Board	VARCHAR2(1)	האם יש לוח בחדר (Y/N)	
Screen	VARCHAR2(1)	האם יש מסך בחדר (Y/N)	

כדי ליצור את טבלת חדרי הלמידה (StudyRooms) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE StudyRooms (
  Room_number INT PRIMARY KEY,
  Library_floor INT NOT NULL,
  Library_section VARCHAR2(50) NOT NULL,
  Room_capacity INT NOT NULL,
  Board VARCHAR2(1) NOT NULL,  -- Y, N
  Screen VARCHAR2(1) NOT NULL  -- Y, N
);
```

טבלת בקשות חדרים (RequestsRooms)

שם השדה	סוג השדה	תיאור	מפתח
Student_id	INT	מספר מזהה של הסטודנט שהגיש את הבקשה (קישור לטבלת Students)	PK, FK
Request_time	DATE	מועד שליחת הבקשה	PK
Start_time	DATE	מועד התחלת השימוש בחדר	
Duration_hours	NUMBER(4,1)	משך הזמן שבו החדר יושכר (בשעות)	
Number_of_people	INT	מספר האנשים שישתמשו בחדר	
Board	VARCHAR2(1)	האם יש צורך בלוח (Y/N)	
Screen	VARCHAR2(1)	האם יש צורך במסך (Y/N)	
Handled_Librarian	INT	מזהה הספרן שטיפל בבקשה (קישור לטבלת Librarians)	FK
Assigned_room	INT	מספר החדר שהוקצה (קישור לטבלת StudyRooms)	FK
Status	VARCHAR2(50)	סטטוס הבקשה (לדוגמה, מאושרת, נדחתה)	

כדי ליצור את טבלת הבקשות לחדרי למידה (RequestsRooms) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE RequestsRooms (
  Student_id INT NOT NULL,
  Request_time DATE NOT NULL,
  Start_time DATE NOT NULL,
  Duration_hours NUMBER(2,1) NOT NULL,
  Number_of_people INT NOT NULL,
  Board VARCHAR2(1) NOT NULL,
  Screen VARCHAR2(1) NOT NULL,
  Handled_Librarian INT,
  Assigned_room INT,
  Status VARCHAR2(50) NOT NULL,
  PRIMARY KEY (Student_id, Request_time),
  FOREIGN KEY (Student_id) REFERENCES Students(Student_id),
  FOREIGN KEY (Handled_Librarian) REFERENCES Librarians(Librarian_id) ON DELETE SET NULL,
  FOREIGN KEY (Assigned_room) REFERENCES StudyRooms(Room_number)
);
```

טבלת בקשות כלליות בספרייה(LibraryRequests)

שם השדה	סוג השדה	תיאור	מפתח
Request_id	INT	מספר מזהה ייחודי לכל בקשה	PK
Request_type	VARCHAR2(50)	סוג הבקשה (לדוגמה, רכישת ספר)	
Request_description	CLOB	תיאור מפורט של הבקשה	
Request_date	DATE	תאריך שליחת הבקשה	
Request_librarian	INT	מספר מזהה של הספרן שיצר את הבקשה (קישור לטבלת Librarians)	FK
Request_status	VARCHAR2(50)	סטטוס הבקשה (מאושרת, ממתינה, נדחתה)	

כדי ליצור את טבלת הבקשות לספרייה (LibraryRequests) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE LibraryRequests (
  Request_id INT PRIMARY KEY,
  Request_type VARCHAR2(50) NOT NULL,
  Request_description CLOB NOT NULL,
  Request_date DATE NOT NULL,
  Request_librarian INT,
  Request_status VARCHAR2(50) NOT NULL,
  FOREIGN KEY (Request_librarian) REFERENCES Librarians(Librarian_id) ON DELETE SET NULL
);
```

טבלת הקצאות בקשות (RequestAssignments)

שם השדה	סוג השדה	תיאור	מפתח
Request_id	INT	מספר מזהה של הבקשה (קישור לטבלת LibraryRequests)	PK, FK
Assigned_librarian	INT	מזהה הספרן שנבחר לטפל בבקשה (קישור לטבלת Librarians)	FK
Completion_date	DATE	תאריך השלמת הטיפול בבקשה	

כדי ליצור את טבלת ההקצאות לבקשות (RequestAssignments) הרצתי את קוד ה-SQL הבא:

```
CREATE TABLE RequestAssignments (
  Request_id INT PRIMARY KEY,
  Assigned_librarian INT,
  Completion_date DATE,
  FOREIGN KEY (Request_id) REFERENCES LibraryRequests(Request_id) ON DELETE CASCADE,
  FOREIGN KEY (Assigned_librarian) REFERENCES Librarians(Librarian_id) ON DELETE SET NULL
);
```

בס"ד

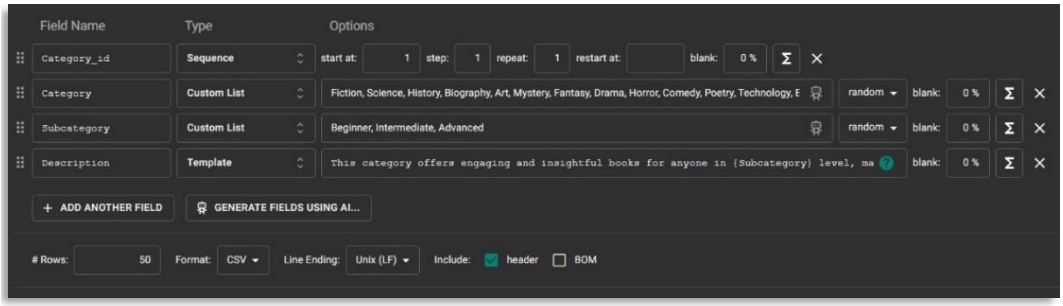
יצירת נתונים

יצירה באמצעות Mockaroo

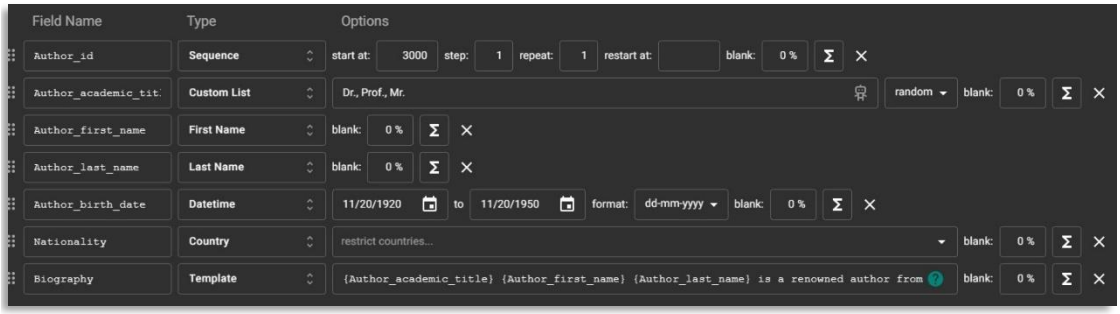
תיאור תהליך יצירת נתוני הדמה
לאחר הגדרת מבנה הטבלאות, נעשה שימוש ב Mockaroo ליצירת נתוני דמה (Mock Data) עבור כל אחת מהטבלאות בפרויקט .

הגדרות הטבלאות באמצעותMockaroo
הטבלאות הבאות הוגדרו באמצעות Mockaroo לצורך יצירת נתוני דמה עבור הפרויקט. בכל תמונה ניתן לראות את תהליך ההגדרה של השדות בטבלה, כולל סוגי הנתונים, מגבלות (כמו ערכים ייחודיים או טווחי תאריכים), קשרים לטבלאות אחרות, והגדרות מותאמות אישית.
תמונות מצורפות לפי הסדר:

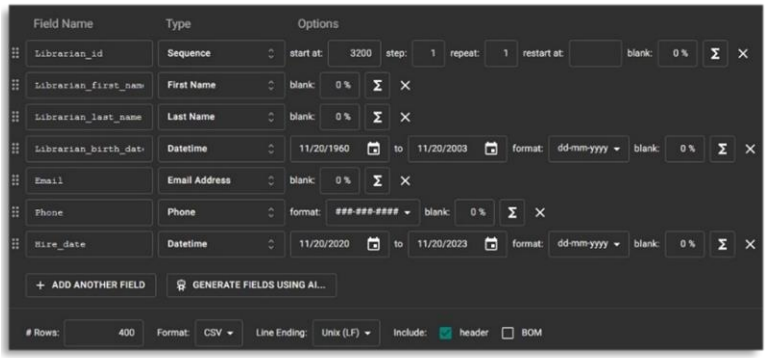
1. BookCategories – קטגוריית ספרים



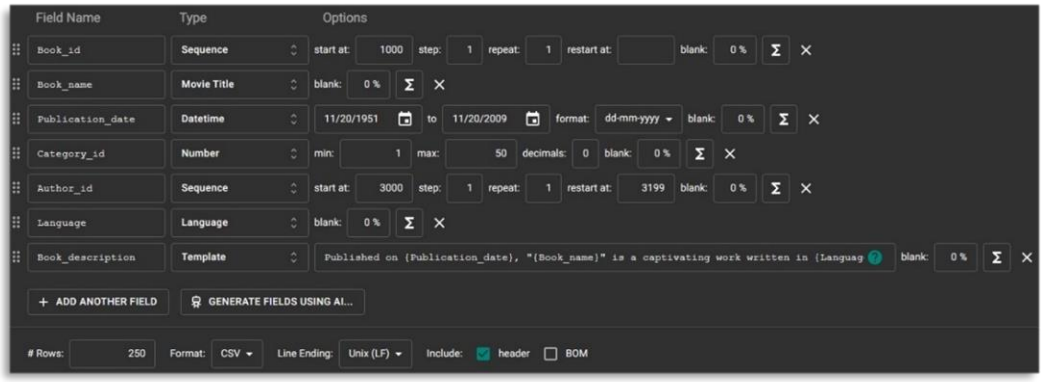
2. BookAuthors – סופרים



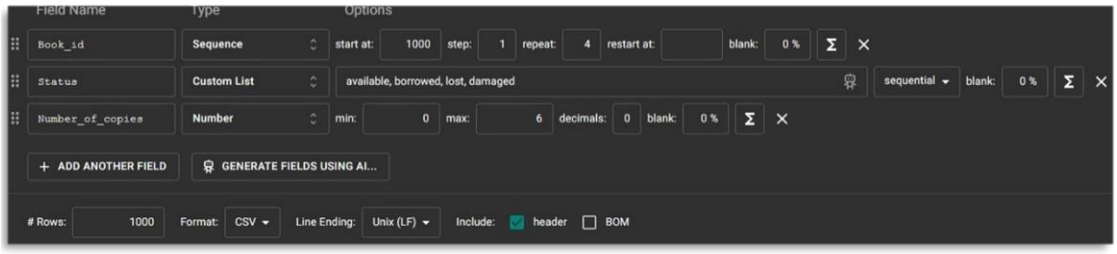
3. Librarians – ספרנים



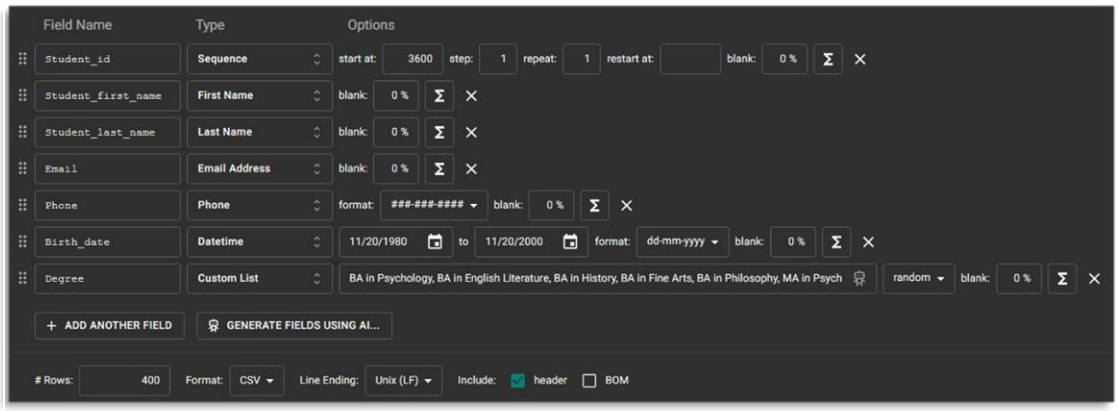
4. Books – ספרים



5. BookStatuses – סטטוסים של ספרים



6. Students – סטודנטים



7. Lent – השאלות

Field Name	Type	Options
Book_id	Number	min: 1000 max: 1249 decimals: 0 blank: 0 %
Student_id	Number	min: 3600 max: 3999 decimals: 0 blank: 0 %
Status	Custom List	borrowed, returned random blank: 0 %
Fine_amount	Number	min: 0 max: 0 decimals: 0 blank: 0 %
Borrow_date	Datetime	10/01/2024 to 10/31/2024 format: dd-mm-yyyy blank: 0 %
Return_date	Datetime	11/01/2024 to 11/30/2024 format: dd-mm-yyyy blank: 0 %

+ ADD ANOTHER FIELD | GENERATE FIELDS USING AI...

Rows: 1000 Format: CSV Line Ending: Unix (LF) Include: header BOM

8. Reviews – ביקורות + סקריפט ל- Review_title:

Field Name	Type	Options
Book_id	Number	min: 1000 max: 1249 decimals: 0 blank: 0 %
Reviewer_id	Number	min: 3600 max: 3999 decimals: 0 blank: 0 %
Review_text	Paragraphs	at least 1 but no more than 2 blank: 0 %
Rating	Number	min: 1 max: 5 decimals: 0 blank: 0 %
Review_date	Datetime	11/01/2023 to 11/30/2024 format: dd-mm-yyyy blank: 0 %
Review_title	Paragraphs	at least 1 but no more than 3 blank: 0 %

+ ADD ANOTHER FIELD | GENERATE FIELDS USING AI...

Rows: 1000 Format: CSV Line Ending: Unix (LF) Include: header BOM

```
if field("Rating") == 1 then
  ["Poor", "Boring", "Disappointing"].sample
elsif field("Rating") == 2 then
  ["Average", "Predictable", "Mildly Entertaining"].sample
elsif field("Rating") == 3 then
  ["Informative", "Educational", "Insightful"].sample
elsif field("Rating") == 4 then
  ["Amazing", "Thought-provoking", "Engaging"].sample
elsif field("Rating") == 5 then
  ["This is awesome", "Excellent", "Heartwarming", "Hilarious"].sample
else
  "No rating provided"
end
```

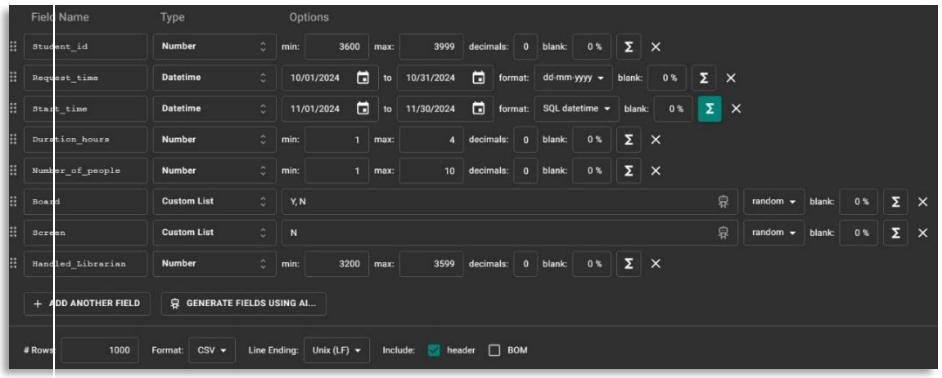
9. StudyRooms – חדרי למידה

Field Name	Type	Options
Room_number	Sequence	start at: 1 step: 1 repeat: 1 restart at: blank: 0 %
Library_floor	Number	min: 1 max: 2 decimals: 0 blank: 0 %
Library_section	Custom List	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T random blank: 0 %
Room_capacity	Number	min: 1 max: 10 decimals: 0 blank: 0 %
Board	Custom List	Y random blank: 0 %
Screen	Custom List	Y, N random blank: 0 %

+ ADD ANOTHER FIELD | GENERATE FIELDS USING AI...

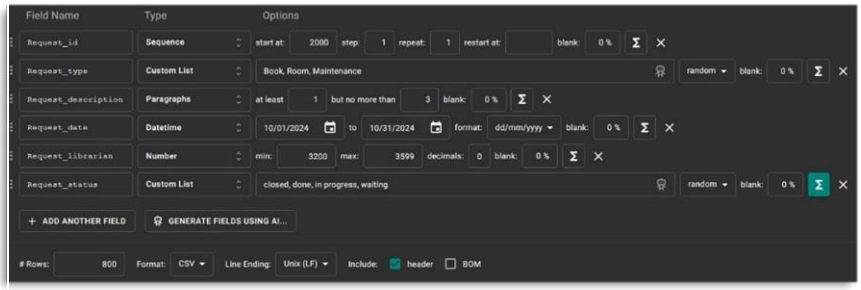
Rows: 50 Format: CSV Line Ending: Unix (LF) Include: header BOM

10. RequestsRooms – בקשות לחדרי למידה + סקריפט ל- Start_time



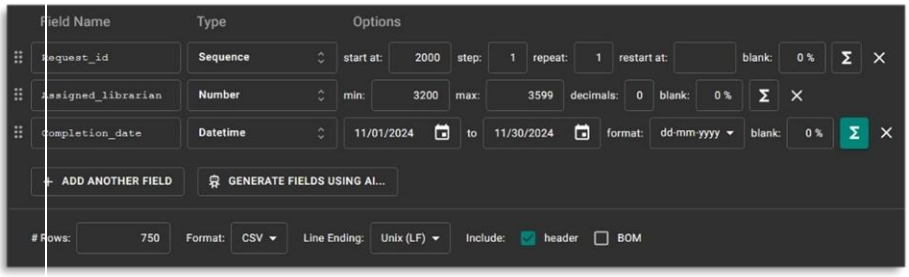

```
if time(this) >= '00:00:00' and time(this) <= '06:59:59' then
    this + hours(12) # Add 12 hours for early morning times
elsif time(this) >= '22:01:00' and time(this) <= '23:59:59' then
    this - hours(12) # Subtract 12 hours for late night times
else
    this # Keep the time as is for other times
end
```

11. LibraryRequests – בקשות לספרייה + סקריפט ל-request_status:




```
if field("Request_id") < 2500 then
  ["closed", "done"].sample
elsif field("Request_id") >= 2500 and field("Request_id") < 2750 then
  "in progress"
else
  "waiting"
end
```

12. RequestAssignments – הקצאות לבקשות + סקריפט ל-Completion_date (בגלל שהגדרתי את המשימות עם id שגדול מ-2500 להיות עם שיבוץ אבל עדיין לא להסתיים אז Completion_date הוא null) :




```
if field("Request_id") >= 2500 then
  nil
else
  this
end
```

התמונות ממחישות את תהליך העבודה ומציגות את ההגדרות המדויקות ששימשו ליצירת הנתונים.

יצירת נתונים באמצעות קוד פיתון וקבצים

תיאור הקוד שלי ליצירת נתונים באמצעות Python וקבצי בסיס

בפרויקט הזה בניתי קוד Python שמטרתו הייתה ליצור נתוני דמה בצורה יסודית ומדויקת עבור מסד הנתונים שלי. הקפדתי לעבוד באופן מסודר, תוך שמירה על התאמה בין הטבלאות, יצירת קשרים בין נתונים ושימוש בנתונים ריאליסטיים. הנה תיאור מפורט של מה שעשיתי:

1. קריאה ועיבוד נתונים ראשוניים
השתמשתי בקובצי JSON ששימשו כבסיס לנתונים:
- `data.json` סיפק מידע על מחברים, ספרים וקטגוריות.
- `category_description.json` כלל תיאורים מפורטים של קטגוריות ותתי-קטגוריות.
- `libraryRequests.json` הכיל תיאורים של בקשות שונות בספרייה.
קראתי את הנתונים האלו לתוך מבני נתונים בפייתון (רשימות ומילונים) והתחלתי לעבד אותם כך שיתאימו לטבלאות שלי.

2. יצירת נתונים רנדומליים
כתבתי פונקציות מותאמות ליצירת נתונים רנדומליים שהתאימו לכל טבלה:
מחברים (Authors):
- עיבדתי שמות מחברים כדי לחלק אותם לשדות `First_name`, `Last_name` ו-`Academic_title`.
- יצרתי תאריכי לידה רנדומליים בטווח 1940-2010 ושייכתי לאומים ממדינות נפוצות.

ספרים (Books):
- יצרתי מזהי ספרים ייחודיים ושייכתי אותם לקטגוריות ולמחברים. לכל ספר הוספתי שפה מתוך רשימה מוגדרת.

סטודנטים (Students):
- כתבתי פונקציה ליצירת מספרי תעודות זהות ישראליות ייחודיות עם חישוב ספרת ביקורת.
- יצרתי כתובות דוא"ל, מספרי טלפון, ותאריכי התחלת לימודים בצורה רנדומלית אך הגיונית.

חדרי לימוד (StudyRooms):
- יצרתי חדרים עם תכונות כמו קיבולת, זמינות לוח/מסך, ומצב החדר.

בקשות חדרים (RoomRequests):
- ניהלתי זמנים פנויים לכל חדר כך שלא ייווצרו התנגשויות בין בקשות.

השאלות ספרים (Lent):
- ניהלתי את תהליך השאלת הספרים, כולל קביעת סטטוסים כמו "מושאל", "אבוד", או "פגום". חישבתי גם קנסות במקרה של החזרה מאוחרת.

ביקורות (Reviews):
- יצרתי ביקורות על ספרים שהוחזרו, כולל דירוגים, כותרות, ותאריכי כתיבת ביקורות.

וכו'

3. שמירה על קשרים בין הטבלאות

דאגתי לשמור על הקשרים הלוגיים בין הטבלאות:
- לדוגמה, כל ספר שויך למחבר ולקטגוריה מתאימה.
- התעודות זהות של הספרנים וסטודנטים הם תעודות זהות ישראליות תקינות – כלומר קיימת פונקציה ייעודית שמחוללת 8 ספרות של תעודת זהות ישראלית ואז מחשב את הספרה האחרונה לפי הפורמט בנוסף דאגתי שלא יהיה התנגשויות בין תעודות זהות ע"י שמירה של set של כל התעודות זהות שקיימות כבר.
- כתובות האימייל הם גם ייחודיות וגם מורכבים מקומבינציה של שם האדם ותאריך הלידה שלו בצורה אקראית.
- בקשות חדרים שויכו לסטודנטים ולחדרים עם לוודא שאין התנגשויות בשימות החדרים ועדיפות למי שביקש את החדר קודם עם אופטימיזציה לתת לסטודנט את המינימום שהוא דרש כדי לתת לכמה שיותר סטודנטים את הגישה לחדרים.
- כל מזהה זר (Foreign Key) הוגדר בצורה שתואמת את מזהה הטבלה המקושרת.

4. יצירת קובצי CSV

בסיום יצירת הנתונים, המרה שלהם לקובצי CSV הייתה שלב חשוב:
- ייצאתי כל טבלה לקובץ CSV משלה, לדוגמה `Lent.csv`, `Students.csv`, `Books.csv`.
- וידאתי שכל הנתונים נשמרו בפורמט התואם למבנה מסד הנתונים שלי.

5. יצירת פקודות INSERT

לאחר יצירת קובצי ה-CSV, המרה שלהם לפקודות SQL הייתה קלה:
- יצרתי קובץ `insert.sql` הכולל את כל פקודות ה-INSERT.
- דאגתי לכך שכל רשומה תכיל את כל השדות הנדרשים ותתאים למבנה הטבלאות שלי.

6. דגש על דיוק וייחודיות

במהלך העבודה, השתמשתי במנגנונים למניעת כפילויות:
- שמרתי מערכי נתונים של כתובות דוא"ל, מזהי תעודות זהות ומספרי טלפון כדי לוודא ייחודיות.
- וידאתי שתאריכים, כמו תאריכי החזרות והשאלות, יהיו ריאליסטיים ולא חופפים.

7. תוצאה

בסיום התהליך, יצרתי מערכת נתונים מלאה ומקושרת שמתאימה לפרויקט שלי. הנתונים שנוצרו הם מציאותיים, והקשרים ביניהם תואמים את המודל שהגדרתי במסד הנתונים.

העבודה על הקוד הזה אפשרה לי גם לבדוק את מסד הנתונים שלי עם נתונים דינמיים וריאליסטיים גם ליצור כמות עצומה של נתונים (כ-100 אלף רשומות) וגם לוודא שהכול עובד בצורה חלקה.

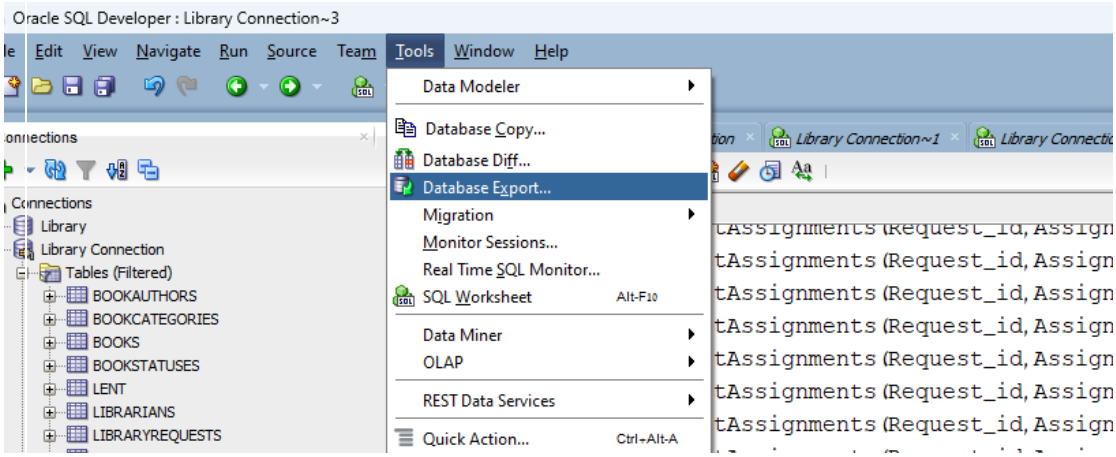
זה צילום מסך מחלק מאוד קטן מהקוד שיצרתי את בקשות הספרייה:

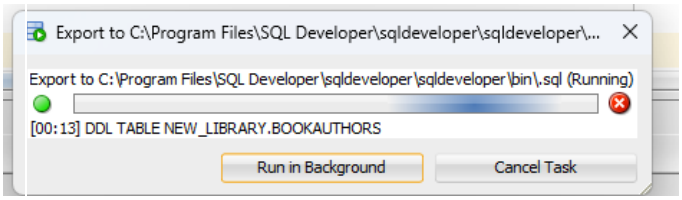
```python
def generate_library_requests(number_of_requests: int, librarians: list) -> list:
    """
    @summary:
        Generate a list of library requests.
    @param number_of_requests: int
        The number of library requests to generate.
    @param librarians: list
        A list of librarians.
    @return: list
        A list of library requests.
    """
    library_requests = []
    # new list for the librarian that hendled the request - only if the status is done or closed or in prog
    # requests_assigned = [] # contains a random librarian id, request_id, request_end_time(case the status
    for i in range(number_of_requests):
        request_type = random.choice(['books', 'room', 'maintenance', 'other'])
        Request_description = get_random_request_description(request_type, library_requests_description)
        Request_date = generate_date_by_days_range(datetime.date(2024, 1, 1), datetime.date(2024, 11, 11))
        Request_librarian = generate_librarian_id(librarians)
        Request_status = random.choice(['waiting', 'in progress', 'done', 'closed'])
        request = {
            'Request_id': i + 1,
            'Request_type': request_type,
            'Request_description': Request_description,
            'Request_date': Request_date,
            'Request_librarian': Request_librarian,
            'Request_status': Request_status
        }
        library_requests.append(request)
    return library_requests
```

בס"ד

גיבוי הנתונים:

יצרתי קובץ Backup לגיבוי הסכמה והנתונים. לאחר מכן יצרתי קובץ drop.sql כדי למחוק את כל הטבלאות ואת הרצתי את ה-backup כדי לראות שהכול עובר כמו שצריך והכול יצא תקין.





תמונה של ה- backup שנוצר:

```
-------------------------------------------------------
--  DDL for Table BOOKAUTHORS
-------------------------------------------------------

  CREATE TABLE "NEW_LIBRARY"."BOOKAUTHORS"
   (    "AUTHOR_ID" NUMBER(*,0),
    "AUTHOR_ACADEMIC_TITLE" VARCHAR2(10 BYTE),
    "AUTHOR_FIRST_NAME" VARCHAR2(100 BYTE),
    "AUTHOR_LAST_NAME" VARCHAR2(100 BYTE),
    "AUTHOR_BIRTH_DATE" DATE,
    "NATIONALITY" VARCHAR2(50 BYTE),
    "BIOGRAPHY" CLOB
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"
 LOB ("BIOGRAPHY") STORE AS SECUREFILE (
  TABLESPACE "USERS" ENABLE STORAGE IN ROW 4000 CHUNK 8192
  NOCACHE LOGGING  NOCOMPRESS  KEEP_DUPLICATES
  STORAGE(INITIAL 262144 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)) ;
-------------------------------------------------------
--  DDL for Table BOOKCATEGORIES
-------------------------------------------------------

  CREATE TABLE "NEW_LIBRARY"."BOOKCATEGORIES"
   (    "CATEGORY_ID" NUMBER(*,0),
    "CATEGORY" VARCHAR2(50 BYTE),
    "SUBCATEGORY" VARCHAR2(50 BYTE),
    "DESCRIPTION" CLOB
   ) SEGMENT CREATION IMMEDIATE
  PCTFREE 10 PCTUSED 40 INITRANS 1 MAXTRANS 255
 NOCOMPRESS LOGGING
  STORAGE(INITIAL 65536 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0 FREELISTS 1 FREELIST GROUPS 1
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)
  TABLESPACE "USERS"
 LOB ("DESCRIPTION") STORE AS SECUREFILE (
  TABLESPACE "USERS" ENABLE STORAGE IN ROW 4000 CHUNK 8192
  NOCACHE LOGGING  NOCOMPRESS  KEEP_DUPLICATES
  STORAGE(INITIAL 262144 NEXT 1048576 MINEXTENTS 1 MAXEXTENTS 2147483645
  PCTINCREASE 0
  BUFFER_POOL DEFAULT FLASH_CACHE DEFAULT CELL_FLASH_CACHE DEFAULT)) ;
-------------------------------------------------------
```