WAN TEC בארי
בית ספר להנדסאים ולטכנאים
בית ברל

מגיש: אלון יערי

ת.ז: 325647170

שם המחנה: טל בר

פרטי מכללה: WAN-TEC בארי

סמל מוסד: 659284

תוכן עניינים

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 למטיילים 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.

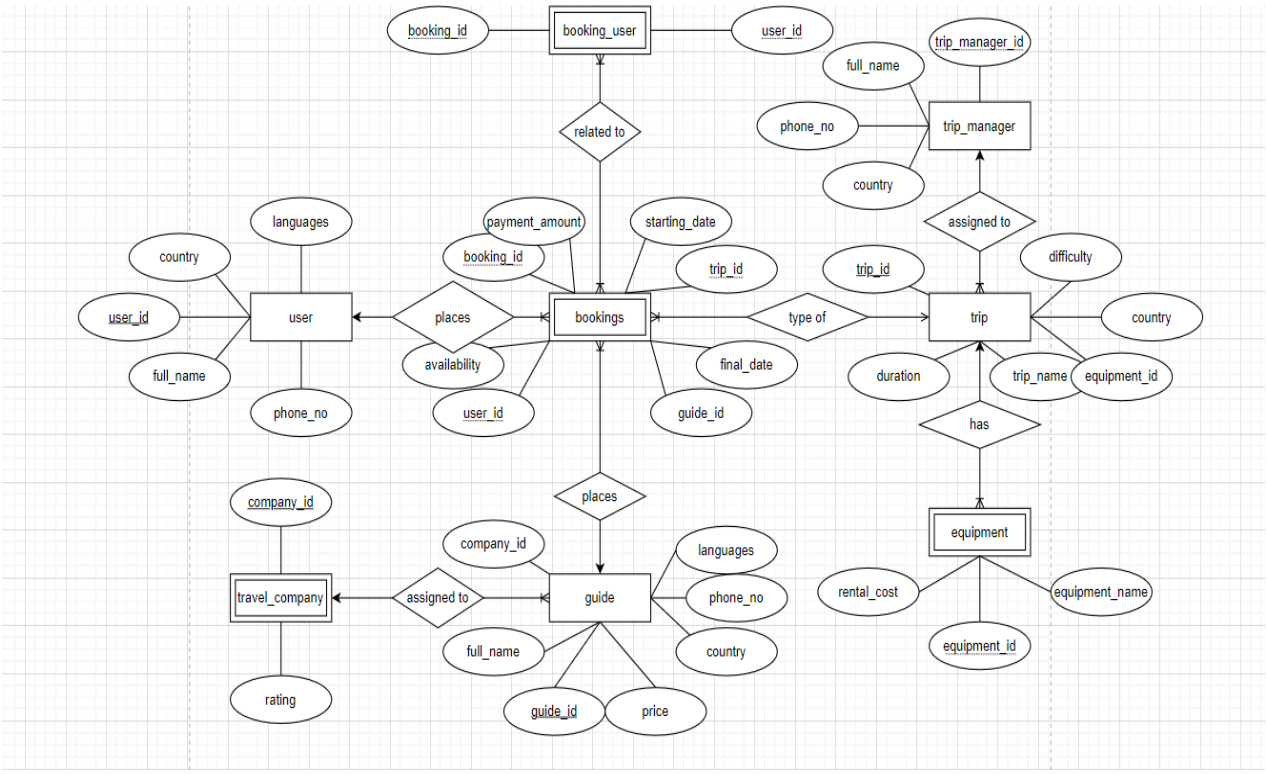
טבלאות

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.

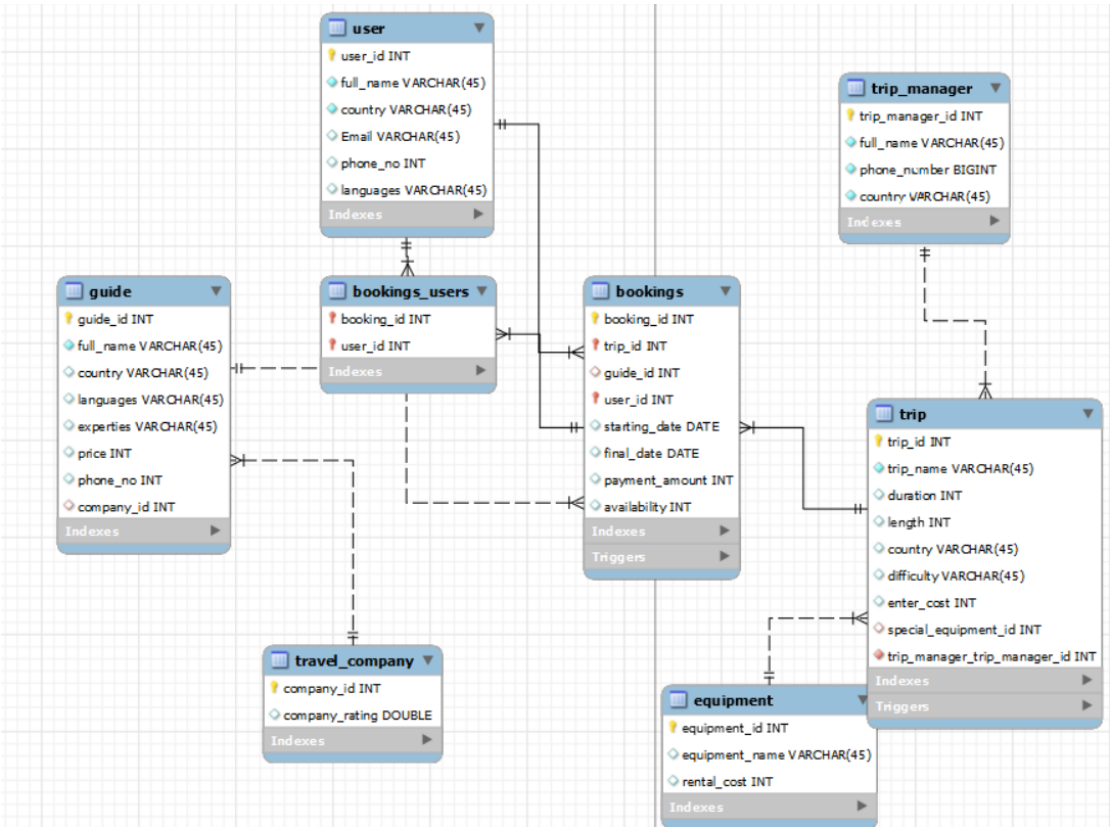
אלון יערי

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_name, COUNT(*) AS num_users FROM bookings INNER JOIN trip ON bookings.trip_id = trip.trip_id GROUP BY 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		

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;
	לרשום משתמש להזמנה	Add_user_to_booking

		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;

2. 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;