

UNIVERSITETI I TETOVES
FAKULTETI I SHEKNCAVE MATEMATIKO-NATYRORE
DEPARTAMENTI I INFORMATIKE



LENDË: DBMS

TEMA: Restaurant Database

Mentori:

Prof.Ass.Grela Ajvazi

Punoi:

Muiz Rexhepi

Tetove 2023

Abstrakti

Baza e të dhënave për restorant është e projektuar për të ruajtur dhe menaxhuar informacionin lidhur me produktet, kategoritë, klientët, punonjësit, porositë dhe elementet e porosisë të një restoranti.

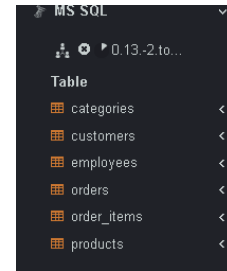
Baza e të dhënave mundëson organizimin dhe kërkimin efikas të të dhënave për të mbështetur operacionet e ndryshme të restoranit.

Baza e të dhënave për restorant lejon menaxhimin efikas të informacionit të produkteve, detajet e klientëve, regjistrat e punonjësve dhe gjurmimin e porosive. Duke organizuar dhe lidhur këto të dhëna, baza e të dhënave mundëson operacione të ndryshme të restorantit si krijimi i raporteve, analiza të shitjeve dhe ofrimi i një përvoje të shkëlqyer për klientët.

Fjale kyce: Kategoritë, produktet, klientet, punojesit, porosite, elementet e porosise.

```
CREATE DATABASE restaurant;
```

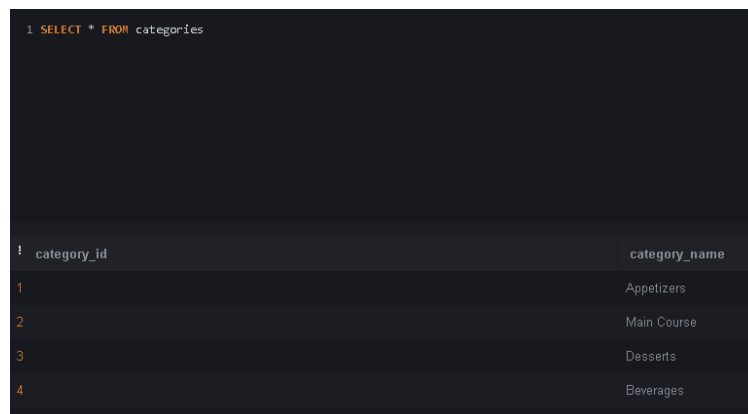
```
USE restaurant;
```



Te gjithë tabelat

Tabela për ruajtjen e kategorive të produkteve

```
CREATE TABLE categories (  
    category_id INT PRIMARY KEY, -- Identifikues unik për çdo kategori  
    category_name VARCHAR(255) -- Emri i kategorisë  
);
```



```
1 SELECT * FROM categories
```

category_id	category_name
1	Appetizers
2	Main Course
3	Desserts
4	Beverages

Tabela për ruajtjen e produkteve

```
CREATE TABLE products (  
    product_id INT PRIMARY KEY, -- Identifikues unik për çdo produkt  
    product_name VARCHAR(255), -- Emri i produktit  
    price DECIMAL(10, 2), -- Çmimi i produktit  
    category_id INT, -- Çelës i huazuar që referencë kategorive  
    FOREIGN KEY (category_id) REFERENCES categories(category_id) -- Mardhënie me tabelën e kategorive  
);
```



```
1 SELECT * FROM products
```

product_id	product_name	price	category_id
1	Chicken Wings	9.99	1
2	Caesar Salad	7.99	1
3	Margherita Pizza	12.99	2
4	Grilled Salmon	18.99	2
5	Chocolate Brownie	6.99	3
6	Cheesecake	8.99	3
7	Coca-Cola	2.49	4
8	Orange Juice	2.99	4

Tabela për ruajtjen e informacionit të klientëve

```
CREATE TABLE customers (  
  customer_id INT PRIMARY KEY, -- Identifikues unik për çdo klient  
  customer_name VARCHAR(255), -- Emri i klientit  
  phone_number VARCHAR(20), -- Numri i telefonit të klientit  
  email VARCHAR(255) -- Adresa email e klientit  
);
```

```
1 SELECT * FROM customers
```

! customer_id	customer_name	phone_number	email
1	John Doe	123-456-7890	john.doe@example.com
2	Jane Smith	987-654-3210	jane.smith@example.com

Tabela për ruajtjen e informacionit të punonjësve

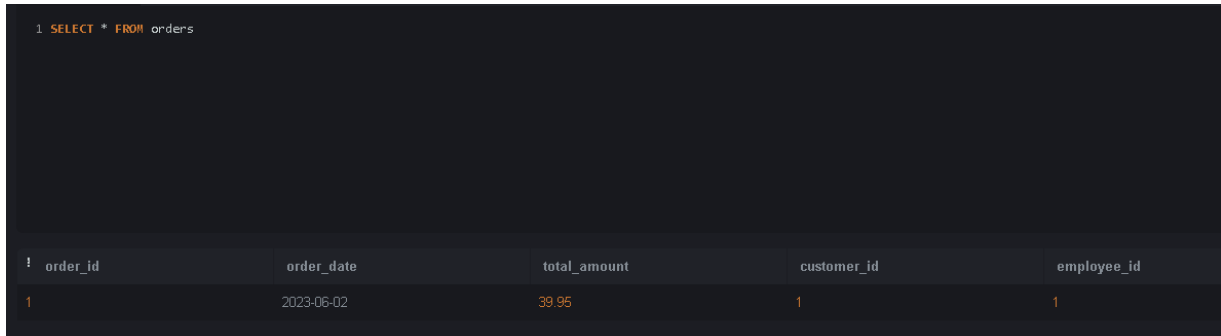
```
CREATE TABLE employees (  
  employee_id INT PRIMARY KEY, -- Identifikues unik për çdo punonjës  
  employee_name VARCHAR(255), -- Emri i punonjësit  
  position VARCHAR(255) -- Pozicioni ose titulli i punonjësit  
);
```

```
1 SELECT * FROM employees
```

! employee_id	employee_name	position
1	Michael Johnson	Waiter
2	Emily Davis	Chef

Tabela për ruajtjen e porosive

```
CREATE TABLE orders (  
  order_id INT PRIMARY KEY,      -- Identifikues unik për çdo porosi  
  order_date DATE,              -- Data e porosisë  
  total_amount DECIMAL(10, 2),  -- Shuma totale e porosisë  
  customer_id INT,              -- Çelës i huazuar që referencë klientëve  
  employee_id INT,              -- Çelës i huazuar që referencë punonjësve  
  FOREIGN KEY (customer_id) REFERENCES customers(customer_id), -- Mardhënie me tabelën e klientëve  
  FOREIGN KEY (employee_id) REFERENCES employees(employee_id)  -- Mardhënie me tabelën e punonjës  
);
```

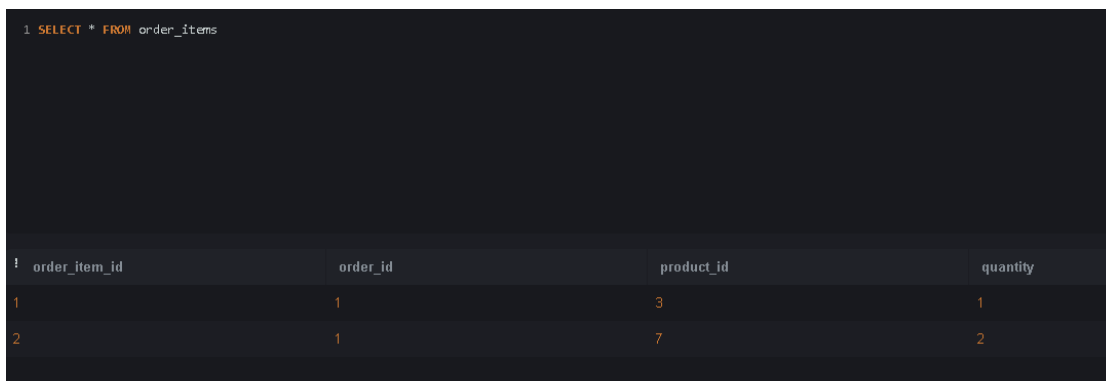


1 SELECT * FROM orders

order_id	order_date	total_amount	customer_id	employee_id
1	2023-06-02	39.95	1	1

Tabela për ruajtjen e produkteve individuale në një porosi

```
CREATE TABLE order_items (  
  order_item_id INT PRIMARY KEY, -- Identifikues unik për çdo element të porosisë  
  order_id INT,                  -- Çelës i huazuar që referencë porosive  
  product_id INT,                -- Çelës i huazuar që referencë produkteve  
  quantity INT,                  -- Sasia e produktit në porosi  
  FOREIGN KEY (order_id) REFERENCES orders(order_id),      -- Mardhënie me tabelën e porosive  
  FOREIGN KEY (product_id) REFERENCES products(product_id) -- Mardhënie me tabelën e produkteve  
);
```



1 SELECT * FROM order_items

order_item_id	order_id	product_id	quantity
1	1	3	1
2	1	7	2

Ju japim vlera tabelave

```
INSERT INTO categories (category_id, category_name)
VALUES
```

```
(1, 'Appetizers'),
(2, 'Main Course'),
(3, 'Desserts'),
(4, 'Beverages');
```

```
INSERT INTO products (product_id, product_name, price, category_id)
VALUES
```

```
(1, 'Chicken Wings', 9.99, 1),
(2, 'Caesar Salad', 7.99, 1),
(3, 'Margherita Pizza', 12.99, 2),
(4, 'Grilled Salmon', 18.99, 2),
(5, 'Chocolate Brownie', 6.99, 3),
(6, 'Cheesecake', 8.99, 3),
(7, 'Coca-Cola', 2.49, 4),
(8, 'Orange Juice', 2.99, 4);
```

```
INSERT INTO customers (customer_id, customer_name, phone_number, email)
VALUES
```

```
(1, 'John Doe', '123-456-7890', 'john.doe@example.com'),
(2, 'Jane Smith', '987-654-3210', 'jane.smith@example.com');
```

```
INSERT INTO employees (employee_id, employee_name, position)
VALUES
```

```
(1, 'Michael Johnson', 'Waiter'),
(2, 'Emily Davis', 'Chef');
```

```
INSERT INTO orders (order_id, order_date, total_amount, customer_id, employee_id)
VALUES (1, '2023-06-02', 39.95, 1, 1);
```

```
INSERT INTO order_items (order_item_id, order_id, product_id, quantity)
VALUES
```

```
(1, 1, 3, 1), -- Margherita Pizza
(2, 1, 7, 2); -- Coca-Cola
```

Me keto query marim informacion mbi porosite dhe mund te kryejme llogaritje matematikore

```
SELECT *
FROM orders;
SELECT *
FROM orders
WHERE customer_id = 1;
SELECT order_id, total_amount
FROM orders;
SELECT customer_id, SUM(total_amount) AS total_spent
FROM orders
GROUP BY customer_id;
SELECT AVG(total_amount) AS average_amount
FROM orders;
SELECT SUM(total_amount) AS total_revenue
FROM orders;
SELECT *
FROM orders
WHERE total_amount = (SELECT MAX(total_amount) FROM orders);
SELECT customer_id, COUNT(*) AS order_count
FROM orders
GROUP BY customer_id;
```

Me keto query mund te bejme lidhje mes dy tabelave

Inner Join:

```
SELECT orders.order_id, orders.order_date, customers.customer_name
FROM orders
INNER JOIN customers ON orders.customer_id = customers.customer_id;
```

Cross Join:

```
SELECT orders.order_id, products.product_name
FROM orders
CROSS JOIN products;
```

Left Join:

```
SELECT orders.order_id, orders.order_date, customers.customer_name
FROM orders
LEFT JOIN customers ON orders.customer_id = customers.customer_id;
```

Right Join:

```
SELECT orders.order_id, orders.order_date, customers.customer_name
FROM orders
RIGHT JOIN customers ON orders.customer_id = customers.customer_id;
```

Procedura

```
CREATE PROCEDURE sp_InsertProduct
    @productName VARCHAR(50),
    @price DECIMAL(10, 2),
    @categoryID INT
AS
BEGIN
    INSERT INTO products (product_name, price, category_id)
    VALUES (@productName, @price, @categoryID);
END;
```

```
CREATE PROCEDURE sp_UpdateProductPrice
    @productID INT,
    @newPrice DECIMAL(10, 2)
AS
BEGIN
    UPDATE products
    SET price = @newPrice
    WHERE product_id = @productID;
END;
```

Ekzekutimi :

```
EXEC sp_InsertProduct 'Pizza Margherita', 12.99, 1;
```

String Functions

```
SELECT UPPER(product_name) AS upper_case_name  
FROM products;
```

-Kthen emrin e produkteve me shkronja të mëdha.

```
SELECT LEFT(emri_klientit, 3) AS inicialet  
FROM klientet;
```

-- Kthen inicialet e emrit të klientëve duke marrë 3 karaktere të para.

Triggers

```
CREATE TRIGGER update_order_total  
AFTER INSERT ON order_items  
FOR EACH ROW  
BEGIN  
    UPDATE orders  
    SET total_price = total_price + NEW.item_price  
    WHERE order_id = NEW.order_id;  
END;
```

-- Kryen përditësimin e cmimit total të porosisë kur shtohet një artikull porosie.

```
CREATE TRIGGER product_delete_trigger  
AFTER DELETE ON products  
FOR EACH ROW  
BEGIN  
    INSERT INTO product_history (product_id, deletion_date)  
    VALUES (OLD.product_id, CURRENT_TIMESTAMP);  
END;
```

-- Regjistron fshirjen e një produkti në tabelën histori_produkti.

Identity Column

```
CREATE TABLE products (  
    product_id INT IDENTITY(1, 1) PRIMARY KEY,  
    product_name VARCHAR(50),  
    price DECIMAL(10, 2)  
);
```

```
INSERT INTO products (product_name, price)  
VALUES ('New Product', 9.99);  
SELECT SCOPE_IDENTITY() AS new_product_id;
```

-- Krijon tabelën e produkteve me një kolonë identiteti.

Unique Constraints

```
ALTER TABLE customers  
ADD CONSTRAINT unique_email  
UNIQUE (email);
```

```
CREATE TABLE orders (  
    order_id INT PRIMARY KEY,  
    customer_id INT,  
    order_date DATE,  
    CONSTRAINT unique_order_customer UNIQUE (order_id, customer_id)  
);
```

-- Siguron që asnjë dy rreshta në tabelën e klientëve të mos kenë të njëjtën vlerë për kolonën e email-it.

Scope Identity

```
INSERT INTO orders (order_date, customer_id)  
VALUES ('2023-06-02', 1);
```

```
SELECT SCOPE_IDENTITY() AS last_order_id;
```

-- Merr ID-në e fundit të futur për porosinë.

@@IDENTITY

```
INSERT INTO customers (customer_name)  
VALUES ('John Doe');
```

```
SELECT @@IDENTITY AS last_customer_id;
```

-- Merr vlerën e fundit të gjeneruar për identitetin në sesionin aktual.

Perfundim

Baza e të dhënave për restorant është një zgjidhje efikase dhe e organizuar për menaxhimin e të dhënave të nevojshme për funksionimin e një restoranti. Me ndihmën e kategorive të produkteve, informacionit të detajuar të produkteve, të dhënave të klientëve, regjistrave të punonjësve dhe gjurmimit të porosive, baza e të dhënave lejon organizimin dhe qasjen e lehtë në të dhënat e rëndësishme për ndërveprimin me klientët dhe menaxhimin e operacioneve të brendshme të restorantit.

Përmes ndërveprimit midis tabelave, bazës së të dhënave mund t'i sigurohet informacioni mbi porositë e klientëve, të dhënat demografike të klientëve, historinë e blerjeve, si dhe të dhënat e stafit të restorantit. Kjo mundëson monitorimin e performancës së restorantit, analizën e preferencave të klientëve dhe sigurimin e një shërbimi më të personalizuar për ta.

Përdorimi i procedurave të ruajtura në bazën e të dhënave lejon automatizimin e detyrave të përditshme, përmirësimin e efikasitetit dhe precizionit në përpunimin e të dhënave. Mundësia për të kryer llogaritje matematikore dhe analiza të thelluara përmes query-ve në bazën e të dhënave lejon njohjen e mëtejshme të performancës së restorantit, identifikimin e trendeve dhe marrjen e vendimeve të informuara për të përmirësuar përvojën e klientëve dhe rezultatet financiare.

Përmbledhtas, baza e të dhënave për restorant është një instrument i rëndësishëm për menaxhimin e të dhënave në një mjedis restoranti. Ajo siguron organizimin e informacionit, përmirëson veprimtarinë operative dhe ndihmon në marrjen e vendimeve strategjike. Duke ofruar qasje të lehtë dhe të shpejtë në të dhënat, baza e të dhënave për restorant ndihmon në optimizimin e efikasitetit dhe në sigurimin e një përvoje të shkëlqyer për klientët.