**תיעוד ניהול מסעדה**

**מגישים:  
מאור לוי-328301981**

**ארי עוז- 321918484**

‏2025

**קישור לגיטאהב:**  
<https://github.com/MAORLEVI14/DBProject1981_8484>

דוח סיכום פרויקט

# **תוכן עניינים**

תיאור כללי של המערכת 2

תיאור הישויות במערכת 3

הסבר על נרמול (3NF) 4

דיאגרמות 5

יצירת הטבלאות- SQL 7

פעולות SQL נוספות 9

שיטות הכנסת נתונים למסד 11

גיבוי ושחזור מסד הנתונים 13

דוגמאות למגבלות (Constraints) 17

דוגמאות לשאילתות SELECT מתקדמות 18

דוגמאות לעדכון (UPDATE) ומחיקה (DELETE) 20

ניהול טרנזקציות – Commit & Rollback 21

# **תיאור כללי של המערכת**

מערכת זו פותחה לניהול כולל של מסעדה. באמצעותה ניתן לנהל לקוחות, עובדים, מנות, הזמנות, שולחנות, תשלומים ועוד.  
המערכת מתבססת על **בסיס נתונים רלציוני,**  הכולל **טבלאות מקושרות ומנורמלות עד לרמה של Third Normal Form (3NF)** – צורת נרמול שלישית – לצורך שמירה על עקביות, הימנעות מכפילויות והבטחת שלמות המידע.

המערכת תוכננה כך שתתמוך בכל היבטי הפעילות המרכזיים של מסעדה מודרנית, ותאפשר:

* מעקב אחרי לקוחות והיסטוריית ההזמנות שלהם
* הקצאת שולחנות וניהול צוות המלצרים
* ניהול תפריט ומנות כולל מחירים וזמני הכנה
* ניהול הזמנות מורכבות עם מספר מנות
* תיעוד תשלומים ונתוני חיוב לפי אמצעי תשלום שונים

# **תיאור הישויות במערכת**

**Customer**

מייצגת את לקוחות המסעדה. כל לקוח מזוהה לפי מזהה ייחודי, וכולל שם מלא וסטטוס הזמנה (למשל: הוזמן, ממתין לאישור, סיום תשלום).

**Waiter**

מייצגת את המלצרים העובדים במסעדה. כוללת מזהה, שם מלא ושנות ניסיון.

**Dish**

טבלה זו מתארת את המנות המוגשות במסעדה, עם מזהה מנה, שם, תיאור קצר, קטגוריה (מנה ראשונה/עיקרית/קינוח), מחיר וזמן הכנה.

**RestTable**

טבלה זו כוללת את כל השולחנות במסעדה, עם מזהה, כמות מושבים וסטטוס (פנוי/תפוס).

**Payment**

מייצגת את פרטי התשלום – האם בוצע, שיטת תשלום ותאריך ביצוע.

**RestOrder**

שומרת את ההזמנות – כולל תאריך, שעה, סטטוס הזמנה, מזהי לקוח, מלצר, שולחן ותשלום.

**part\_of\_order**

טבלה מקשרת בין מנות להזמנות. כל הזמנה יכולה לכלול מספר מנות.

# **הסבר על נרמול (3NF)**

**📄 Customer**

כל שדה תלוי ישירות רק במפתח הראשי (customer\_id) ואין תלות בין שדות עצמם.

**📄 Waiter**

כל שדה כמו שם, תאריך התחלה ושכר מתאר את המלצר באופן עצמאי ותלוי רק ב־waiter\_id ללא תלות בין השדות.

**📄 Dish**

שם המנה, מחיר וזמן ההכנה תלויים רק ב־dish\_id אין קשר ישיר בין המחיר לזמן ההכנה, ולכן אין תלות מעבר.

**📄 RestTable**

שולחן מזוהה לפי table\_id ומכיל רק את מספר המושבים – שדה יחיד שתלוי ישירות רק במפתח.

**📄 RestOrder**

הזמנה מזוהה לפי order\_id כל שדה אחר כמו מזהה לקוח, מלצר, תאריך או שולחן תלוי רק במזהה ההזמנה, ולא אחד בשני.

**📄 part\_of\_order**

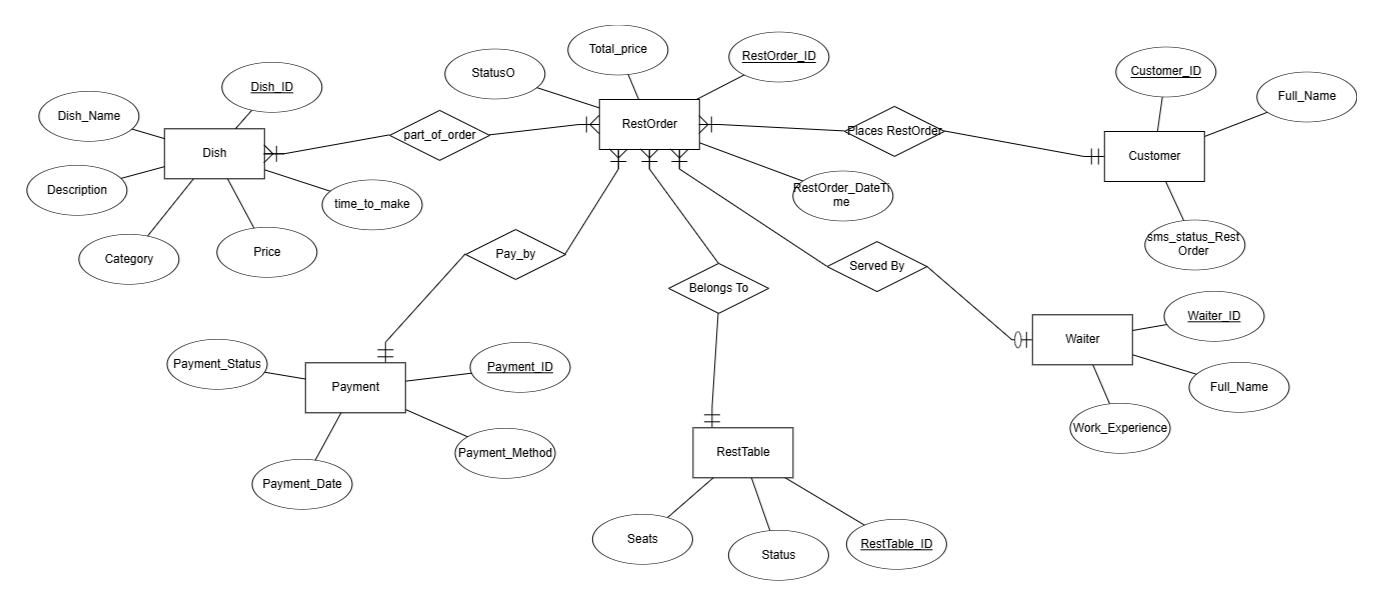
מזהה של מנה בתוך הזמנה נקבע על ידי צמד המפתחות (order\_id, dish\_id) והכמות תלויה אך ורק בשילוב הזה, ללא תלות בין העמודות.

**📄 Payment**

הטבלה מתארת תשלום מזוהה לפי payment\_idכאשר הסכום, אמצעי התשלום והתאריך תלויים אך ורק בו, ולא אחד בשני.

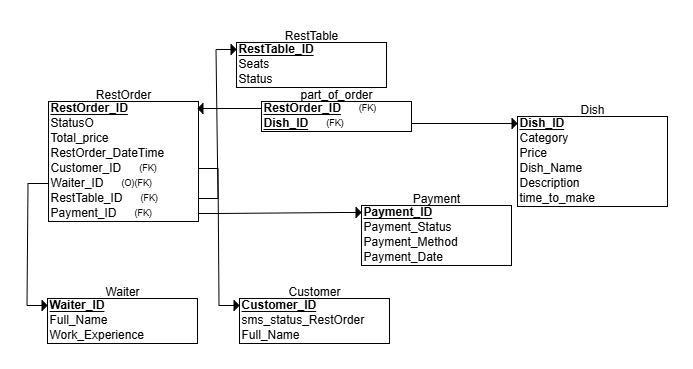
**מבנה זה מבטיח ייעול מקום, שלמות לוגית והימנעות מכפילויות**.

# **דיאגרמות**

דיאגרמת ERD

דיאגרמת DSD

דיאגרמת DSD



דיאגרמת DSD

# **SQL – יצירת הטבלאות**

CREATE TABLE Dish (  
 Category VARCHAR(15) NOT NULL,  
 Price FLOAT NOT NULL,  
 Dish\_ID INT NOT NULL,  
 Dish\_Name VARCHAR(15) NOT NULL,  
 Description VARCHAR(50) NOT NULL,  
 time\_to\_make VARCHAR(15) NOT NULL,  
 PRIMARY KEY (Dish\_ID)  
);

CREATE TABLE Payment (  
 Payment\_ID INT NOT NULL,  
 Payment\_Status VARCHAR(10) NOT NULL,  
 Payment\_Method VARCHAR(10) NOT NULL,  
 Payment\_Date DATE NOT NULL,  
 PRIMARY KEY (Payment\_ID)  
);

CREATE TABLE Waiter (  
 Waiter\_ID INT NOT NULL,  
 Full\_Name VARCHAR(50) NOT NULL,  
 Work\_Experience INT NOT NULL,  
 PRIMARY KEY (Waiter\_ID)  
);

CREATE TABLE Customer (  
 sms\_status\_order VARCHAR(50) NOT NULL,  
 Customer\_ID INT NOT NULL,  
 Full\_Name VARCHAR(50) NOT NULL,  
 PRIMARY KEY (Customer\_ID)  
);

CREATE TABLE RestTable (  
 Seats INT NOT NULL,  
 Table\_ID INT NOT NULL,  
 Status0 VARCHAR(10) NOT NULL,  
 PRIMARY KEY (Table\_ID)  
);

CREATE TABLE RestOrder (  
 Status0 VARCHAR(10) NOT NULL,  
 Total\_price FLOAT NOT NULL,  
 Order\_ID INT NOT NULL,  
 Order\_DateTime DATE NOT NULL,  
 Customer\_ID INT NOT NULL,  
 Waiter\_ID INT,  
 Table\_ID INT NOT NULL,  
 Payment\_ID INT NOT NULL,  
 PRIMARY KEY (Order\_ID),  
 FOREIGN KEY (Customer\_ID) REFERENCES Customer(Customer\_ID),  
 FOREIGN KEY (Waiter\_ID) REFERENCES Waiter(Waiter\_ID),  
 FOREIGN KEY (Table\_ID) REFERENCES RestTable(Table\_ID),  
 FOREIGN KEY (Payment\_ID) REFERENCES Payment(Payment\_ID)  
);

CREATE TABLE part\_of\_order (  
 Order\_ID INT NOT NULL,  
 Dish\_ID INT NOT NULL,  
 PRIMARY KEY (Order\_ID, Dish\_ID),  
 FOREIGN KEY (Order\_ID) REFERENCES RestOrder(Order\_ID),  
 FOREIGN KEY (Dish\_ID) REFERENCES Dish(Dish\_ID)  
);

# **פעולות SQL נוספות**

הכנסת מנה חדשה לטבלת המנות:

INSERT INTO Dish (Category, Price, Dish\_ID, Dish\_Name, Description, time\_to\_make)

VALUES

('Main Course', 49.99, 1, 'Steak', 'Grilled steak with sauce', '25 min'),

('Dessert', 19.99, 2, 'Chocolate Cake', 'Rich chocolate cake', '15 min'),

('Appetizer', 29.99, 3, 'Bruschetta', 'Tomato and basil on toast', '10 min');

יצירת טבלה חדשה:

CREATE TABLE Dish

(

Category VARCHAR(15) NOT NULL,

Price FLOAT NOT NULL,

Dish\_ID INT NOT NULL,

Dish\_Name VARCHAR(15) NOT NULL,

Description VARCHAR(50) NOT NULL,

time\_to\_make VARCHAR(15) NOT NULL,

PRIMARY KEY (Dish\_ID)

);

בחירה של טבלה והצגת הנתונים:

-- הצגת כל הנתונים בטבלת Dish

SELECT \* FROM Dish;

-- הצגת כל הנתונים בטבלת Payment

SELECT \* FROM Payment;

-- הצגת כל הנתונים בטבלת Waiter

SELECT \* FROM Waiter;

-- הצגת כל הנתונים בטבלת Customer

SELECT \* FROM Customer;

-- הצגת כל הנתונים בטבלת RestTable

SELECT \* FROM RestTable;

-- הצגת כל הנתונים בטבלת RestOrder

SELECT \* FROM RestOrder;

-- הצגת כל הנתונים בטבלת part\_of\_order

SELECT \* FROM part\_of\_order;

מחיקת טבלה:

DROP TABLE part\_of\_order;

DROP TABLE RestOrder;

DROP TABLE Payment;

DROP TABLE Customer;

DROP TABLE Waiter;

DROP TABLE RestTable;

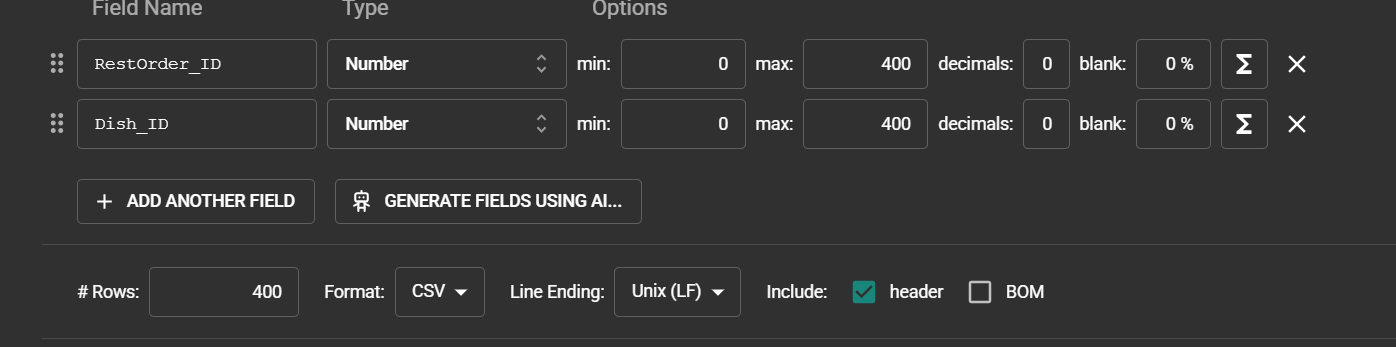
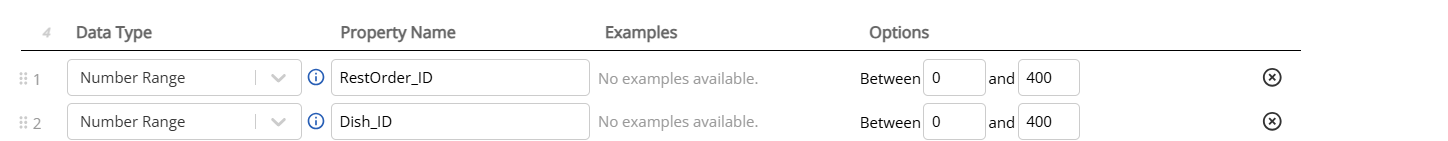
DROP TABLE Dish;

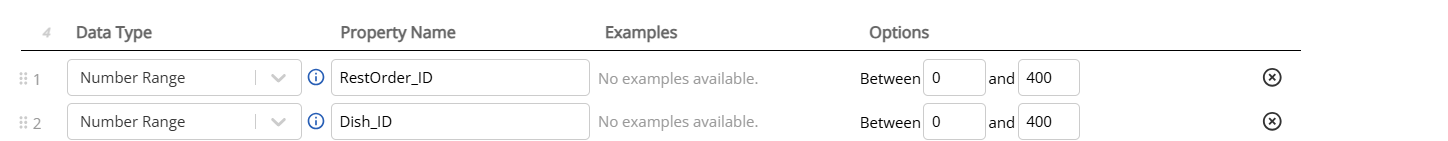
# **שיטות הכנסת נתונים למסד**

במהלך העבודה על המערכת, השתמשנו בשלוש שיטות עיקריות להזנת נתונים לטבלאות במסד הנתונים.  
שיטות אלו אפשרו ניפוי שגיאות, בדיקת נתונים, וטעינת מידע שונה בכמות גדולה.

## ייבוא מקובץ CSV – Mockaroo / GenerateData

השתמשנו באתרים כמו Mockaroo ו GenerateData ליצירת נתונים סינתטיים איכותיים בפורמט CSV





## סקריפט ב־Python

1. import csv  
   import os  
     
   csv\_folder = "csv\_files"  
   output\_folder = "sql\_output"  
   os.makedirs(output\_folder, exist\_ok=True)  
     
     
   tables = [  
    "Dish",  
    "Payment",  
    "Waiter",  
    "Customer",  
    "RestTable",  
    "RestOrder",  
    "part\_of\_order"  
   ]  
     
   # פונקציה להמרת ערך לתוך SQL (מספרים, תאריכים, טקסט  
   def format\_value(val):  
    val = val.strip()  
    if val == "":  
    return "NULL"  
    try:  
    float(val) # לבדוק אם זה מספר  
    return val  
    except ValueError:  
    if '-' in val and len(val.split('-')) == 3: # תאריך (פורמט: YYYY-MM-DD)  
    return f"'{val}'"  
    safe\_val = val.replace("'", "''") # לברוח גרשיים  
    return f"'{safe\_val}'"  
     
   # עיבוד כל קובץ  
   for table in tables:  
    csv\_path = os.path.join(csv\_folder, f"{table}.csv")  
    sql\_path = os.path.join(output\_folder, f"{table}.sql")  
     
    with open(csv\_path, mode='r', encoding='utf-8') as csv\_file:  
    reader = csv.reader(csv\_file)  
    headers = next(reader) # שורת כותרת  
     
    with open(sql\_path, mode='w', encoding='utf-8') as sql\_file:  
    for row in reader:  
    values = [format\_value(v) for v in row]  
    insert\_stmt = f"INSERT INTO {table} ({', '.join(headers)}) VALUES ({', '.join(values)});"  
    sql\_file.write(insert\_stmt + "\n")

## **3. הזנה ידנית**

## הוזנו לעיתים ערכים באופן ידני בעזרת הממשק כדי שיהיה במאגר מידע שנוכל לבצע שאילתות SQL

# **גיבוי ושחזור מסד הנתונים**

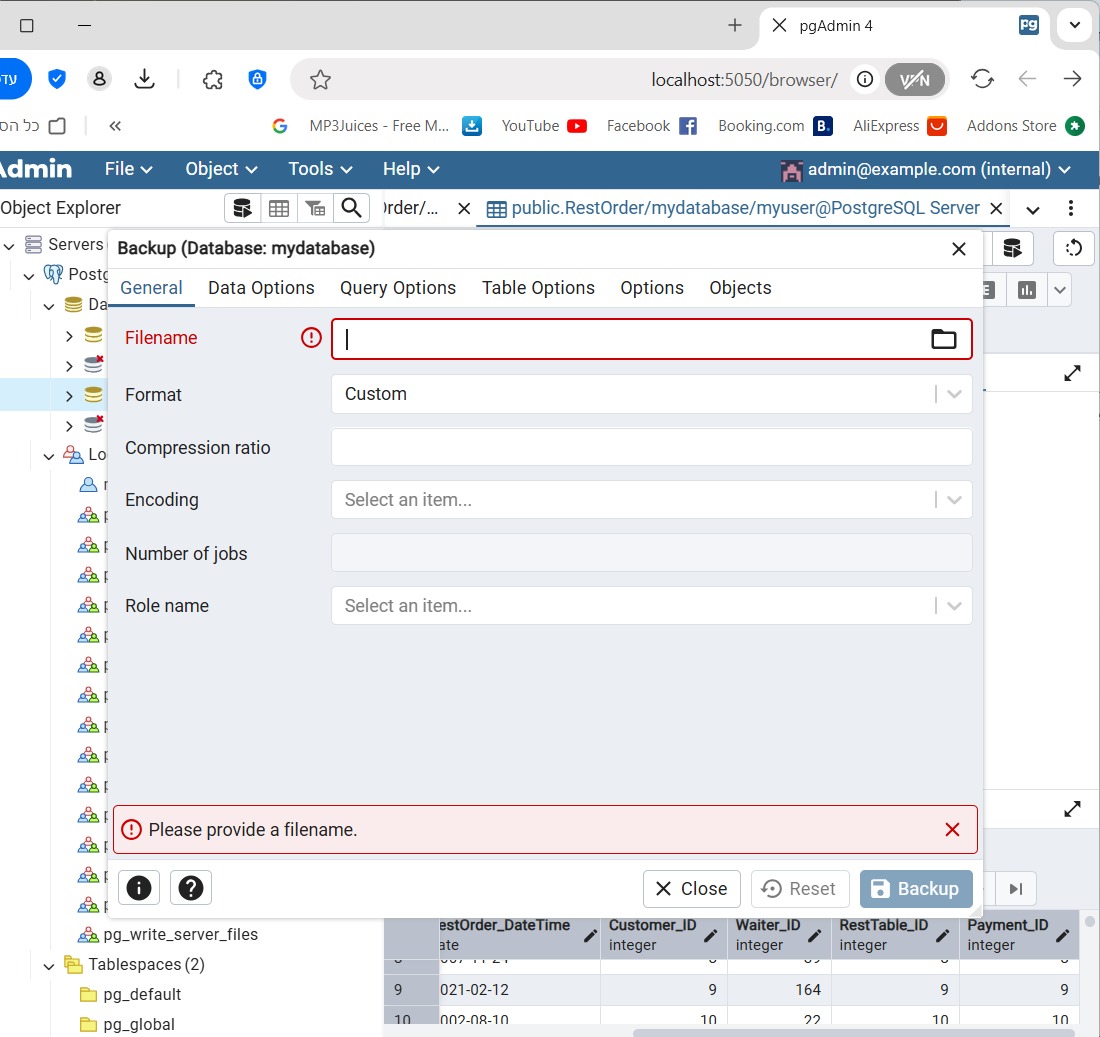
במסגרת הפרוייקט בוצע גיבוי של בסיס הנתונים באמצעות pgAdmin וכלל את כל הטבלאות, הנתונים, והקשרים.

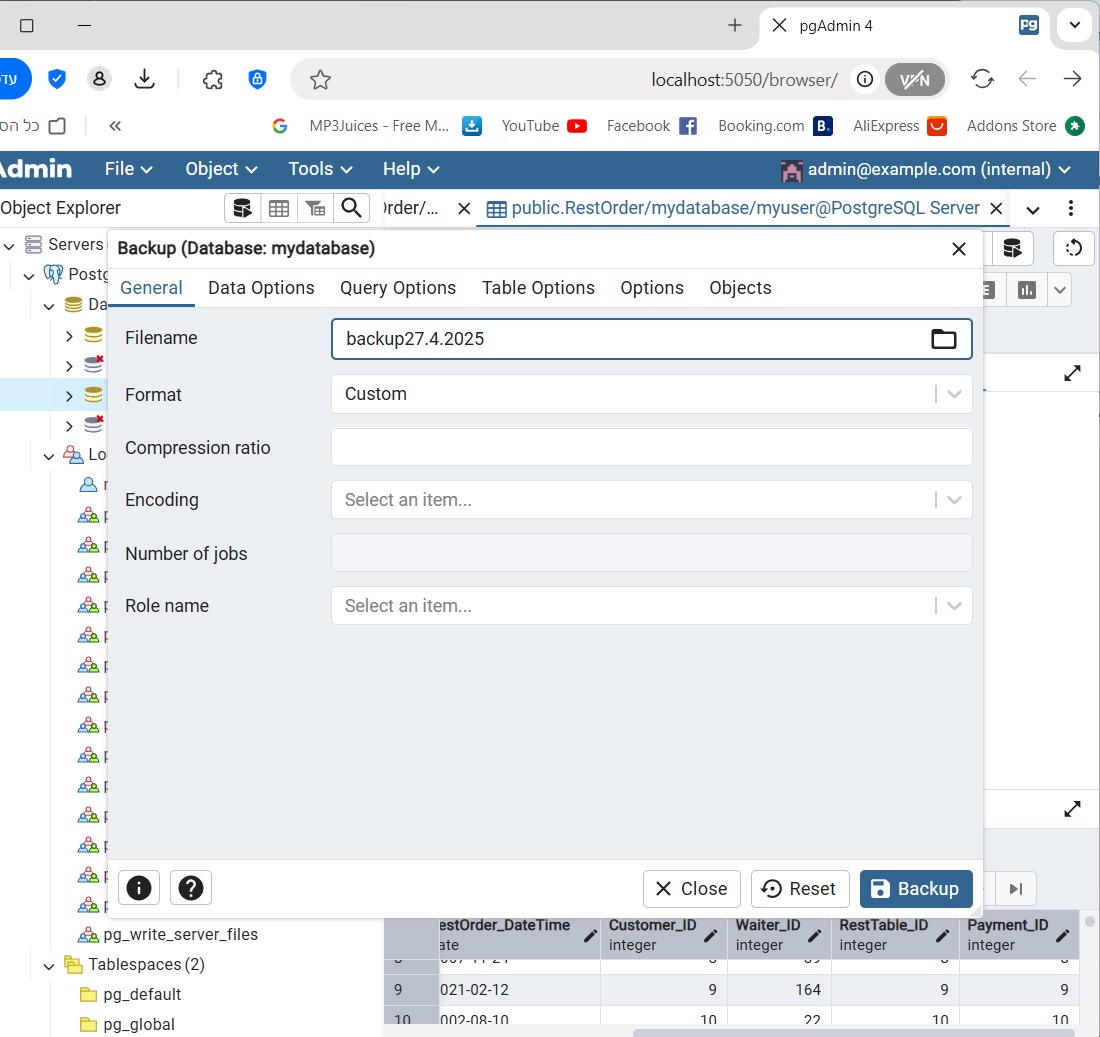
הגיבוי נשמר כקובץ backup ובמידת הצורך ניתן יהיה לעשות שחזור מלא של הנתונים.

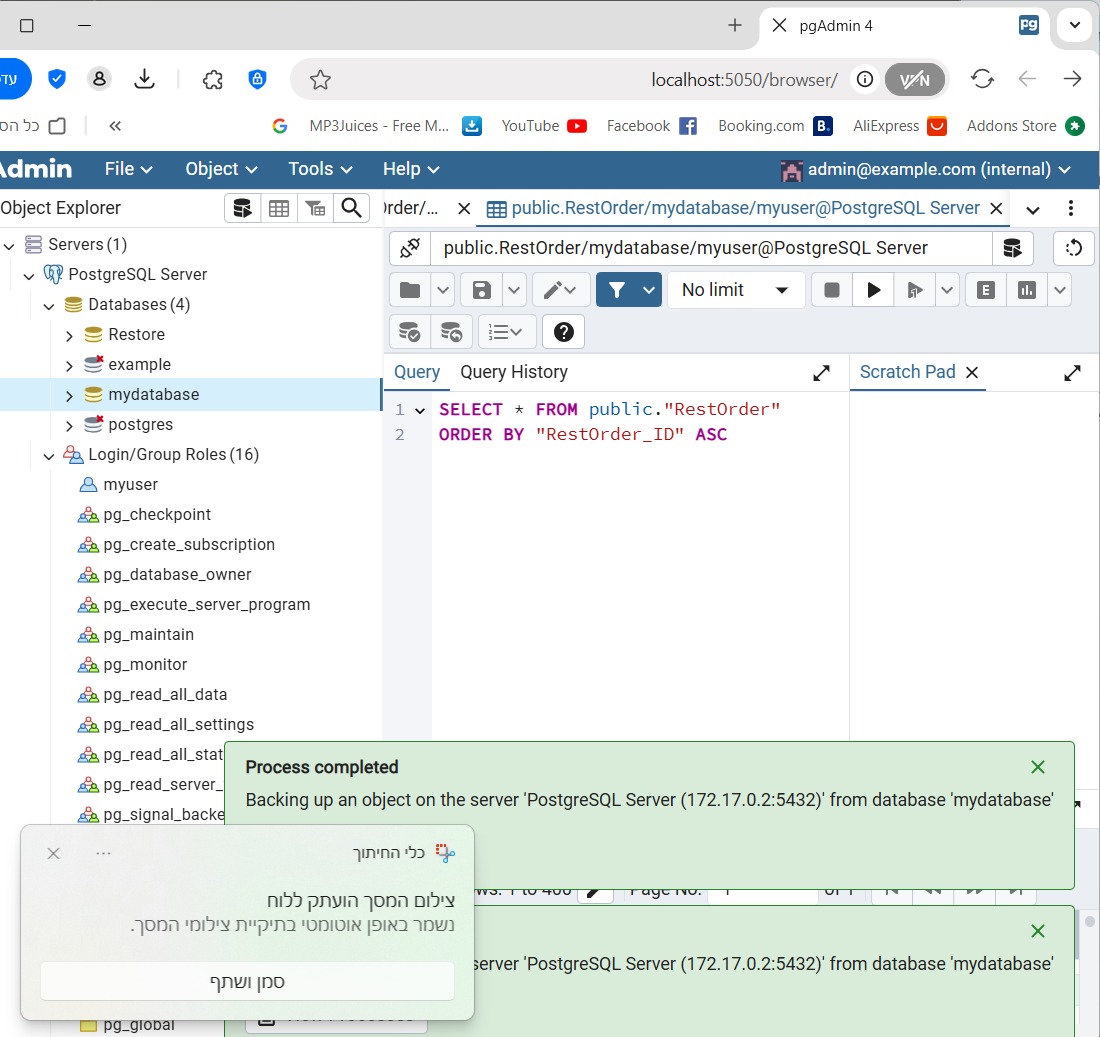
כמו כן ניסינו גם לשחזר על מנת לראות שהגיבוי אכן עובד, השחזור בוצע על ידי טעינת קובץ הגיבוי למסד נתונים חדש או קיים וביצוע פעולת Restore דרך הממשק הגרפי.

במהלך תהליך השחזור בדקנו שלא נפגע תקינות הקשרים ושלמותם.

## 📸 שלבי גיבוי:

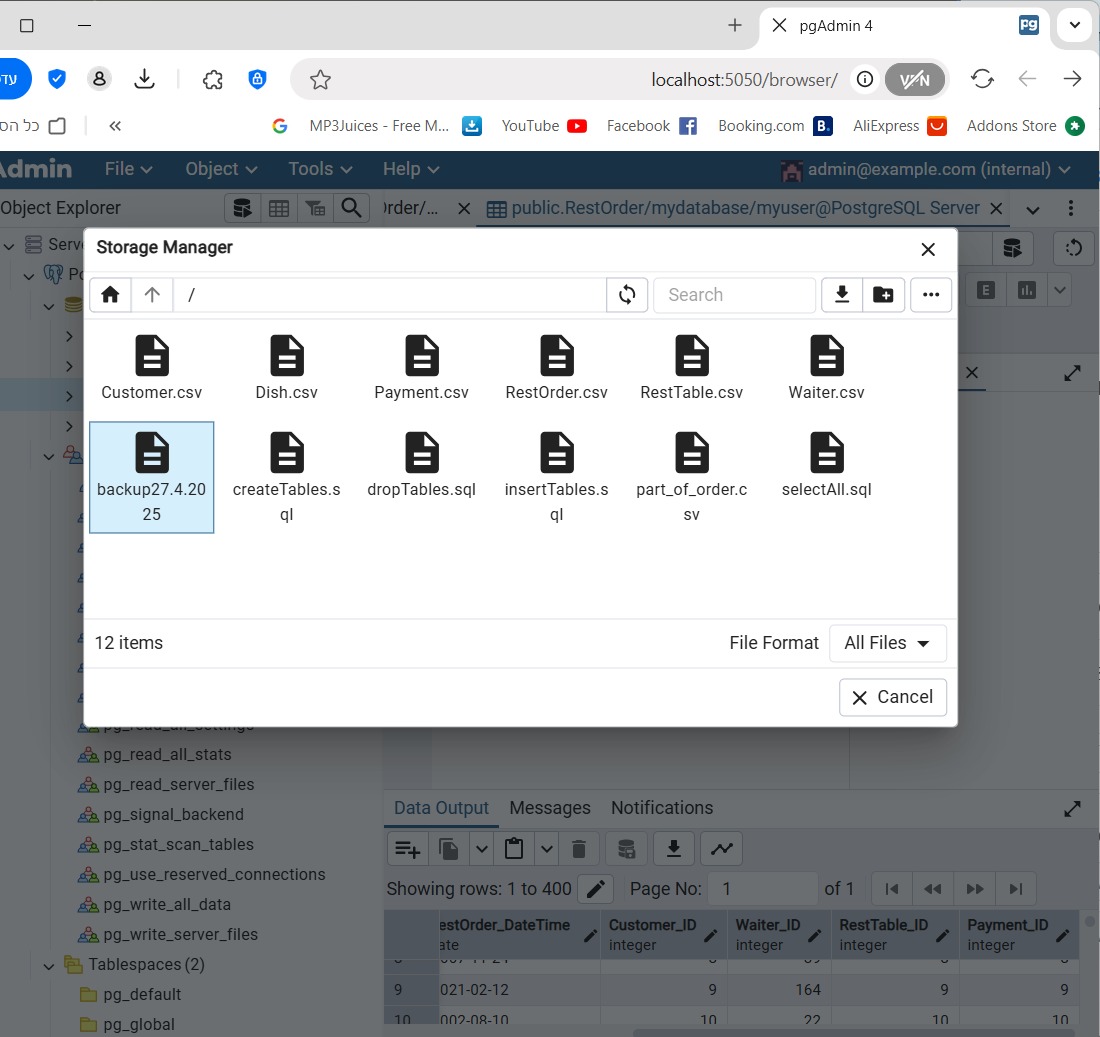
פתיחת תפריט הגיבוי

חירת שם הקובץב

תהליך הגיבוי הושלם בהצלחה

## 📸 שלבי שחזור:

פתיחת אפשרות Restore

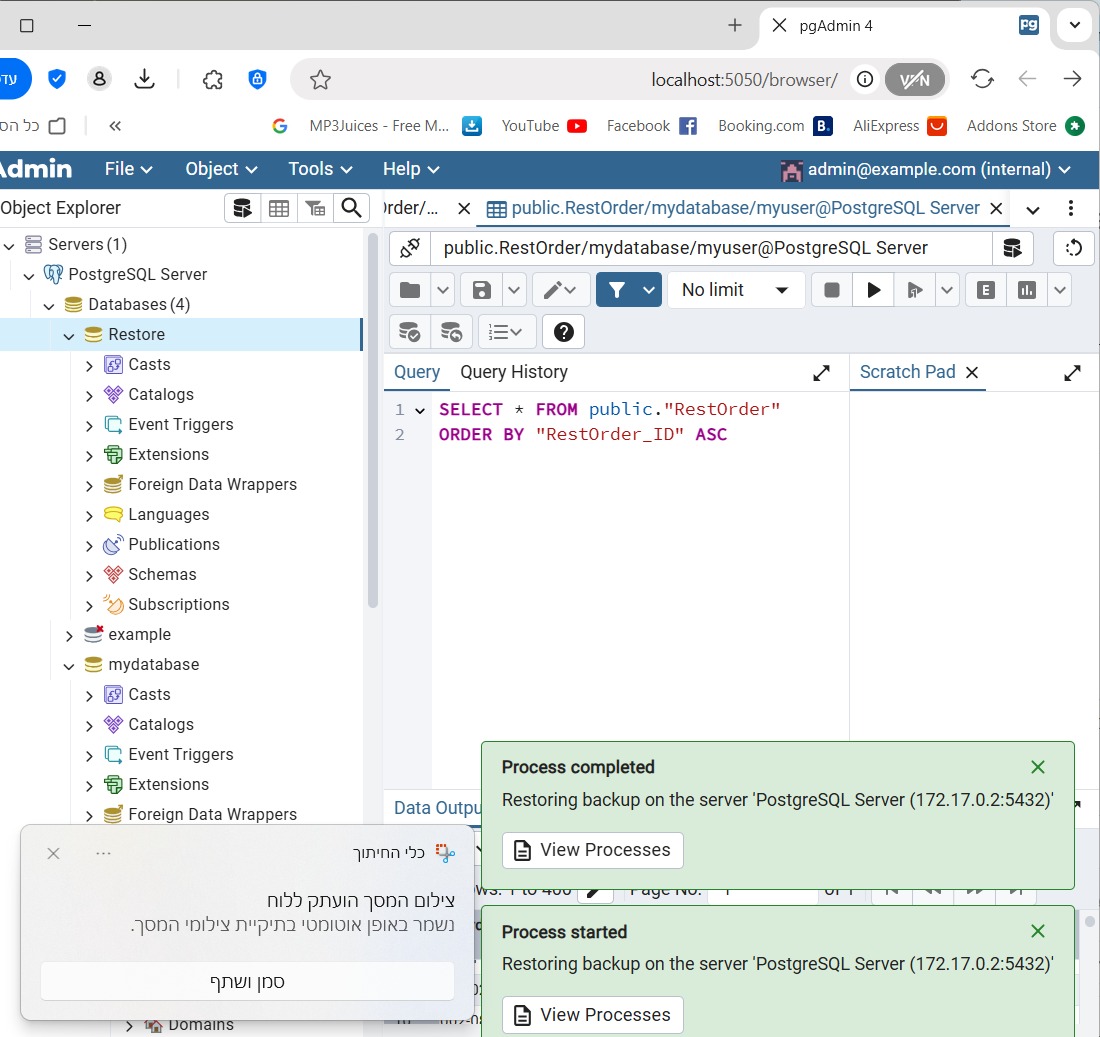


**בחירת הקובץ**

תמונה שמכילה טקסט, צילום מסך, תוכנה, סמל מחשב

תוכן שנוצר על-ידי בינה מלאכותית עשוי להיות שגוי.

**השחזור הושלם בהצלחה**



# **דוגמאות למגבלות (Constraints)**

בשלב ב' הוספנו מגבלות (Constraints) לטבלאות במסד הנתונים כדי להבטיח תקינות לוגית של המידע. כל אילוץ נבחר בקפידה כדי למנוע טעויות נפוצות:

* **הגבלת זמן הכנה של מנה לשעה מקסימום:**
  + מונע הכנסת ערכים לא הגיוניים לשדה זמן הכנה.
  + דוגמה: ניסיון להכניס מנה עם זמן הכנה של שעתיים יגרום לשגיאה.
* **הגבלת אורך שם מנה ל-15 תווים:**
  + מבטיח שמות קצרים וקריאים בתפריט.
* **הגבלת תיאור מנה ל-250 תווים:**
  + מונע תיאורים ארוכים מדי.
* **ערך ברירת מחדל לאמצעי תשלום (Cash):**
  + אם לא הוזן אמצעי תשלום, ייקבע אוטומטית 'Cash'.
* **תאריך תשלום לא עתידי:**
  + מונע טעויות של תאריכים שגויים.
* **ניסיון עבודה של מלצר לא שלילי:**
  + מונע הכנסת ערכים שליליים לשנות ניסיון.
* **שם מלצר ייחודי:**
  + מונע כפילויות של שמות עובדים.

**המחשה:**

ALTER TABLE Dish ADD CONSTRAINT check\_time\_to\_make CHECK (time\_to\_make <= TIME '01:00:00');

-- ניסיון להכניס מנה עם זמן הכנה 02:00:00 יגרום לשגיאה

# **דוגמאות לשאילתות SELECT מתקדמות**

בשלב ב' נכתבו שאילתות מתקדמות שמאפשרות ניתוחים חכמים על בסיס הנתונים:

* **שליפת מנות שלוקח להן מעל 20 דקות להכין, כולל שם הלקוח והמלצר:**
  + מאפשרת למנהל לזהות מנות איטיות ולשפר תהליכים.

SELECT Dish.Dish\_Name, Dish.time\_to\_make, Customer.Full\_Name AS Customer, Waiter.Full\_Name AS Waiter

FROM RestOrder

JOIN part\_of\_order ON RestOrder.RestOrder\_ID = part\_of\_order.RestOrder\_ID

JOIN Dish ON Dish.Dish\_ID = part\_of\_order.Dish\_ID

JOIN Customer ON Customer.Customer\_ID = RestOrder.Customer\_ID

LEFT JOIN Waiter ON Waiter.Waiter\_ID = RestOrder.Waiter\_ID

WHERE Dish.time\_to\_make > '00:20:00';

* **חישוב הכנסה יומית:**
  + מאפשר ניתוח רווחיות לפי ימים.

SELECT TO\_CHAR(RestOrder.RestOrder\_DateTime,'FMDay') AS Day, SUM(RestOrder.Total\_price) AS DailyRevenue

FROM RestOrder

GROUP BY Day

ORDER BY DailyRevenue DESC;

* **לקוחות שהזמינו יותר ממנה אחת:**
  + מסייע בזיהוי לקוחות קבועים/גדולים.

SELECT Customer.Full\_Name, COUNT(part\_of\_order.Dish\_ID) AS DishesOrdered

FROM Customer

JOIN RestOrder ON Customer.Customer\_ID = RestOrder.Customer\_ID

JOIN part\_of\_order ON RestOrder.RestOrder\_ID = part\_of\_order.RestOrder\_ID

GROUP BY Customer.Full\_Name

HAVING COUNT(part\_of\_order.Dish\_ID) > 1;

# **דוגמאות לעדכון (UPDATE) ומחיקה (DELETE)**

* **עדכון סטטוס שולחן לפנוי כאשר התשלום הושלם:**
  + מבטיח ששולחנות יתפנו אוטומטית לאחר תשלום.

UPDATE RestTable

SET Status = 'Available'

WHERE RestTable\_ID IN (

SELECT RestOrder.RestTable\_ID

FROM RestOrder

JOIN Payment ON RestOrder.Payment\_ID = Payment.Payment\_ID

WHERE Payment.Payment\_Status = 'Completed'

);

* **עדכון זמן הכנה רנדומלי לכל מנה:**
  + מאפשר בדיקות עומס וסטטיסטיקות שונות.

UPDATE Dish

SET time\_to\_make = DATE\_TRUNC('minute', '00:00'::time + (RANDOM() \* INTERVAL '60 minutes'))::time;

* **מחיקת מנות שלא הוזמנו מעולם:**
  + מנקה נתונים לא רלוונטיים.

DELETE FROM Dish

WHERE Dish\_ID NOT IN (

SELECT DISTINCT Dish\_ID FROM part\_of\_order

);

* **מחיקת לקוחות שלא ביצעו הזמנה:**
  + שומר על מסד נתונים נקי.

DELETE FROM Customer

WHERE Customer\_ID NOT IN (

SELECT Customer\_ID FROM RestOrder);

# **ניהול טרנזקציות – Commit & Rollback**

בשלב ב' תרגלנו שמירה (Commit) וביטול (Rollback) של שינויים במסד הנתונים. כך ניתן להבטיח שכל שינוי קריטי יתבצע בשלמותו או יתבטל לחלוטין במקרה של תקלה.

* **Commit:** שומר את כל השינויים שבוצעו בטרנזקציה.
* **Rollback:** מבטל את כל השינויים שבוצעו מאז ה-Commit האחרון.

להלן המחשה של תהליך הטרנזקציה באמצעות צילומי מסך:



