מיני פרויקט בבסיסי נתונים : מחלקת ניהול נכסי ואולמות הקולנוע

יעקב פולק וסנדי פרדס

תיאור הפרויקט: הקבוצה בחרה לבנות בסיס נתונים של רשת בתי קולנוע. אנחנו נתמקד בטבלאות לניהול הנכסים שזה כולל את סניפי הקולנוע, אולמות הקולנוע והמושבים בתוך האולמות.

בבסיס יהיו לנו 6 טבלאות כמו שיפורט:

<u>טבלאות:</u>

טבלת הסניפים (cinemas):

טבלה זאת תכיל את הנתונים על הסניפים שיש לרשת בתי הקולנוע שלנו. המידע בטבלה יכיל את הנתונים על הנכסים שזה המיקום, השם ועוד מידע כללי. כמובן שיש גם מידע שלא ממש נשתמש בו לניהול הנכסים אבל זה מידע שנצרך לארגון וזאת הטבלה הכי הגיונית לשמור בה את המידע הזה.

Column	Type	Description
cinema_id (key)	int	מכיל את המיקוד של הנכס לפי משרד הפנים. המיקוד הוא יחיד בכל הארץ ולכן משמש כמפתח
cinema_name	varchar	שם הסניף כמו שנהוג שנותנים כינוי לסניף של קולנוע
city	varchar	שם העיר שבה נמצא הסניף
address	varchar	כתובת של הסניף
establishment	date	שנת הקמת הסניף

על מנת ליצור את הטבלה הזאת באמצעות phpMyAdmin השתמשנו בקוד הבא:

```
CREATE TABLE `id20366218_sqlproject`.`cinemas` (
 `cinema_id` INT NOT NULL ,
 `cinema_name` VARCHAR(25) NOT NULL,
`city` VARCHAR(25) NOT NULL,
 `address` VARCHAR(25) NOT NULL,
`establishment` DATE NOT NULL
PRIMARY KEY (`cinema_id`)
) ENGINE = InnoDB;
```

טבלת החדרים (rooms):

טבלה זאת תכיל את הנתונים על חדר הקרנות בקולנוע. בטבלה זאת יהיו לנו שני מפתחות היות וניתן שיהיה אותו מספר חדר בשני סניפים שונים של הרשת שלנו. בנוסף נצרך כאן ש-cinema_id

יהיה מפתח זר מטבלת cinemas על מנת לדאוג שיהיו לנו נתונים אמיתיים ולא יהיה מצב שנכניס חדר לסניף שלא קיים. בנוסף עבור העמודה is_imax אז טיפוס המשתנה הוא tinyint בגודל 1 שזה כמו משתנה בוליאני שלא קיים בphpMyAdmin.

Column	Type	Description
cinema_id(key)	int	מכיל את המיקוד של הנכס לפי משרד הפנים. המיקוד הוא יחיד בכל הארץ ולכן משמש כמפתח
room_id(key)	int	מכיל את מספר החדר כאשר המספר הראשון זה הקומה בנכס והשאר זה החדר בקומה
room_name	varchar	שם האולם
num_seats	int	מספר הכיסאות
num_rowes	int	מספר השורות
is_imax	tinyint	משתנה בוליאני האם קיימת אפשרות להקרנה של imax

על מנת ליצור את הטבלה הזאת באמצעות phpMyAdmin השתמשנו בקוד הבא:

```
CREATE TABLE `id20366218_sqlproject`.`rooms` (
 `cinema_id` INT NOT NULL ,
 `room_id` INT NOT NULL ,
 `room_name` VARCHAR(256)  NULL ,
 `num_seats` INT NOT NULL ,
 `num_rowes` INT NOT NULL,
 `is_imax` TINYINT NULL,
PRIMARY KEY (`cinema_id`, `room_id')
) ENGINE = InnoDB;
```

והקוד ליצירת המפתח זר:

```
ALTER TABLE `rooms` ADD FOREIGN KEY
(`cinema_id`)
REFERENCES
`cinemas`(`cinema_id`)
ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

טבלת מושבים (seat):

בטבלה זאת יהיה את הנתונים על כל המושבים לפי בתי הקולנוע. גם כאן נצטרך לדאוג לכך שהמספר מזהה של הסניף והמספר מזהה של החדר יהיו מפתחות זרים על מנת שלא יהיו מצבים לא תקינים. גם כאן את העמודה של is_accessibility נגדיר מטיפוס tinyint והיות ורוצים שיהיה

בוליאני. המפתח כאן יהיה המספר מושב ביחד עם מספר הסניף היות ובכל סניף לא ניתן שיהיו שני מושבים עם אותו מספר מזהה.

Column	Type	Description
Seat_id(key)	int	מספר זיהוי של המושב.
cinema_id(key)	int	מכיל את המיקוד של הנכס לפי משרד הפנים. המיקוד הוא יחיד בכל הארץ ולכן משמש כמפתח.
room_id	int	מכיל את מספר החדר כאשר המספר הראשון זה הקומה בנכס והשאר זה החדר בקומה.
row	int	מספר שורה מקדמת האולם של המושב.
seat	int	מספר המושב מהמעבר.
is_accessibility	tinyint	האם המושב נגיש לנכים.

על מנת ליצור את הטבלה הזאת באמצעות phpMyAdmin השתמשנו בקוד הבא:

```
CREATE TABLE `id20366218_sqlproject`.`seat` (
`seat_id` INT NOT NULL ,
`cinema_id` INT NOT NULL ,
`room_id` INT NOT NULL ,
 `row` INT NOT NULL ,
 `seat` INT NOT NULL ,
 `is_accessibility` TINYINT(1) NOT NULL ,
PRIMARY KEY (`seat_id`, `cinema_id')
) ENGINE = InnoDB;
```

הקוד ליצירת המפתחות הזרים מהטבלה של rooms:

```
ALTER TABLE `seat` ADD FOREIGN KEY
(`cinema_id`, `room_id`)
REFERENCES
`rooms`(`cinema_id`, `room_id`)
ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

טבלת הזמנות (bookings):

בטבלה הזאת נשמור את הנתונים על הזמנות של חדרי קולנוע לאירועים פרטיים. נשמור כאן כמובן את פרטי החדר ועוד פרטים על ההזמנה כמו השעות ומספר ההזמנה. בטבלה הזאת המפתחות יהיו מספר ההזמנה ומספר הסניף היות ומספר הזמנה לבד לא יספיק כי בשני סניפים ניתן שיהיה

את אותו מספר הזמנה אבל באותו סניף לא ולכן מספיק את שניהם בתור מפתח ולא צריך את המספר חדר גם.

Column	Type	Description
booking_id(key)	int	מספר ההזמנה. כל סניף מנהל את המספור בפני עצמו
cinema_id(key)	int	מכיל את המיקוד של הנכס לפי משרד הפנים. המיקוד הוא יחיד בכל הארץ ולכן משמש כמפתח
room_id	int	מכיל את מספר החדר כאשר המספר הראשון זה הקומה בנכס והשאר זה החדר בקומה
date	date	תאריך ההזמנה
start_time	time	שעת תחילת האירוע
end_time	time	שעת סיום האירוע
movie_name	varchar	שם הסרט שיוקרן באירוע

על מנת ליצור את הטבלה הזאת באמצעות phpMyAdmin השתמשנו בקוד הבא:

```
CREATE TABLE `id20366218_sqlproject`.`bookings` (
 `booking_id` INT NOT NULL ,
 `cinema_id` INT NOT NULL ,
 `room_id` INT NOT NULL ,
 `date` DATE NOT NULL ,
 `start_time` TIME NOT NULL ,
`end_time` TIME NOT NULL ,
 `movie_name` VARCHAR(256) NOT NULL ,
PRIMARY KEY (`booking_id`, `cinema_id`)
) ENGINE = InnoDB;
```

הקוד ליצירת המפתחות הזרים מהטבלה של rooms:

```
ALTER TABLE `bookings` ADD FOREIGN KEY
(`cinema_id`, `room_id`)
REFERENCES
`rooms`(`cinema_id`, `room_id`)
ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

טבלת הקרנות (projection):

טבלה זאת תכיל את הנתונים על הקרנות של סרטים כמו שיש בפועל בכל בית קולנוע את ההקרנות מראש. כאן ההתמקדות זה לא הסרטים אלא המיקום וניהול חדרי ההקרנות ואורכי הסרטים. גם כאן צריך את המספר הקרנה והמספר סניף שיהיו המפתחות מאותו הסבר של הטבלה של הזמנות.

Column	Type	Description
projection_id (key)	int	מספר הקרנה. כל סניף מנהל את המספור בפני עצמו
cinema_id (key)	int	מכיל את המיקוד של הנכס לפי משרד הפנים. המיקוד הוא יחיד בכל הארץ ולכן משמש כמפתח
room_id	int	מכיל את מספר החדר כאשר המספר הראשון זה הקומה בנכס והשאר זה החדר בקומה
date	date	תאריך ההקרנה
start_time	time	שעת תחילת ההקרנה
end_time	time	שעת סיום ההקרנה
movie_name	varchar	שם הסרט שיוקרן

על מנת ליצור את הטבלה הזאת באמצעות phpMyAdmin השתמשנו בקוד הבא:

```
CREATE TABLE `id20366218_sqlproject`.`projection` (
 `projection_id` INT NOT NULL ,
 `cinema_id` INT NOT NULL ,
 `room_id` INT NOT NULL ,
 `movie_name` INT NOT NULL ,
 `date` DATE NOT NULL ,
 `start_time` TIME NOT NULL ,
PRIMARY KEY (`projection_id`, `cinema_id')
 `end_time` TIME NOT NULL ) ENGINE = InnoDB;
```

הקוד ליצירת המפתחות הזרים מהטבלה של rooms:

```
ALTER TABLE `projection` ADD FOREIGN KEY
(`cinema_id`, `room_id`)
REFERENCES
`rooms`(`cinema_id`, `room_id`)
ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

טבלת כרטיסים (tickets):

נעשה טבלה נוספת שתכיל את הכרטיסים שנמכרו. המפתח יהיה מספר כרטיס ואת מספר הסניף(היות ולכל סניף יש מספור כרטיסים עצמי). בנוסף במקום לשמור שוב את כל פרטי המושב נשמור רק את המספר מזהה שלו ואז נוכל לקבל את כל הפרטים שלו מהטבלה של המושבים.

Column	Type	Description
ticket_id (key)	int	מכיל את המספר המזהה של הכרטיס
cinema_id (key)	int	מכיל את המיקוד של הנכס לפי משרד הפנים. המיקוד הוא יחיד בכל הארץ ולכן משמש כמפתח
room_id	int	מכיל את מספר החדר כאשר המספר הראשון זה הקומה בנכס והשאר זה החדר בקומה
client_name	varchar	מכיל את שם הלקו
movie_name	varchar	מכיל את שם הסרט
seat_id	int	מכיל את המספר הסידורי של המושב
date	date	תאריך הקרנת הסרט

על מנת ליצור את הטבלה הזאת באמצעות phpMyAdmin השתמשנו בקוד הבא:

```
CREATE TABLE `id20366218_sqlproject`.`tickets` (
 `ticket_id` INT NOT NULL ,
 `cinema_id` INT NOT NULL ,
 `room_id` INT NOT NULL ,
 `client_name` VARCHAR(256) NOT NULL ,
 `movie_name` VARCHAR(256) NOT NULL ,
 `seat_id` INT NOT NULL ,
 `date` DATE NOT NULL ,
 PRIMARY KEY (`ticket_id`, `cinema_id`)
) ENGINE = InnoDB;
```

הקוד ליצירת המפתחות הזרים מהטבלה של rooms:

```
ALTER TABLE `tickets` ADD FOREIGN KEY
(`cinema_id`, `room_id`, `seat_id`)
 REFERENCES
 `seat`(`cinema_id`, `room_id`, `seat_id`)
 ON DELETE RESTRICT
ON UPDATE RESTRICT;
```

שאילתות ופרוצדורות:

Booked_rooms view:

על מנת שיהיה נוח לדעת איזה חדרים תפוסים לאירוע או להקרנה אז ניצור view שבעצם יכיל את הנתונים הנדרשים שזה המספר סניף, מספר חדר, תאריך ושעות.

הקוד ליצירת ה-view:

```
CREATE VIEW booked_rooms AS
SELECT cinema_id, room_id, date, start_time, end_time
FROM (
SELECT cinema_id, room_id, date, start_time, end_time
FROM bookings
UNION ALL
SELECT cinema_id, room_id, date, start_time, end_time
FROM projection )
AS booked_rooms
ORDER BY date, start_time
```

הצגה איך יופיע בבסיס נתונים:

cinema_id	room_id	date	start_time	end_time
6	8	1950-09-04	08:00:00	10:00:00
10	8	1950-09-04	08:00:00	10:00:00
4	9	1950-09-04	10:00:00	13:00:00
6	4	1950-09-04	12:00:00	13:00:00
1	4	1951-07-25	10:00:00	11:00:00
8	1	1951-07-25	10:00:00	11:00:00
7	2	1951-07-25	12:00:00	13:00:00
5	10	1951-07-25	13:00:00	15:00:00
8	10	1953-04-25	10:00:00	12:00:00
10	9	1953-04-25	14:00:00	17:00:00
3	2	1954-09-08	12:00:00	13:00:00
6	4	1957-05-14	11:00:00	14:00:00

שאילתא למציאת כמות מושבים נגישים בכל אולם:

על מנת שנוכל לדעת כמה מושבים נגישים יש בכל אולם למשל עבור הזמנת האולם לאירוע אז נעשה שאילתא שתחזיר לנו את כמות המושבים הנגישים בכל אולם.

הקוד לשאילתא:

```
SELECT cinema_id, room_id,
COUNT(*) AS num_accessible_seats
FROM seat
WHERE is_accessibility = true
GROUP BY cinema_id, room_id
```

הצגה איך יופיע בבסיס נתונים:

cinema_id	room_id	num_accessible_seats
1	1	18
1	2	12
1	3	12
1	4	19
1	5	15
1	6	8
2	1	15
2	2	16
2	3	16
2	4	13

שאילתא לקבלת שם הקולנוע עם הכי הרבה הזמנות:

במידה ונרצה לדעת איזה מסניפי הקולנוע עם הכי הרבה הזמנות אז נוכל להשתמש בשאילתא הבאה על מנת לקבל את השם של הסניף הזה.

במידה ונרצה לקבל את דירוג כל הסניפים לפי סדר כמות ההזמנות ניתן פשוט להוריד מהשאילתא את השורה של LIMIT 1 ונקבל את כולם.

הקוד לשאילתא:

```
SELECT c.cinema_name,
COUNT(DISTINCT b.booking_id) + COUNT(DISTINCT p.projection_id)
AS total_events
FROM cinemas c
LEFT JOIN rooms r ON c.cinema_id = r.cinema_id
LEFT JOIN bookings b ON r.room_id = b.room_id
```

```
AND c.cinema_id = b.cinema_id
LEFT JOIN projection p ON r.room_id = p.room_id
AND c.cinema_id = p.cinema_id
GROUP BY c.cinema_name
HAVING COUNT(DISTINCT b.booking_id) +
COUNT(DISTINCT p.projection_id) > 0 ORDER BY total_events DESC
LIMIT 1;
```

הצגה איך יופיע בבסיס נתונים:

cinema_name	total_events
Cinema 1	129

במידה ונוריד את ההגבלה נקבל עבור כל סניף את מספר האירועים באופן הבא:

cinema_name	total_events
Cinema 1	129
Cinema 2	124
Cinema 4	114
Cinema 7	114
Cinema 3	114
Cinema 8	113
Cinema 6	109
Cinema 9	107
Cinema 10	106
Cinema 5	105
Cinema 11	102
Cinema 12	102

שאילתא לקבלת מושבים פנויים לפי סניף וסרט:

על מנת שנוכל לדעת אילו מושבים פנויים עבור סרט מסוים כמו למשל בהזמנה של לקוחות באתר הקולנוע שיוכלו לקבל לפי שם הקולנוע, שם הסרט והתאריך את כל אופציית המושבים הפנויים.

בקוד הנתון הוכנס תאריך, שם סרט ושם סניף. כמובן שזה ניתן לשינוי לפי הנצרך.

הקוד לשאילתא:

```
SELECT DISTINCT c.cinema_name, p.date, p.movie_name, r.room_name,
 s.row, s.seat
 FROM bookings b
 JOIN cinemas c ON b.cinema_id = c.cinema_id
 JOIN rooms r ON b.cinema_id = r.cinema_id AND b.room_id = r.room_id
 JOIN projection p ON b.cinema_id = p.cinema_id
```

```
 AND b.room_id = p.room_id
 JOIN seat s ON r.cinema_id = s.cinema_id AND r.room_id = s.room_id
 LEFT JOIN tickets t ON b.cinema_id = t.cinema_id
 AND b.room_id = t.room_id AND t.date = p.date
 AND t.seat_id = s.seat_id
WHERE p.date = '2021-04-16'
 AND p.movie_name = 'Movie 9' AND t.ticket_id IS NULL
ORDER BY r.room_name, s.row, s.seat;
```

לצורך נוחות לשימוש ניצור פרוצדורה שתקבל מהמשתמש את מספר הסניף, התאריך ואת שם הסרט ותחזיר טבלה עם כל המקומות הפנויים.

הקוד ליצירת הפרוצדורה:

```
CREATE PROCEDURE GetFreeSeats(
 IN p_cinema_id INT,
 IN p_movie_name VARCHAR(100),
 IN p_date DATE )
BEGIN
 SELECT c.cinema_name, p.date, p.movie_name, r.room_name,
 s.row, s.seat
 FROM bookings b
 JOIN cinemas c ON b.cinema_id = c.cinema_id
 JOIN rooms r ON b.cinema_id = r.cinema_id AND b.room_id = r.room_id
  JOIN projection p ON b.cinema_id = p.cinema_id AND
 b.room_id = p.room_id
JOIN seat s ON r.cinema_id = s.cinema_id AND r.room_id = s.room_id
LEFT JOIN tickets t ON b.cinema_id = t.cinema_id AND
 b.room_id = t.room_id AND t.date = p.date AND t.seat_id = s.seat_id
 WHERE p.cinema_id = p_cinema_id AND p.movie_name = p_movie_name
AND p.date = p_date AND t.ticket_id IS NULL
ORDER BY r.room_name, s.row, s.seat;
END;
```

בהרצת השאילתא עם הפרמטרים שכתובים בבסיס הנתונים נקבל:

cinema_name	date	movie_name	room_name ▲ 1	row	seat
Cinema 6	2021-04-16	Movie 9	Room sx8XLAS	1	1
Cinema 6	2021-04-16	Movie 9	Room sx8XLAS	1	2
Cinema 6	2021-04-16	Movie 9	Room sx8XLAS	1	3
Cinema 6	2021-04-16	Movie 9	Room sx8XLAS	1	4
Cinema 6	2021-04-16	Movie 9	Room sx8XLAS	1	5
Cinema 6	2021-04-16	Movie 9	Room sx8XLAS	1	6

שאילתא לבדיקת נצילות האולמות להקרנות שהיו בעבר:

השאילתא הבאה נותנת לנו טבלה שתכיל את הפרטים על הקרנות שבוצעו בעבר וניתן היה להקרין אותם באולם אחר שהיה מכיל פחות מושבים וכבה לא להפעיל אולם גדול. הנתונים זה על הקרנות שבוצעו בעבר וישמש את הנהלת הקולנוע לסטטיסטיקות ותכנון עתידי של הקרנות.

הקוד לשאילתא:

```
SELECT p.projection_id, p.cinema_id, p.room_id, MAX(p.date) AS date, MAX(p.start_time)
 AS start_time, MAX(p.end_time) AS end_time, MAX(p.movie_name) AS movie_name,
 MAX(r.num_seats - COALESCE(t.num_tickets, 0)) AS empty_seats,
 MAX(r2.room_id) AS alternative_room_id
FROM projection p
JOIN rooms r ON p.cinema_id = r.cinema_id AND p.room_id = r.room_id
LEFT JOIN (
  SELECT cinema_id, room_id, COUNT(*) AS num_tickets
  FROM tickets
  WHERE date < CURDATE()
  GROUP BY cinema_id, room_id
) t ON p.cinema_id = t.cinema_id AND p.room_id = t.room_id
JOIN rooms r2 ON p.cinema_id = r2.cinema_id AND p.room_id != r2.room_id AND
r2.num_seats >= (COALESCE(t.num_tickets, 0)) AND r2.num_seats < r.num_seats AND
 (t.num_tickets IS NULL)
WHERE p.date < CURDATE()
GROUP BY p.projection_id
ORDER BY MAX(p.date) DESC, MAX(p.start_time) DESC
```

הצגה איך יופיע בבסיס נתונים:

projection_id	cinema_id	room_id	date	start_time	end_time	movie_name	empty_seats	alternative_room_id
54158	7	1	2023-06-16	10:00:00	12:00:00	Movie 9	100	9
17430	9	1	2023-04-12	10:00:00	12:00:00	Movie 1	110	10
358334	9	5	2023-03-13	18:49:31	20:49:31	Movie 600	80	8
609891	4	9	2023-03-13	18:49:31	20:49:31	Movie 464	120	10
637439	5	7	2023-03-13	18:49:31	20:49:31	Movie 781	100	10
659074	11	5	2023-03-13	18:49:31	20:49:31	Movie 111	80	10
381477	2	7	2023-03-13	18:49:31	20:49:31	Movie 784	70	2
572229	7	9	2023-03-13	18:49:31	20:49:31	Movie 806	90	8
776220	6	9	2023-03-13	18:49:31	20:49:31	Movie 156	100	10
223836	3	10	2023-03-13	18:49:31	20:49:31	Movie 339	70	9
317485	10	9	2023-03-13	18:49:31	20:49:31	Movie 13	80	10
874386	7	2	2023-03-13	18:49:31	20:49:31	Movie 93	110	10

שאילתות אינטגרציה:

בשלב זה של הפרויקט קיבלנו טבלאות של המזנון של הקולנוע.

מבנה הטבלאות שקיבלנו הוא טבלה מרכזית לכל העובדים ואז טבלאות לעובד שהוא מוכר ולעובד שהוא איש תחזוקה. בנוסף יש טבלאות לספקים ולמוצרים וטבלאות המקשרות בין הרכיבים.

על מנת להתאים את הטבלאות שקיבלנו אז הוספנו לטבלה workers עמודה נוספת על מנת שנוכל לקשר בין עובד לסניף. לאחר שקיים הקישור הזה אז נוכל להתחיל לבצע שאילתות. שינוי נוסף שביצענו זה להוסיף עמודה של שעה לטבלת orders על מנת שנוכל לקבל אינדיקציה לקניות במזנון ביחס לסרטים המוקרנים באותה השעה באותו הסניף. הוספנו גם עמודה של מספר הסניף לכל הזמנה היות ולא נוכל לדעת את זה מפרטי העובד שהרי הגדרנו שעובד יכול לעבוד בכמה סניפים.

בנוסף להתאמות שנדרשו בעמודות נדרשו גם התאמות בשורות על מנת שנקבל תוצאות כאשר נבצע שאילתות ולכן היה צורך בהכנסת נתונים נוספים על מנת שתהיה התאמה.

שינוי נוסף שביצענו זה שהמפתח לטבלה workers כעת יהיה גם תעודת הזהות של העובד וגם מספר הסניף. הסיבה שנצרכנו לשינוי זה היא על מנת שעובד אחד יוכל לעבוד ביותר מסניף אחד.

שאילתא לקבלת סכום מכירות פר סרט:

על מנת לדעת כמה מכירות התבצעו במזנון עבור כל סרט אז לא נוכל לדעת במדויק היות ולא חייבנו הצגת כרטיס בביצוע קניה(בדומה לשדה תעופה) אלא נעשה בדיקה של סכום כל המכירות שהתבצעו בזמן ההקרנה עם הוספה של חצי שעה קודם ההקרנה. במידה ונרצה נתונים יותר מדויקים ניתן לבצע חישובים וסטטיסטיקות נוספות על התוצאה לפי הנדרש(במקרה כזה נהפוך את השאילתא להיות VIEW).

הקוד לשאילתא:

```
SELECT p.cinema_id, p.room_id, p.projection_id, p.movie_name,
 SUM(o.total_price) AS total_sales
FROM projection AS p
JOIN orders AS o ON p.date = o.date
```

```
JOIN seller_order AS so ON o.order_number = so.order_number
JOIN workers AS w ON so.i_d = w.i_d AND p.cinema_id = w.cinema_id
WHERE o.time >= DATE_SUB(p.start_time, INTERVAL 30 MINUTE) - Order
 time should be greater than projection start time minus half an
hour
AND o.time <= p.end_time -- Order time should be smaller than
projection end time
GROUP BY p.movie_name, p.cinema_id, p.room_id;
```

הצגה איך יופיע בבסיס נתונים:

cinema_id	room_id	projection_id	movie_name	total_sales
8	1	173226	Movie 114	25538
1	10	704880	Movie 122	300
8	5	435178	Movie 122	25538
10	9	317485	Movie 13	300
1	9	302819	Movie 130	300
12	2	519731	Movie 137	43810
6	10	96450	Movie 160	55568
12	10	929350	Movie 167	57313

שאילתא שתיתן לכל מוצר כמות מכירות לכל סניף:

במידה ונרצה לדעת כל מוצר כמה נמכר ממנו בכל אחד מהסניפים אז נבצע שאילתא שבודקת לכל מוצר לפי ההזמנות שלו ובודקת מי היה המוכר של ההזמנה ולפי זה לדעת באיזה סניף בוצעה המכירה ונחזיר בטבלה את סכום המכירות.

הקוד לשאילתא:

```
SELECT p.product_code, c.cinema_id, c.cinema_name,
 COUNT(op.amount) AS sales_count
FROM product p
JOIN order_product op ON p.product_code = op.product_code
JOIN orders o ON op.order_number = o.order_number
JOIN seller_order so ON o.order_number = so.order_number
JOIN sellers s ON so.i_d = s.i_d
JOIN workers w ON s.i_d = w.i_d
JOIN cinemas c ON w.cinema_id = c.cinema_id
```

```
GROUP BY p.product_code, c.cinema_id;
```

הצגה איך יופיע בבסיס נתונים:

product_code	cinema_id	cinema_name	sales_count
AB-118	9	Cinema 9	2
AB-118	12	Cinema 12	2
AC-375	3	Cinema 3	2
AC-375	4	Cinema 4	2
AC-375	7	Cinema 7	2
AC-375	8	Cinema 8	4

פרוצדורה שתקבל מספר סניף ותחזיר את כל הספקים המספקים מוצרים שנמכרים בסניף:

כאן אנחנו יוצרים פרוצדורה שתקבל כפרמטר את מספר הסניף שעליו נרצה לדעת את המידע ונקבל בחזרה את רשימת הספקים המספקים מוצרים שנמכרים בסניף הנתון. בנוסף לפרטי הספק אז אנחנו מוסיפים עמודה שתכיל את כמות המוצרים שאותו ספק מספק לאותו סניף.

הקוד ליצירת הפרוצדורה:

```
CREATE PROCEDURE GetSuppliersForCinema(IN p_cinema_id INT)
BEGIN
SELECT s.*, COUNT(DISTINCT op.product_code) AS num_items_sold
FROM
suppliers AS s
INNER JOIN supplier_product AS sp ON s.i_d = sp.i_d
INNER JOIN product AS p ON sp.product_code = p.product_code
INNER JOIN order_product AS op ON p.product_code = op.product_code
INNER JOIN seller_order AS so ON op.order_number = so.order_number
INNER JOIN workers AS w ON so.i_d = w.i_d
WHERE w.cinema_id = p_cinema_id
GROUP BY s.i_d;
END;
```

כעת כאשר נרצה להשתמש בפרוצדורה נצטרך לקרוא לה עם פרמטר בצורה הבאה:

```
CALL GetSuppliersForCinema(1);
```

בדוגמא כאן זימנו את הפרוצדורה עם פרמטר 1 כלומר שנקבל את כל הספקים שמספקים לסינמה 1 ואת כמות הפריטים.

נראה דוגמא להרצה:

Showing rows 0 - 24 (71 total, Query took 0.0034 seconds.)

```
CALL GetSuppliersForCinema(4);
```

[Edit inline] [Edit] [Create PHP code]

1 > >> | Show all | Number of rows: 25 Filter rows: Se

Extra options

i_d	seniority	phone_number	first_name	last_name	num_items_sold
501166920	38	050-6246295	Vince	Hughes	1
504519245	44	050-6796547	Chubby	Ramirez	1
505943930	2	050-6463139	John	Chan	1
507253123	8	050-6035086	Kurt	Kotto	1
509595867	34	050-6154936	Mary-Louise	Roberts	1

פרוצדורה להכנסת הזמנה חדשה לטבלת ההזמנות:

יש לנו טבלה של orders שבעצם מכילה את כל ההזמנות שבוצעו במזנון. במידה ונרצה להוסיף הזמנה חדשה(דבר שכל הזמן מתבצע) אז יצרנו כאן פרוצדורה שתקבל את פרטי ההזמנה ואת פרטי המוכר ותבצע קודם בדיקה שהנתונים נכונים(שהמספר זהות זה של עובד שהוא גם מוכר באותו הסניף). העדכון מתבצע בהתאם לתוצאת הבדיקה ונשלחת למשתמש הודעה מתאימה.

הקוד ליצירת הפרוצדורה:

```
CREATE PROCEDURE InsertOrder(
  IN p_order_number INT,
  IN p_date DATE,
  IN p_total_price DECIMAL(10,2),
  IN p_ordering_phone VARCHAR(10),
  IN p_time TIME,
  IN p_seller_id INT,
  IN p_cinema_id  INT
)
BEGIN
  DECLARE v_is_worker INT;
  -- Check if the seller is a worker in the cinema
  SELECT COUNT(*) INTO v_is_worker
  FROM workers AS wo
  INNER JOIN sellers AS se ON wo.i_d = se.i_d
  WHERE wo.i_d = p_seller_id AND wo.cinema_id = p_cinema_id;
  IF v_is_worker > 0 THEN
    -- Seller is a worker in the cinema, proceed with the insertion
      INSERT INTO orders (order_number, date, total_price,
                          ordering_phone, time, cinema_id)
```

```
        VALUES (p_order_number, p_date, p_total_price, p_ordering_phone
, p_time, p_cinema_id);
 -- Insert any additional logic or actions you require for the order
        SELECT 'Order inserted successfully.' AS message;
   ELSE
        SELECT 'Seller is not a worker in the cinema.' AS message;
   END IF;
END;;
```

כעת כאשר נרצה להשתמש בפרוצדורה נצטרך לקרוא לה עם פרמטר בצורה הבאה:

```
CALL InsertOrder(
123,--order_number
 '2023-06-07',--date
 100.00,--total_price
 '1234567890',--ordering_phone
 '10:00:00',--time
 456,--id
 789);--cinema_id
```

יצירת View שתביא עבור כל סניף את העובד המצטיין:

במידה ונרצה לקבל עבור כל סניף את העובד שביצע מכירות בהכי הרבה כסף אז נבצע את השאילתא הבאה. ניצור את זה בתור View על מנת שיוצג תמיד עבור דרבון העובדים וכדומה שיוצג תמיד העובד במקום הראשון.

הקוד ליצירת הView:

```
CREATE VIEW cinema_max_sales AS
SELECT c.cinema_name, w.first_name, w.last_name, total_sales
FROM cinemas AS c
JOIN workers AS w ON c.cinema_id = w.cinema_id
JOIN (
     SELECT so.i_d, SUM(o.total_price) AS total_sales
     FROM seller_order AS so
     JOIN orders AS o ON so.order_number = o.order_number
     GROUP BY so.i_d
) AS s ON w.i_d = s.i_d
```

```sql
WHERE s.total_sales = (
      SELECT MAX(total_sales)
      FROM (
        SELECT SUM(o.total_price) AS total_sales
        FROM seller_order AS so
        JOIN orders AS o ON so.order_number = o.order_number
        GROUP BY so.i_d
      ) AS t
      WHERE t.total_sales = s.total_sales
)
GROUP BY c.cinema_id;
```

הצגה איך יופיע בבסיס נתונים:

cinema_name	first_name	last_name	total_sales
Cinema 1	Jody	Melvin	62868
Cinema 2	Russell	Bale	95981
Cinema 3	Diane	Sinise	7780
Cinema 4	Ray	Harnes	110928
Cinema 5	Freddy	Kahn	19143
Cinema 6	Barry	Hagar	159454
Cinema 7	Albertina	MacPherson	65801
Cinema 8	Scott	Kretschmann	69183
Cinema 9	Diane	Sinise	7780
Cinema 10	Maura	England	93632

פרוצדורה להכנסת עובד חדש:

ניצור פרוצדורה על מנת לפשט את תהליך הוספת עובד לבסיס הנתונים. בעת הוספת עובד נבדוק שהוא לא קיים כבר ונכניס אותו לסוג העובד הרצוי.

```sql
CREATE PROCEDURE CreateWorker11(
  IN p_id INT,
  IN p_cinema_id INT,
  IN p_first_name VARCHAR(50),
  IN p_last_name VARCHAR(50),
  IN p_phone_number VARCHAR(20),
  IN p_address VARCHAR(100),
  IN p_year_of_birth INT,
```

```
  IN p_worker_type VARCHAR(50),
  IN p_salary DECIMAL(10,2)
)
BEGIN
 DECLARE v_worker_count INT;
 -- Check if the worker already exists in the table
  SELECT COUNT(*) INTO v_worker_count
  FROM workers
  WHERE i_d = p_id AND cinema_id = p_cinema_id;
  IF v_worker_count > 0 THEN
    SELECT 'Worker already exists.' AS message;
  ELSE -- Insert the worker into the workers table
    INSERT INTO workers (i_d, cinema_id, first_name, last_name,
phone_number, address, year_of_birth)
    VALUES (p_id, p_cinema_id, p_first_name, p_last_name,
p_phone_number, p_address, p_year_of_birth);
  -- Insert the worker into the appropriate type table (seller,
maintenance, etc.)
  IF p_worker_type = 'seller' THEN
    INSERT INTO sellers (i_d, salary)
    VALUES (p_id, p_salary);
  ELSEIF p_worker_type = 'maintenance' THEN
    INSERT INTO maintenance (i_d, salary)
    VALUES (p_id, p_salary);
  END IF;
  SELECT 'Worker created successfully.' AS message;
END;;;
```

כעת כאשר נרצה להשתמש בפרוצדורה נצטרך לקרוא לה עם פרמטר בצורה הבאה:

```
CALL CreateWorker11(546444, 12, 'John', 'Doe', '1234567890', '123
Main St', 1990, 'seller', 5000.00);
```

יצירת Trigger שתמנע הכנסה של שתי הזמנות ע"י אותו עובד בשני סניפים:

היות והגדרנו שעובד יכול לעבוד ביותר מסניף אחד אז על מנת שלא ייווצר מצב שבו אותו עובד יוכל להכניס שני הזמנות בו זמנית בשני סניפים שונים אז יצרנו Trigger שבעצם בודק בעת הכנסת הזמנה חדשה לטבלת orders שלא בוצעה בשעה האחרונה הזמנה של אותו עובד בסניף אחר.

הקוד ליצירת הטריגר:

```
CREATE TRIGGER prevent_duplicate_orders
```

```
BEFORE INSERT ON orders
FOR EACH ROW
BEGIN
  DECLARE seller_id INT;
  SET seller_id = (SELECT i_d FROM seller_order WHERE order_number =
  NEW.order_number);
  IF EXISTS (
   SELECT 1
     FROM orders AS ord
     JOIN seller_order AS seo
     ON ord.order_number = seo.order_number
     JOIN workers AS wo
     ON ord.i_d = wo.i_d
     WHERE
     wo.cinema_id <> NEW.cinema_id
     AND ord.date = NEW.date
     AND ord.time >= DATE_SUB(NEW.time, INTERVAL 1 HOUR)
     AND ord.time <= NEW.time AND seo.i_d = seller_id
  ) THEN
     SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Seller cannot sell
in two places at the same time within the last hour.';
  END IF;
END;
```

<u>פיתוח אפליקציה יישומית:</u>

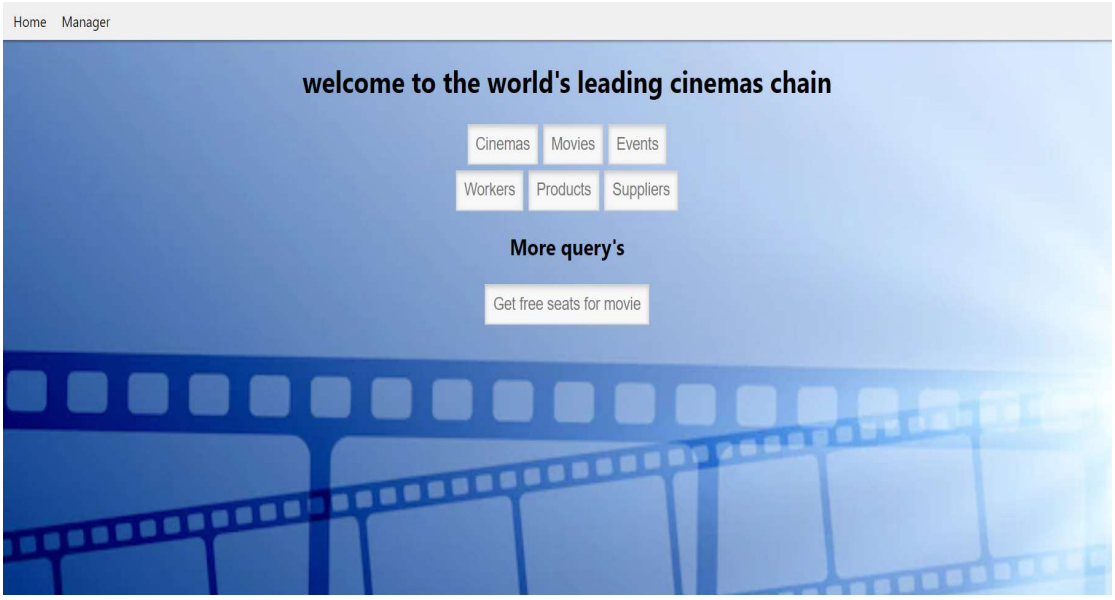
על מנת שנוכל להציג את הנתונים בעזרת node.js בצד שרת ו-React בצד לקוח אז העברנו את בסיס הנתונים מ-phpMyAdmin לשרת mysql מקומי ע"י שימוש באופציית Export ב-phpMyAdmin.

צורת הבניה זה ע"י שימוש בספריות express ו-mysql2 בצד שרת על מנת לנהל את הנתיבים והגישה לבסיס הנתונים בעוד בצד לקוח השתמשנו ב-React.

חילקנו את צד הלקוח לשני חלקים כאשר חלק אחד זה למשתמשים ששם ניתן לקבל את הטבלאות שמכילות נתונים על בתי הקולנוע, חדרי הקולנוע, מוצרים הנמכרים במזנון ומושבים פנויים לפי סניף, תאריך ושם הסרט. החלק השני והיותר מרכזי זה למנהל הקולנוע ששם הוא יכול לקבל מידע על כמות המכירות במזנון לפי סרט, כמות מכירות לפי מוצר לכל סניף, עובד מצטיין בכל סניף ומידע נוסף שחשוב להנהלה. בנוסף יש למנהל אופציה להוסיף עובדים חדשים, לשנות משכורות לעובדים קיימים, ולהוסיף הזמנות שמתבצעות במזנון.

את הנתונים אנחנו לוקחים מהבסיס נתונים על ידי שאילתות sql וכן על ידי שימוש בפרוצדורות שייצרנו במהלך הפרויקט.

המסך הראשי ללקוחות:



בעת התחברות מופיע למשתמש בעצם לחצנים לבחירת הנתונים שהוא מעוניין לקבל. לאחר לחיצה על נתונים שהוא מעוניין לקבל הטבלה תופיע באופן הבא:

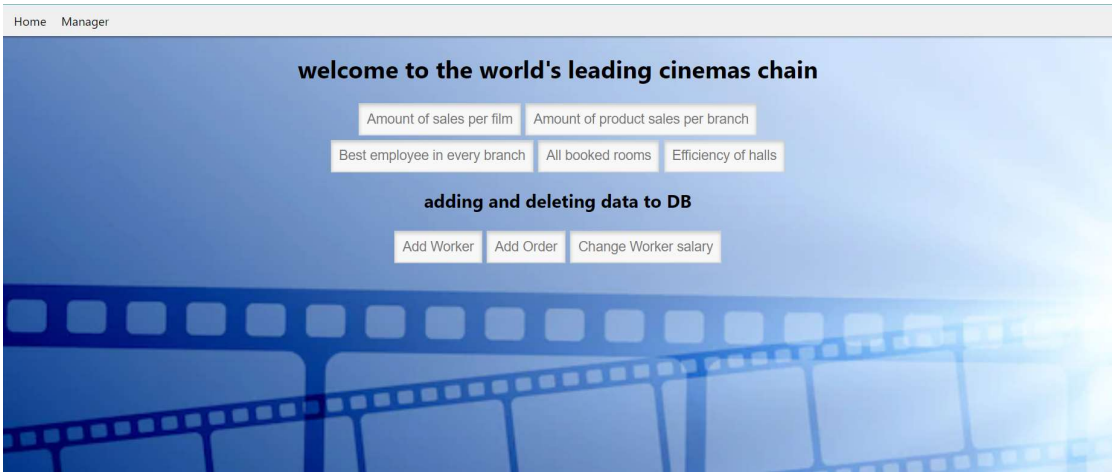
Response Table			
movie_name	start_time	end_time	date
Movie 477	18:24:02	20:24:02	2005-05-13T21:00:00.000Z
Movie 603	16:00:41	18:00:41	1968-04-11T22:00:00.000Z
Movie 480	10:35:28	12:35:28	2014-01-03T22:00:00.000Z
Movie 741	18:49:31	20:49:31	2023-03-12T22:00:00.000Z
Movie 685	20:38:30	22:38:30	1976-02-12T22:00:00.000Z
Movie 11	19:18:12	21:18:12	2018-02-10T22:00:00.000Z
Movie 265	20:38:30	22:38:30	1976-02-12T22:00:00.000Z
Movie 25	18:49:31	20:49:31	2023-03-12T22:00:00.000Z
Movie 60	18:24:02	20:24:02	2005-05-13T21:00:00.000Z
Movie 337	20:38:30	22:38:30	1976-02-12T22:00:00.000Z
Movie 458	16:28:20	18:28:20	1992-04-02T21:00:00.000Z

כאשר הלחצנים מעל עדיין נשארים לבחירת טבלה אחרת.

במידה והמשתמש מעוניין במידע על מושבים פנויים אז יפתח לו שדות להכנסת הנתונים באופן הבא:

המסך הראשי למנהלים:

צורת ההצגה זהה להצגה ללקוחות. המסך הראשי נראה כך:



כאשר גם כאן ניתן לבחור טבלאות מסוימות ולקבל את הנתונים. במידה ורוצים לבצע שינויים אז לחיצה על האפשרות הזאת תפתח שדות למילוי נתונים לצורך ביצוע הפעולה.

למשל עבור הוספת הזמנה זה יראה כך:

במידה ויש שגיאה בהכנסה כמו למשל הכנסת עובד שקיים כבר אז האתר יקפיץ למשתמש שהפעולה לא התבצעה ואת הסיבה לכך.