

****

**מגיש: אלון יערי**

**ת.ז: 325647170**

**שם המחנה: טל בר**

**פרטי מכללה: WAN-TEC בארי**

**סמל מוסד: 659284**

תוכן עניינים

[רקע תיאורטי: 3](#_Toc134360206)

[מטרות פונקציונליות: 3](#_Toc134360207)

[רשימת משתמשים מורשים: 4](#_Toc134360208)

[טבלאות 4](#_Toc134360209)

[תרשימים 5](#_Toc134360210)

[תרשים ERD: 5](#_Toc134360211)

[תרשים DSD: 5](#_Toc134360212)

[קוד הפרויקט 6](#_Toc134360213)

[יצירת הטבלאות 6](#_Toc134360214)

[שאילתות 7](#_Toc134360215)

[טריגרים 11](#_Toc134360216)

[פרוצדורות 13](#_Toc134360217)

## רקע תיאורטי:

הרעיון של חיבור בין אנשים עם תחומי עניין ותוכניות טיול דומים אינו חדש, אך התקדמות הטכנולוגיה הקלה על כך. אתרי רשתות חברתיות ואפליקציות נסיעות הפכו יותר ויותר פופולריות, ומספקות פלטפורמות לאנשים להתחבר עם אחרים, לחלוק חוויות ולתכנן טיולים יחד.

מחקרים הראו כי קשרים ואינטראקציות חברתיות חשובים לרווחתנו ולאושרנו. בפרט, מחקרים מצאו שלתמיכה חברתית יכולה להיות השפעה חיובית על בריאות הנפש, רמות הלחץ ושביעות הרצון הכללית מהחיים. טיול עם אחרים יכול לספק תחושת אחווה, חוויות משותפות ורשת תמיכה, שיכולה לעזור להקל על תחושות הבדידות והבידוד.

מלבד ההטבות החברתיות, שמגיעות בטיול עם שותפים, לטיול עם אחרים יכולים להיות גם יתרונות מעשיים, כמו חלוקת עלויות, בטיחות והיכולת לחקור מקומות חדשים בביטחון רב יותר. מחקרים הראו גם שלאנשים העוסקים בפעילויות בחוץ ומבלים בטבע יש תוצאות בריאותיות גופניות ונפשיות טובות יותר, כולל רמות מתח נמוכות יותר וסיכון מופחת למחלות כרוניות.

TrailMates שואף להתבסס על היתרונות הללו על ידי מתן פלטפורמה לחיבור בין אנשים החולקים אהבה לפעילויות חוצות ולנסיעות הרפתקאות. על ידי התאמת משתמשים על סמך תחומי העניין, הציוד ותוכניות הנסיעה שלהם, האפליקציה תוכל לעזור לאפשר חיבורים משמעותיים ולספק פלטפורמה לתכנון ותיאום טיולים יחד. על ידי שיתוף ציוד והוצאות, משתמשים יכולים להפוך את הנסיעות לזמינות ונגישות יותר, תוך קידום קיימות והפחתת ההשפעה הסביבתית של נסיעות.

בנוסף ליתרונות של קשרים חברתיים וחלוקת עלויות, הפרויקט שלך יכול גם לקדם חילופי תרבות וצמיחה אישית. כאשר אנשים מטיילים עם אחרים מרקעים ותרבויות שונות, יש להם הזדמנות ללמוד על נקודות מבט ודרכי חיים חדשות. זה יכול להוביל לצמיחה אישית ולהבנה עמוקה יותר של העולם הסובב אותנו.

לסיכום, לפרויקט, יש פוטנציאל לספק שירות רב ערך לאנשים שנהנים מפעילויות חוצות ומנסיעות הרפתקאות, תוך קידום קשרים חברתיים, בניית קהילה, נסיעות בר קיימא, צמיחה אישית ויתרונות בריאותיים רבים. לאפליקציה יש פוטנציאל ליצור קהילה גלובלית של חובבי טבע שיכולים לחלוק את החוויות שלהם, ללמוד אחד מהשני, להעניק השראה זה לזה, לחקור יעדים חדשים ולרדוף אחר התשוקות שלהם.

## מטרות פונקציונליות:

1. ניהול משתמשים – ממסד הנתונים ישמור מידע רלוונטי על המשתמשים. המידע יכיל נתונים כמו מיקום, שפה, פרטי קשר ועוד.
2. ניהול טיולים – ממסד הנתונים ישמור מידע רלוונטי על הטיולים. המידע יכיל כמו אורך, מיקום, דרגת קושי ציוד מיוחד ועוד.
3. ניהול ציוד - מסד הנתונים אמור לאפשר למשתמשים לנהל את מלאי הציוד שלהם, כולל הוספה, עריכה והסרה של פריטים.
4. אחזור נתונים – ממסד הנתונים יאפשר למשתמשים לאחזר נתונים במהירות ובקלות. המערכת צריכה לספק יכולות חיפוש וסינון כדי לעזור למשתמשים לאתר את הנתונים הנדרשים במהירות ויעילות.
5. חיפוש וסינון – המערכת תאפשר למשתמשים למצוא את הטיול המתאים עבורם לפי קריטריונים שונים.
6. אלגוריתם התאמה - מסד הנתונים צריך לכלול אלגוריתם שמתאים למשתמשים על סמך תחומי העניין, הציוד ותוכניות הנסיעה שלהם. האלגוריתם צריך לקחת בחשבון גורמים כמו מיקום, שפה ורמת פעילות מועדפת.

### רשימת משתמשים מורשים:

1. **מטיילים** – האפליקציה תהיה קודם כל עבור המטיילים. היא תעזור להם למצוא חברים, מקומות אותנטיים ומסלולי טיול חדשים. בנוסף האפליקציה תעזור למטייילים מבחינת הציוד הדרוש, וההתארגנות למסע שלהם.
2. **מדריכים/מארחים מקומיים** – האפליקצייה תיתן אפשרות למקומיים להיפגש עם התיירים ואולי גם להדריך אותם. כך יווצר קשר בין המקומיים לתיירים, קשר שיכול להועיל ולתרום לשני הצדדים.
3. **מנהלי טיולים** **אזוריים** – אלו המשתמשים שיהיו אחראים על ניהול וארגון טיולים במדינה ספציפית. תהיה להם גישה מיוחדת לתכונות מסוימות בתוך האפליקציה שיאפשרו להם ליצור ולנהל מסלולים. הם גם יהיו אחראים לוודא שכל סידורי הנסיעה הדרושים קיימים, ויהוו נקודת הקשר העיקרית לנוסעים שיש להם שאלות או חששות לגבי הטיול שלהם.
4. **מנהל/אדמין** - אלה אנשים שיש להם גישה ושליטה כללית על מערכת TrailMates. הם אחראים לניהול המערכת כולה ולוודא שהכל יתנהל בצורה חלקה.

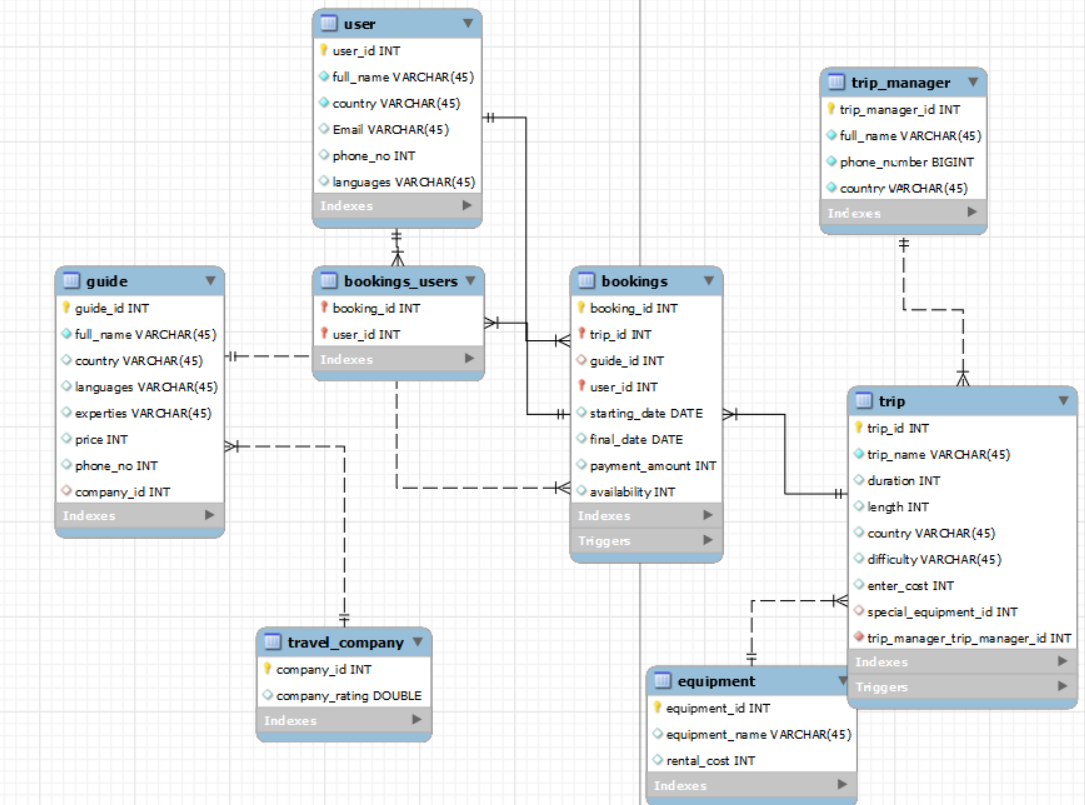
## טבלאות

1. **טבלת משתמשים**: טבלה זו תשמור מידע על כל משתמש, כגון שם המשתמש, תעודת זהות, כתובת האימייל, מספר פלאפון וכל פרט אישי רלוונטי אחר.
2. **טבלת טיולים**: טבלה זו תשמור מידע על כל טיול שמשתמש יוצר או מצטרף אליו, כגון שם הטיול, אורך הטיול (בזמן - שעות ובמרחק - קילומטרים), מיקום, סוג פעילות, רמת מיומנות והמלווים המועדפים. לכל נסיעה יכול להיות משתמש אחד או יותר, וכל משתמש יכול להיות חלק מטיול אחד או יותר.
3. **טבלת מדריכים**: טבלה זו תשמור מידע על מדריכים המציעים את שירותיהם למשתמשים. זה יכול לכלול את שמם, מיקום, שפות מדוברות, תחומי התמחות ותמחור. כל מדריך יכול להיות משויך לטיול אחד או יותר, ולכל טיול יכול להיות מדריך אחד או יותר.
4. **טבלת הציוד:** טבלה זאת תיתן מספר סידורי לכל סוג ציוד. בנוסף, היא תשמור את "שם הציוד", ומחיר ההשכרה שלו לשעה.
5. **טבלת חברות הטיולים:** טבלה זאת תשמור את חברות הטיולים, ואת המספר הסידורי שלהם. כמו כן, דירוג החברה יהיה שמור בטבלה, וכל מדריך יוכל להשתייך לחברת טיולים אחת בלבד.
6. **טבלת ההזמנות:** טבלה זאת תכיל את כל ההזמנות שנעשו דרך האפליקציה. פרטי ההזמנה שישמרו הם מספר הסידורי של הטיול הנבחר, ותעודת זהות של המדריך והמטיילים. כמו כן, טבלה זאת תוכל לשמור את התאריך שבו הטיול יצא ויסתיים, וכן אם התקבלה הנחה על הטיול. בנוסף ישמרו מספר המקומות הפנויים בהזמנה זאת.
7. **טבלת ההרשמה:** טבלה זאת תכיל את מספר תעודות הזהות של המשתמשים אשר נרשמו לטיול מסוים, אשר יזוהה בטבלה באמצעות המספר המזהה של ההזמנה.

## תרשימים

### תרשים ERD:

### תרשים DSD:



## קוד הפרויקט

### יצירת הטבלאות

CREATE TABLE `bookings` ( `booking\_id` int NOT NULL AUTO\_INCREMENT, `trip\_id` int NOT NULL, `guide\_id` int DEFAULT NULL, `user\_id` int NOT NULL, `starting\_date` date DEFAULT NULL, `final\_date` date DEFAULT NULL, `payment\_amount` int DEFAULT NULL, `availability` int DEFAULT NULL, PRIMARY KEY (`booking\_id`,`trip\_id`,`user\_id`), UNIQUE KEY `booking\_id\_UNIQUE` (`booking\_id`), KEY `trip\_id\_idx` (`trip\_id`), KEY `guide\_id\_idx` (`guide\_id`), KEY `user\_id\_idx` (`user\_id`), CONSTRAINT `guide\_id` FOREIGN KEY (`guide\_id`) REFERENCES `guide` (`guide\_id`), CONSTRAINT `trip\_id` FOREIGN KEY (`trip\_id`) REFERENCES `trip` (`trip\_id`), CONSTRAINT `user\_id` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`) )

CREATE TABLE `equipment` ( `equipment\_id` int NOT NULL AUTO\_INCREMENT, `equipment\_name` varchar(45) DEFAULT NULL, `rental\_cost` int DEFAULT NULL, PRIMARY KEY (`equipment\_id`), UNIQUE KEY `equipment\_id\_UNIQUE` (`equipment\_id`) , CONSTRAINT `rental\_cost\_positive\_check` CHECK (`rental\_cost` > 0 OR `rental\_cost` IS NULL)

CREATE TABLE `guide` ( `guide\_id` int NOT NULL, `full\_name` varchar(45) NOT NULL, `country` varchar(45) DEFAULT NULL, `languages` varchar(45) DEFAULT NULL, `experties` varchar(45) DEFAULT NULL, `price` int DEFAULT NULL, `phone\_no` int DEFAULT NULL, `company\_id` int DEFAULT NULL, PRIMARY KEY (`guide\_id`), UNIQUE KEY `guide\_id\_UNIQUE` (`guide\_id`), KEY `company\_id\_idx` (`company\_id`), CONSTRAINT `company\_id` FOREIGN KEY (`company\_id`) REFERENCES `travel\_company` (`company\_id`) )

CREATE TABLE `travel\_company` (`company\_id` int NOT NULL AUTO\_INCREMENT, `company\_rating` double DEFAULT NULL, PRIMARY KEY (`company\_id`), UNIQUE KEY `company\_id\_UNIQUE` (`company\_id`),CONSTRAINT `company\_rating\_check` CHECK ((`company\_rating` between 0 and 10)))

) CREATE TABLE `trip` ( `trip\_id` int NOT NULL AUTO\_INCREMENT, `trip\_name` varchar(45) NOT NULL, `duration` int unsigned DEFAULT NULL, `length` int unsigned DEFAULT NULL, `country` varchar(45) DEFAULT NULL, `difficulty` varchar(45) DEFAULT NULL, `enter\_cost` int unsigned DEFAULT NULL, `special\_equipment\_id` int DEFAULT NULL, PRIMARY KEY (`trip\_id`), UNIQUE KEY `trip\_id\_UNIQUE` (`trip\_id`), KEY `special\_equipment\_id\_idx` (`special\_equipment\_id`), CONSTRAINT `special\_equipment\_id` FOREIGN KEY (`special\_equipment\_id`) REFERENCES `equipment` (`equipment\_id`), CONSTRAINT `duration\_positive\_check` CHECK (`duration` > 0 OR `duration` IS NULL),CONSTRAINT `length\_positive\_check` CHECK (`length` > 0 OR `length` IS NULL), CONSTRAINT `enter\_cost\_positive\_check` CHECK (`enter\_cost` > 0 OR `enter\_cost` IS NULL) );

CREATE TABLE `user` (`user\_id` int NOT NULL AUTO\_INCREMENT, `full\_name` varchar(45) NOT NULL, `country` varchar(45) NOT NULL, `Email` varchar(45) DEFAULT NULL, `phone\_no` int DEFAULT NULL, `languages` varchar(45) DEFAULT NULL, PRIMARY KEY (`user\_id`),UNIQUE KEY `user\_id\_UNIQUE` (`user\_id`))

CREATE TABLE `bookings\_users` ( `booking\_user\_id` int NOT NULL AUTO\_INCREMENT, `booking\_id` int DEFAULT NULL, `user\_id` int DEFAULT NULL, PRIMARY KEY (`booking\_user\_id`), KEY `booking\_id` (`booking\_id`), KEY `user\_id` (`user\_id`), CONSTRAINT `bookings\_users\_ibfk\_1` FOREIGN KEY (`booking\_id`) REFERENCES `bookings` (`booking\_id`), CONSTRAINT `bookings\_users\_ibfk\_2` FOREIGN KEY (`user\_id`) REFERENCES `user` (`user\_id`) )

CREATE TABLE trip\_manager (

trip\_manager\_id INT AUTO\_INCREMENT PRIMARY KEY,

full\_name VARCHAR(50) NOT NULL,

phone\_number INT,

country VARCHAR(50) NOT NULL

);

### שאילתות

|  |  |  |
| --- | --- | --- |
| שם השאילתה | יעוד | קוד |
| הטיולים היקרים ביותר (מטיילים) | להחזיר את חמשת הטיולים היקרים ביותר. | SELECT t.trip\_name, AVG(b.payment\_amount) AS avg\_payment  FROM bookings b  INNER JOIN trip t ON b.trip\_id = t.trip\_id  GROUP BY b.trip\_id  ORDER BY avg\_payment DESC  LIMIT 5; |
| חברת הטיולים הרווחית ביותר (מדריכים) | להחזיר את חברת הטיולים הרווחית ביותר בשנה האחרונה | SELECT travel\_company.company\_id, SUM(bookings.payment\_amount) AS revenue  FROM travel\_company  INNER JOIN guide ON travel\_company.company\_id = guide.company\_id  INNER JOIN bookings ON guide.guide\_id = bookings.guide\_id  WHERE bookings.starting\_date BETWEEN DATE\_SUB(CURRENT\_DATE(), INTERVAL 1 YEAR) AND CURRENT\_DATE()  GROUP BY travel\_company.company\_id  ORDER BY revenue DESC; |
| המדריך המבוקש ביותר (מטיילים ואדמינים) | להחזיר את מספר הטיולים של כל מדריך. | SELECT bookings.guide\_id, COUNT(bookings.booking\_id) AS num\_bookings  FROM bookings  LEFT JOIN bookings\_users ON bookings.booking\_id = bookings\_users.booking\_id  GROUP BY bookings.guide\_id  ORDER BY num\_bookings DESC; |
| מידע על ציוד(מטיילים ומדריכים) | להחזיר את הציוד המבוקש ביותר ואת מחירו. | SELECT equipment.equipment\_name, COUNT(bookings.booking\_id) AS num\_bookings, AVG(equipment.rental\_cost) AS avg\_rental\_cost  FROM equipment  INNER JOIN trip ON equipment.equipment\_id = trip.special\_equipment\_id  INNER JOIN bookings ON trip.trip\_id = bookings.trip\_id  GROUP BY equipment.equipment\_name; |
| מחיר כל טיול. (מטיילים) | להחזיר את רשימת הטיולים ומחירם. | SELECT trip\_id, SUM(payment\_amount) AS total\_payment  FROM bookings  GROUP BY trip\_id; |
| טיולים והזמנות (מטיילים) | להחזיר את שם הטיולים ואת המשתמשים שנרשמו אליו | SELECT trip.trip\_name, COUNT(\*) AS num\_users  FROM bookings  INNER JOIN trip ON bookings.trip\_id = trip.trip\_id  INNER JOIN bookings\_users ON bookings.booking\_id = bookings\_users.booking\_id  GROUP BY trip.trip\_name  ORDER BY num\_users DESC; |
| חברת הטיולים הטובה ביותר. (מטיילים ומדריכים) | להחזיר את חברת הטילים בעלת הדירוג הגבוה ביותר | SELECT company\_id, company\_rating  FROM travel\_company  ORDER BY company\_rating DESC  LIMIT 1; |
| דוברי הצרפתית (מטיילים) | למשתמשים דוברי הצרפתית, להחזיר את המדריכים אשר יודעים צרפתית. | SELECT \*  FROM guide  WHERE languages LIKE '%French%'; |
| המומחים (מטיילים) | להחזיר את כלל המדריכים אשר מומחים בטיפוס הרים | SELECT \*  FROM guide  WHERE expertise LIKE '%mountain climbing%'; |
| מדריכים משולבים (מטיילים) | מדריכים אשר יודעים שפה מסויימת וגם מומחים בתחום ספציפי. | SELECT \*  FROM guide  WHERE languages LIKE '%French%' AND expertise LIKE '%mountain climbing%'; |
| רווחי המדריכים (מדריכים ואדמינים) | מחזיר את רשימת המדריכים, בסדר יורד, לפי גודל הכנסתם מהטיולים. | SELECT g.guide\_id, g.full\_name, COUNT(\*) AS num\_bookings, AVG(b.payment\_amount) AS avg\_payment  FROM guide g  LEFT JOIN bookings b ON g.guide\_id = b.guide\_id  GROUP BY g.guide\_id, g.full\_name  ORDER BY num\_bookings DESC; |
| שווי הציוד המושכר (אדמינים) | מחזיר את שווי הציוד המושכר המשומם באפליקציה | SELECT SUM(e.rental\_cost) AS total\_rental\_cost  FROM bookings b  INNER JOIN trip t ON b.trip\_id = t.trip\_id  INNER JOIN equipment e ON t.special\_equipment\_id = e.equipment\_id  WHERE DATEDIFF(b.final\_date, b.starting\_date) > 1; |
| אורך הטיול המצטבר לפי רמת קושי. (מטיילים) | מחזיר את האורך המצטבר של כלל הטיולים באותה רמת קושי. | SELECT trip.difficulty, AVG(trip.duration) as avg\_duration  FROM trip  GROUP BY trip.difficulty; |
| בחירת טיול לפי מדינה ורמת קושי. (מטיילים) | מחזיר את כלל הטיולים בברזיל אשר ברמת קושי מתקדמת | SELECT \*  FROM trip  WHERE difficulty = 'Moderate' AND country = 'Brazil'; |
| מידע כללי על טיולים יקרים. (מטיילים) | למשתמשים שרוצים להתפנק, מידע על כל הטיולים היקרים. | SELECT trip\_name, duration, enter\_cost  FROM trip  WHERE enter\_cost > 1000  ORDER BY enter\_cost DESC; |
| הזמנות של משתמשים. (מטיילים ואדמינים) | להחזיר את כמות ההזמנות שבוצעה ע"י כל משתמש. | SELECT u.full\_name, COUNT(\*) AS booking\_count  FROM user u  LEFT JOIN bookings b ON u.user\_id = b.user\_id  GROUP BY u.user\_id  ORDER BY booking\_count DESC; |
| טיולים של מדריכים. (אדמינים) | להחזיר את פרטי המדריכים שמספר הטיולים שלהם עולה על... | SELECT bookings.guide\_id, full\_name, COUNT(\*) as num\_trips  FROM bookings  INNER JOIN guide ON bookings.guide\_id = guide.guide\_id  GROUP BY guide\_id, full\_name  HAVING num\_trips >= 3; |
| מציאת טיול מתאים. (מטיילים) | לאפשר למשתמש לבחור את הטיול המתאים ביותר עבורו, גם במדינה וגם ברמת הקושי. | SELECT \*  FROM trip  WHERE (country = 'France' OR country = 'Spain')  AND difficulty is null or difficulty = 'Moderate' |
| מציאת איש קשר להזמנה (מטיילים) | מחזירה את פרטי איש הקשר בהזמנה מסויימת. | SELECT tm.full\_name, tm.phone\_number, tm.country  FROM trip\_manager tm  JOIN trip t ON tm.country = t.country  JOIN bookings b ON t.trip\_id = b.trip\_id  WHERE b.booking\_id = 14; |
| עדכון טיול על ידי מנהל האזור (מנהל טיולים אזורי) | לעדכן את פרטי הטיול שעליו המנהל האזורי אחראי. | UPDATE trip  SET destination = <new\_destination>,  start\_date = <new\_start\_date>,  end\_date = <new\_end\_date>,  difficulty = <new\_difficulty>  WHERE trip\_id = <trip\_id>  AND country = <manager\_country>; |
| עדכון כל מחירי הטיולים במדינה מסויימת | במידה והממשלה החליטה לעלות את מחירי הטיולים במדינתה, לעלות את כולם ביחד. | UPDATE trip SET price = price \* 1.1 WHERE country = 'France'; |
| עדכון כתובת המייל של המשתמשים. | במידה והחברה רוצה שמשתמשיה ישתמשו בכתובת אימייל מטעמה. | UPDATE user  SET Email = CONCAT(full\_name, '@trailmates.com'); |
| מחיקת מנהלי אזור לא פעילים | במידה ומנהל האזור לא אחראי על אף מסלול, למחוק אותו מהמאגר. | DELETE FROM trip\_manager WHERE trip\_manager\_id NOT IN (SELECT DISTINCT trip\_manager\_id FROM trip); |
| מחיקת מדריכים לא פעילים | מחיקת מדריכים אשר לא הדריכו במשך שנה. | DELETE FROM guide  WHERE guide\_id NOT IN (  SELECT DISTINCT g.guide\_id  FROM (  SELECT guide\_id FROM guide  ) g  INNER JOIN bookings b ON g.guide\_id = b.guide\_id  WHERE(b.final\_date >= DATE\_SUB(NOW(), INTERVAL 1 YEAR) OR b.starting\_date <= DATE\_ADD(NOW(), INTERVAL 1 YEAR))  ); |

### טריגרים

|  |  |  |
| --- | --- | --- |
| שם הטריגר + טבלה | יעוד | קוד |
| Check\_booking\_date  \_and\_get  \_trip\_price(bookings) | לבדוק אם תאריך הטיול הגיוני (כלומר, לא היה בעבר). בנוסף, מחשבת את עלויות הטיול, ומכניסה לרשומה המתאימה את התוצאה. | CREATE DEFINER=`root`@`localhost` TRIGGER `check\_booking\_date\_and\_get\_trip\_price` BEFORE INSERT ON `bookings` FOR EACH ROW BEGIN    DECLARE trip\_enter\_cost DECIMAL(10,2);  DECLARE trip\_special\_equipment\_rent DECIMAL(10,2);  DECLARE trip\_guide\_cost DECIMAL(10,2);  DECLARE trip\_duration INT;  DECLARE guide\_name VARCHAR(255);  DECLARE equipment\_name VARCHAR(255);    IF NEW.starting\_date < NOW() THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Booking date must be in the future';  END IF;        SELECT enter\_cost, special\_equipment\_id INTO trip\_enter\_cost, @equipment\_id  FROM trip  WHERE trip\_id = NEW.trip\_id;  SELECT rental\_cost INTO trip\_special\_equipment\_rent  FROM equipment  WHERE equipment\_id = @equipment\_id;    SELECT price INTO trip\_guide\_cost  FROM guide  WHERE guide\_id = NEW.guide\_id;  SET trip\_duration = DATEDIFF(NEW.final\_date, NEW.starting\_date);    SET NEW.payment\_amount = trip\_enter\_cost + (trip\_special\_equipment\_rent \* trip\_duration) + (trip\_guide\_cost \* trip\_duration);  END |
| Calculate\_trip  \_length(bookings) | מחשב את הפרש הימים בין התאריך ההתחלתי לסופי, ומכניס ברשומה המתאימה את התוצאה. | CREATE DEFINER=`root`@`localhost` TRIGGER `calculate\_trip\_length` AFTER UPDATE ON `bookings` FOR EACH ROW BEGIN  IF NEW.starting\_date IS NOT NULL AND NEW.final\_date IS NOT NULL THEN  UPDATE `trip` SET `length` = DATEDIFF(NEW.final\_date, NEW.starting\_date)  WHERE `trip`.`trip\_id` = NEW.trip\_id;  END IF;  END |
| Check\_entered  \_cost(trip) | בודק אם מחיר הכניסה לטיול תקין. | CREATE DEFINER=`root`@`localhost` TRIGGER `check\_entered\_cost` BEFORE INSERT ON `trip` FOR EACH ROW BEGIN  IF NEW.enter\_cost < 0 THEN  SIGNAL SQLSTATE '45000'  SET MESSAGE\_TEXT = 'Entered cost cannot be negative';  END IF;  END |

### פרוצדורות

|  |  |  |
| --- | --- | --- |
| שם | יעוד | קוד |
| Add\_equipment | להכניס ציוד לequipment | CREATE DEFINER=`root`@`localhost` PROCEDURE `add\_equipment`(IN equipment\_name VARCHAR(255), IN rental\_cost INT)  BEGIN  INSERT INTO equipment(equipment\_name, rental\_cost)  VALUES(equipment\_name, rental\_cost);  END |
| Add\_trip | להכניס ציוד לtrip | CREATE DEFINER=`root`@`localhost` PROCEDURE `add\_trip`(  IN trip\_name VARCHAR(255),  IN duration INT,  IN length INT,  IN country VARCHAR(255),  IN difficulty VARCHAR(255),  IN enter\_cost INT,  IN special\_equipment\_id INT  )  BEGIN  DECLARE user\_id INT;  DECLARE trip\_id INT;    SET user\_id = LAST\_INSERT\_ID();    INSERT INTO trip (trip\_name, duration, length, country, difficulty, enter\_cost, special\_equipment\_id)  VALUES (trip\_name, duration, length, country, difficulty, enter\_cost, special\_equipment\_id);    SET trip\_id = LAST\_INSERT\_ID();    END |
| Add\_user\_to\_booking | לרשום משתמש להזמנה | CREATE DEFINER=`root`@`localhost` PROCEDURE `add\_user\_to\_booking`(  IN p\_user\_id INT,  IN p\_booking\_id INT  )  BEGIN  DECLARE v\_availability INT;  DECLARE v\_already\_booked INT;    -- Check if user is already signed to this booking  SELECT COUNT(\*) INTO v\_already\_booked FROM bookings\_users WHERE user\_id = p\_user\_id AND booking\_id = p\_booking\_id;    IF v\_already\_booked > 0 THEN  SELECT 'User is already signed to this booking' AS message;  ELSE  -- Check if booking is still available  SELECT availability INTO v\_availability FROM bookings WHERE booking\_id = p\_booking\_id;    IF v\_availability <= 0 THEN  SELECT 'Booking is already full' AS message;  ELSE  -- Add user to booking and decrease availability  INSERT INTO bookings\_users(user\_id, booking\_id) VALUES(p\_user\_id, p\_booking\_id);  UPDATE bookings SET availability = availability - 1 WHERE booking\_id = p\_booking\_id;  SELECT 'User added to booking' AS message;  END IF;  END IF;  END |
| calulate\_user\_stats | להביא פרטי משתמש בהתאם לID שנכנס | CREATE DEFINER=`root`@`localhost` PROCEDURE `calculate\_user\_stats`(IN p\_user\_id INT)  BEGIN  DECLARE total\_days INT DEFAULT 0;  DECLARE combined\_length INT DEFAULT 0;  SELECT SUM(DATEDIFF(b.final\_date, b.starting\_date)) INTO total\_days  FROM bookings b  INNER JOIN bookings\_users bu ON b.booking\_id = bu.booking\_id  WHERE (b.user\_id = p\_user\_id OR bu.user\_id = p\_user\_id) AND b.final\_date >= CURDATE();  SELECT SUM(t.length) INTO combined\_length  FROM trip t  INNER JOIN bookings b ON t.trip\_id = b.trip\_id  INNER JOIN bookings\_users bu ON b.booking\_id = bu.booking\_id  WHERE (b.user\_id = p\_user\_id OR bu.user\_id = p\_user\_id) AND b.final\_date >= CURDATE();  CREATE TEMPORARY TABLE IF NOT EXISTS temp\_user\_stats (  user\_id INT,  total\_days INT,  combined\_length INT  );  INSERT INTO temp\_user\_stats VALUES (  p\_user\_id,  total\_days,  combined\_length  );  SELECT \* FROM temp\_user\_stats;    DROP TEMPORARY TABLE IF EXISTS temp\_user\_stats;  END |
| delete\_unused\_equipment | למחוק ציוד לא הכרחי, שלא מופיע בשום טבלה | CREATE DEFINER=`root`@`localhost` PROCEDURE `delete\_unused\_equipment`()  BEGIN  DELETE FROM equipment WHERE equipment\_id NOT IN (SELECT DISTINCT special\_equipment\_id FROM trip WHERE special\_equipment\_id IS NOT NULL);  END |

Views:

1. CREATE VIEW popular\_destinations AS

SELECT t.country, COUNT(\*) AS total\_trips

FROM trip t

INNER JOIN bookings b ON t.trip\_id = b.trip\_id

GROUP BY t.country

ORDER BY total\_trips DESC;

1. CREATE VIEW guide\_trip\_count AS

SELECT g.guide\_id, g.full\_name, COUNT(\*) AS trip\_count

FROM guide g

INNER JOIN bookings b ON g.guide\_id = b.guide\_id

GROUP BY g.guide\_id, g.full\_name

HAVING trip\_count > 1;