Öğrenci Adı Soyadı : Muhammed Kızıldağ Özgür Meşe Yasir Arslan

Öğrenci Numarası: 170423960 170423061 170423528

T.C. MARMARA ÜNİVERSİTESİ TEKNOLOJİ FAKÜLTESİ 2025-2026 EĞİTİM ÖĞRETİM YILI GÜZ DÖNEMİ BİLGİSAYAR MÜHENDİSLİĞİ BÖLÜMÜ VERİTABANI YÖNETİM SİSTEMLERİ DERSİ PROJE-FİNAL DÖKÜMANI

Tüm sorular soru metnini altındaki alanlara cevaplandırılacaktır. Bu belgenin formatını bozmayınız, sadece gerekli alanları doldurunuz.

1. **Gerçekleştirdiğiniz veri tabanı projesi için grup arkadaşlarınızın isimlerini yazınız ve projenize ait veri tabanı/diğer yazılım bileşenleri hakkında bilgi veriniz. (5 p)**

Proje Adı: Yemek Sipariş Uygulaması (Food Delivery App)

Muhammed Kızıldağ Özgür Meşe Yasir Arslan

Veritabanı Bileşenleri:

Projemizde veritabanı olarak PostgreSQL 16 kullandık. Veritabanımız Docker container içerisinde çalışmaktadır. Veritabanı şemamızda toplam 10 adet tablo bulunmaktadır:

users: Sistemdeki tüm kullanıcıların bilgilerini tutar (UUID, isim, email, şifre hash, rol)
customer: Müşteri detay bilgilerini tutar (telefon numarası, seçili adres)
restaurant_owner: Restoran sahibi bilgilerini tutar (vergi no, IBAN)
restaurant: Restoran bilgilerini tutar (isim, açıklama, şehir, ilçe, minimum sipariş tutarı, puan)
products: Ürün bilgilerini tutar (isim, fiyat, durum, kategori, resim)
address: Müşteri adres bilgilerini tutar (şehir, ilçe, mahalle, sokak, numara)
cart: Sepet bilgilerini tutar
cart_items: Sepetteki ürünleri tutar (miktar, fiyat, seçenekler)
orders: Sipariş bilgilerini tutar (durum, toplam tutar)
order_details: Sipariş detaylarını tutar (ürün, miktar, birim fiyat)
Diğer Yazılım Bileşenleri:

Backend: Node.js ve Express.js framework'ü kullanılarak REST API geliştirildi. Veritabanı bağlantısı için pg (node-postgres) kütüphanesi kullanıldı. Kimlik doğrulama için JWT (JSON Web Token) ve şifre hashleme için bcrypt kullanıldı.

Frontend: Next.js 15 framework'ü ile React tabanlı kullanıcı arayüzü geliştirildi. Stil için Tailwind CSS kullanıldı.

Veritabanı Yönetimi: Migration işlemleri için node-pg-migrate kütüphanesi kullanıldı. Bu sayede veritabanı şeması versiyon kontrolü altında tutuldu.

Konteynerizasyon: PostgreSQL veritabanı Docker Compose ile konteyner ortamında çalıştırıldı.

2. Gerçekleştirdiğiniz veri tabanı projesi için proje dokümanınızı ve dosyalarınızı içeren herkese açık Github bağlantılarınızı paylaşınız. (5 p)
   https://github.com/muhammedkizildag/VTYS

3. **Gerçekleştirdiğiniz projenin amacını detaylı bir şekilde açıklayınız. (10 p)**

Bu projede bir online yemek sipariş platformu geliştirdik.

Sistemimizde iki tür kullanıcı bulunmaktadır: Müşteri ve Restoran Sahibi. Müşteriler sisteme kayıt olup giriş yapabilir, adreslerini tanımlayabilir, kendi ilçelerindeki restoranları görüntüleyebilir, sepetlerine ürün ekleyebilir ve sipariş verebilir. Restoran sahipleri ise restoranlarını ve ürünlerini yönetebilir, gelen siparişleri takip edebilir.

Veritabanımızda kullanıcılar, müşteriler, restoran sahipleri, restoranlar, ürünler, adresler, sepet, sepet ürünleri, siparişler ve sipariş detayları olmak üzere 10 tablo tasarladık. Tablolar arasında foreign key ilişkileri kurarak veri bütünlüğünü sağladık. Ayrıca yeni kullanıcı kaydında otomatik olarak müşteri ve restoran sahibi kayıtları oluşturan bir trigger tanımladık.

Bu proje sayesinde veritabanı dersinde öğrendiğimiz ilişkisel veritabanı tasarımı, normalizasyon, foreign key, trigger ve SQL sorguları gibi konuları pratiğe dökmüş olduk.

4. **Tasarladığınız veri tabanı mimarisinde hangi tablo ve ilişkileri kullanıldığınızı açıklayınız. (10 p)**

Projemizde PostgreSQL veritabanı kullanarak 10 adet tablo tasarladık. Tablolar ve aralarındaki ilişkiler şu şekildedir:

Tablolar

1. **users:** Tüm kullanıcıların temel bilgilerini tutar (user_id, name, email, password_hash, role, created_at). Role alanı ENUM tipinde olup CUSTOMER veya RESTAURANT_OWNER değerlerini alır.

2. **customer:** Müşterilere özel bilgileri tutar (customer_id, phone_number, selected_address_id).

3. **restaurant_owner:** Restoran sahiplerine özel bilgileri tutar (owner_id, phone_number, tax_id_number, iban).

4. **restaurant:** Restoran bilgilerini tutar (restaurant_id, owner_id, name, description, city, district, min_order_price, rating).

5. **products:** Ürün bilgilerini tutar (product_id, restaurant_id, name, unit_price, status, description, image_url, category).

6. **address:** Adres bilgilerini tutar (address_id, customer_id, title, city, district, neighbourhood, street, number).

7. **cart:** Sepet bilgilerini tutar (cart_id, address_id, restaurant_id, created_at, updated_at).

8. **cart_items:** Sepetteki ürünleri tutar (cart_item_id, cart_id, product_id, quantity, options, price).

9. **orders:** Sipariş bilgilerini tutar (order_id, customer_id, restaurant_id, address_id, status, total_price, created_at).

10. **order_details:** Sipariş detaylarını tutar (order_detail_id, order_id, product_id, quantity, unit_price_at_order).

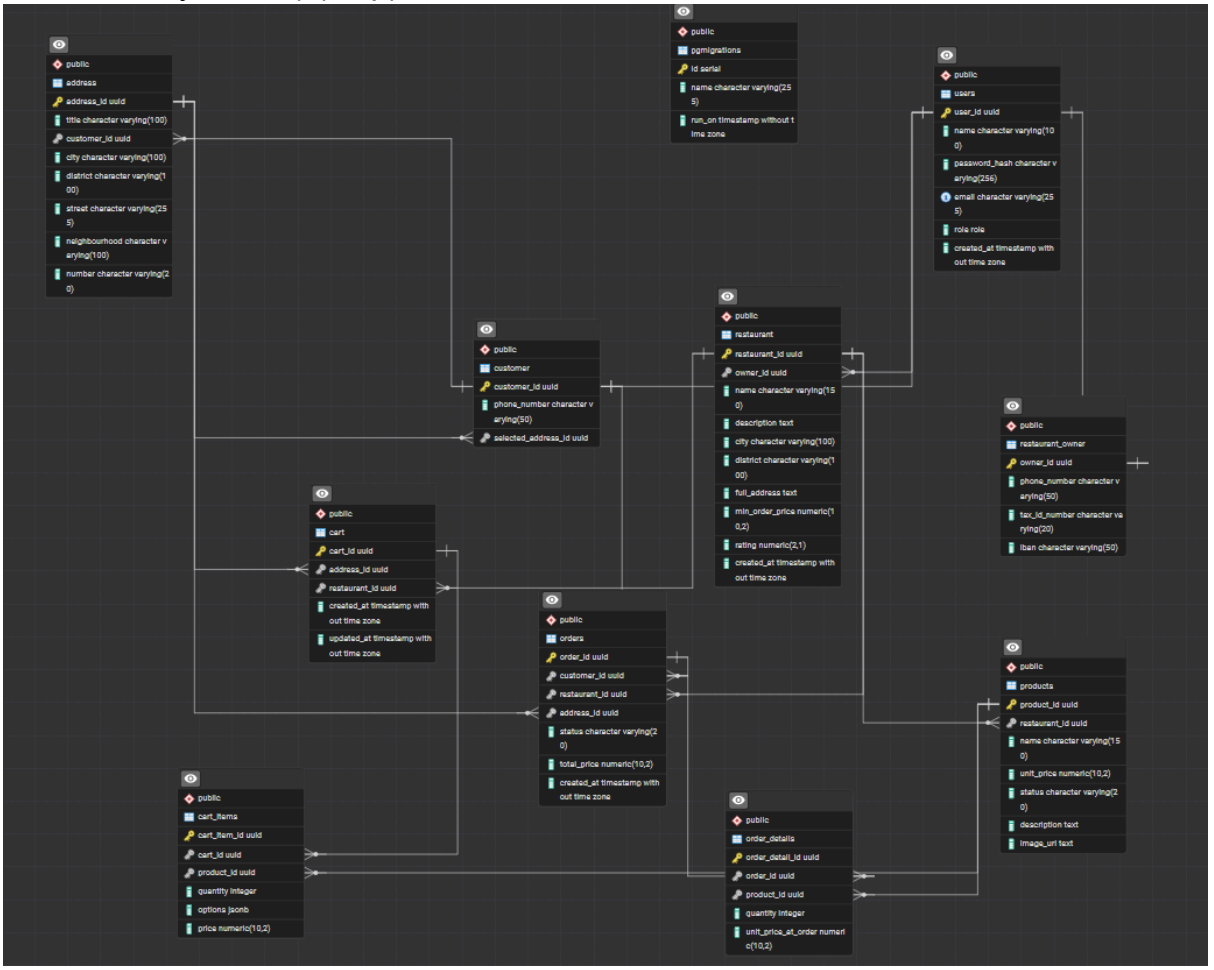
İlişkiler (Foreign Key):

- customer.customer_id → users.user_id (1:1)
- restaurant_owner.owner_id → users.user_id (1:1)
- customer.selected_address_id → address.address_id (N:1)
- restaurant.owner_id → users.user_id (N:1)
- products.restaurant_id → restaurant.restaurant_id (N:1)
- address.customer_id → customer.customer_id (N:1)
- cart.address_id → address.address_id (N:1)
- cart.restaurant_id → restaurant.restaurant_id (N:1)
- cart_items.cart_id → cart.cart_id (N:1)
- cart_items.product_id → products.product_id (N:1)
- orders.customer_id → customer.customer_id (N:1)
- orders.restaurant_id → restaurant.restaurant_id (N:1)
- orders.address_id → address.address_id (N:1)
- order_details.order_id → orders.order_id (N:1)
- order_details.product_id → products.product_id (N:1)

Silme Davranışları:
Tablolarda ON DELETE CASCADE ve ON DELETE SET NULL kuralları tanımladık. Örneğin bir kullanıcı silindiğinde ilgili customer ve restaurant_owner kayıtları da otomatik silinir (CASCADE). Siparişteki ürün silinirse product_id NULL olur (SET NULL) böylece sipariş geçmişi korunur.

5. Veri tabanı ER (Entity Relationship) diagramının bilgisayar ortamında çizilmiş halini paylaşınız. (Ara raporda eksik kısımlar bu raporda giderilmelidir ve ER çizme programlarından faydalanıbilir. Elle çizim, çizip fotoğrafını çekme vb.

kabul edilmeyecektir.) (10 p)



6. **Herhangi iki tablonuz için DDL (create) kodları yazılmalıdır. (10 p)**

Users tablosu

```
CREATE EXTENSION IF NOT EXISTS "pgcrypto";

CREATE TYPE role AS ENUM ('CUSTOMER', 'RESTAURANT_OWNER');

CREATE TABLE users (
    user_id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    password_hash VARCHAR(256) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    role role NOT NULL,
    created_at TIMESTAMP DEFAULT NOW()
);
```

Orders tablosu

```
CREATE TABLE orders (
    order_id UUID DEFAULT gen_random_uuid() PRIMARY KEY,
    customer_id UUID,
    restaurant_id UUID,
    address_id UUID,
```

```
    status VARCHAR(20) NOT NULL CHECK (status IN ('pending', 'confirmed',
'preparing', 'out_for_delivery', 'delivered', 'cancelled')) DEFAULT 'pending',
    total_price DECIMAL(10, 2) NOT NULL CHECK (total_price >= 0),
    created_at TIMESTAMP DEFAULT NOW(),

    CONSTRAINT fk_order_customer
        FOREIGN KEY (customer_id)
        REFERENCES customer(customer_id)
        ON DELETE SET NULL,

    CONSTRAINT fk_order_restaurant
        FOREIGN KEY (restaurant_id)
        REFERENCES restaurant(restaurant_id)
        ON DELETE SET NULL,

    CONSTRAINT fk_order_address
        FOREIGN KEY (address_id)
        REFERENCES address(address_id)
        ON DELETE SET NULL
);
```

7. 5 adet DML (update, insert, delete) içeren kodları yazılmalıdır. (10 p)

1)

```
INSERT INTO users (name, password_hash, email, role)
VALUES ('Ahmet Yılmaz', '$2b$10$xKwG7bQHj8vZmN...', 'ahmet@email.com',
'CUSTOMER');
```

	user_id [PK] uuid	name character varying (100)	password_hash character varying (256)	email character varying (255)	role role	created_at timestamp withou
1	36e138ac-0859-467d-816d-c0eb2eca2...	Muhammed Kızıldağ	$2b$10$v97jM4GY9nS9wgFbSZARjevxHAF9VvkcIu2Eb3ifKm/ip8bUALvkG	mhmmd0536@gmail.c...	CUSTOM...	2025-12-15 20:31
2	3ccf2449-9acc-4178-afb5-e7e1bda38e...	özgür meşe	$2b$10$G65idT9Pcds6qGgD0869g.WkvgnRah5TRhXZaNic6Rs/yNmbA7...	timba@gmail.com	CUSTOM...	2025-12-17 17:38
3	5e6d1822-cc17-4cb9-9078-cf0386eaea...	Ahmet Yılmaz	$2b$10$YTE5KO.5OoAwhJ8RRtfy1umzfrDmhnxdgwpWghszb.6ZdOmK5...	ahmetyilmaz@ornek.co...	CUSTOM...	2025-12-18 03:02
4	176f7745-e22b-48b5-a780-8997bb06e...	Mahmut Adıgüzel	$2b$10$tVRF87G9d12PI5XiHLFo7uQ36g82PnZtRrW7VBworCe1hor4g3...	mahmut@ornek.com	CUSTOM...	2025-12-18 05:33

2)

```
INSERT INTO products (restaurant_id, name, unit_price, status, description,
category)
VALUES ('a1b2c3d4-e5f6-7890-abcd-ef1234567890', 'Karışık Pizza', 149.90,
'available', 'Sucuk, sosis, mantar, mısır, zeytin', 'Pizza');
```

	product_id [PK] uuid	restaurant_id uuid	name character varying (150)	unit_price numeric (10,2)	status character varying (20)	description text	image_url text
1	a1a200a6-bed8-4340-9f2a-dc2552c5...	4dc56a7e-fc79-4c6b-884d-ee918bc2b1...	Burger	3.00	[default]	sd	https://images.unsplash.com
2	e7f8a5ea-d44d-4bb7-b76e-4ccdac51...	4dc56a7e-fc79-4c6b-884d-ee918bc2b1...	Döner	54.00	unavailable	asd	https://images.unsplash.com
3	35518d8b-7e4f-440f-8663-4036422c7...	000d7f03-5f8b-442a-a0a8-0d7e466cd5...	İskender	50.00	unavailable	asd	https://images.unsplash.com
4	072cd641-09c3-4b64-bdb7-cd24238b...	000d7f03-5f8b-442a-a0a8-0d7e466cd5...	Tavuk Pilav	48.00	unavailable	sadd	https://images.unsplash.com
5	5bb30b14-d011-46f0-ab88-003df4db...	000d7f03-5f8b-442a-a0a8-0d7e466cd5...	Kola	55665.00	available	asd	https://images.unsplash.com
6	959ce6c1-bad9-44d8-aafe-cf9d39f17...	c68bfaa4-990a-4c99-b326-bec7b380c3...	Adana Kebab	150.00	available		https://i.nefisyemektarifleri.c

3)

```
UPDATE orders
SET status = 'preparing'
WHERE order_id = 'f1e2d3c4-b5a6-7890-abcd-ef1234567890';
```

4)

```
UPDATE products
SET unit_price = 169.90
WHERE product_id = 'a1b2c3d4-e5f6-7890-abcd-ef1234567890';
```

5)
DELETE FROM cart_items
WHERE cart_item_id = 'c1d2e3f4-a5b6-7890-abcd-ef1234567890';

8. Projenize ait kendi belirlediğiniz toplam 5 adet SQL sorgusu yazınız. Sorguların genel yapısı aşağıdaki ileri seviye kriterleri mutlaka sağlamalıdır:
· En az üç tabloyu birleştiren (JOIN) yapıya sahip olmalı,
· Kümeleme (Aggregate) fonksiyonlarını (COUNT, SUM, AVG, MIN, MAX vb.) içermeli,
· Varsa GROUP BY ve HAVING deyimlerini kapsamalıdır.
Her bir madde için sorgunun amacını, SQL kodunu, cevabını ve sonuç çıktısına ait ekran görüntüsünü raporunuza ekleyiniz. Proje sunum anında bu sorgular SQL ortamında gösterilecek ve açıklanacaktır. (10 p)

1)

```
const result = await query(
  `
    SELECT
        c.cart_id,
        r.restaurant_id,
        r.name AS restaurant_name,
        r.min_order_price,
        p.product_id,
        p.name AS product_name,
        p.image_url,
        ci.quantity,
        ci.price AS unit_price,
        (ci.quantity * ci.price) AS total_line_price,
        ci.options
    FROM
        cart_items ci
    -- 1. Ürünün bağlı olduğu sepeti bul
    INNER JOIN
        cart c ON ci.cart_id = c.cart_id
    -- 2. Sepetin bağlı olduğu adresi bul (KÖPRÜ BURASI)
    INNER JOIN
        address a ON c.address_id = a.address_id
    -- 3. Adresin bağlı olduğu müşteriyi bul
    INNER JOIN
        customer cust ON a.customer_id = cust.customer_id
    -- 4. Sepetin ait olduğu restoranı bul
    INNER JOIN
        restaurant r ON c.restaurant_id = r.restaurant_id
    -- 5. Ürün detaylarını bul
    INNER JOIN
        products p ON ci.product_id = p.product_id
    WHERE
        cust.customer_id = $1
        -- ÖNEMLİ: Sadece müşterinin şu an SEÇİLİ olan adresindeki sepeti getir
        AND cust.selected_address_id = c.address_id
  `,
  [userId]
);
```

	cart_id uuid	restaurant_id uuid	restaurant_name character varying (150)	min_order_price numeric (10,2)	product_id uuid	product_name character varying (150)
1	47ba230e-7dfa-4d8a-a8e7-f32d30b91e...	c68bfaa4-990a-4c99-b326-bec7b380c3...	Adana Kebapçısı	119.00	959ce6c1-bad9-44d8-aafe-cf9d39f17...	Adana Kebab

Bu sorgu; sadece "Sepette ne var?" sorusuna cevap vermez, aynı zamanda "Bu ürün hangi restorandan?", "Ürünün resmi ne?", "Kaç tane var?", "Kullanıcının şu anki seçili adresine mi ait?" sorularının hepsini tek seferde cevaplar.

2)

```
const result = await query(
  `
  SELECT
      c.cart_id,
      r.restaurant_id,
      r.name AS restaurant_name,
      r.min_order_price,
      p.product_id,
      p.name AS product_name,
      p.image_url,
      ci.quantity,
      ci.price AS unit_price,
      (ci.quantity * ci.price) AS total_line_price,
      ci.options
  FROM
      cart_items ci
  -- 1. Ürünün bağlı olduğu sepeti bul
  INNER JOIN
      cart c ON ci.cart_id = c.cart_id
  -- 2. Sepetin bağlı olduğu adresi bul (KÖPRÜ BURASI)
  INNER JOIN
      address a ON c.address_id = a.address_id
  -- 3. Adresin bağlı olduğu müşteriyi bul
  INNER JOIN
      customer cust ON a.customer_id = cust.customer_id
  -- 4. Sepetin ait olduğu restoranı bul
  INNER JOIN
      restaurant r ON c.restaurant_id = r.restaurant_id
  -- 5. Ürün detaylarını bul
  INNER JOIN
      products p ON ci.product_id = p.product_id
  WHERE
      cust.customer_id = $1
      -- ÖNEMLİ: Sadece müşterinin şu an SEÇİLİ olan adresindeki sepeti getir
      AND cust.selected_address_id = c.address_id
`,
  [user_id]
```

	cart_id uuid	restaurant_id uuid	restaurant_name character varying (150)	min_order_price numeric (10,2)	product_id uuid	product_name character varying (150)
1	47ba230e-7dfa-4d8a-a8e7-f32d30b91e...	c68bfaa4-990a-4c99-b326-bec7b380c3...	Adana Kebapçısı	119.00	959ce6c1-bad9-44d8-aafe-cf9d39f17...	Adana Kebab

Bu sorgu, kullanıcının sadece o an seçili olan adresindeki (örneğin "Ev") aktif sepetini ve içindeki ürünleri tüm detaylarıyla (restoran adı, ürün resmi, fiyatlar vb.) getirir.

Çalışma Mantığı:

1. Zincirleme Bağlantı: Sepet adrese bağlı olduğu için, `cart_items` tablosundan başlayıp Adres tablosu üzerinden Müşteri tablosuna ulaşır.
2. Filtreleme (Kritik Nokta): `cust.selected_address_id = c.address_id` satırı sayesinde, kullanıcının diğer kayıtlı adreslerinde (örneğin "İş") kalmış eski sepetleri getirmez, sadece o an işlem yaptığı adresteki sepeti çeker.

3)

```
const result = await query(`
    SELECT
        a.address_id,
        a.title,          -- Örn: Ev, İş
        a.city,
        a.district,
        a.neighbourhood,
        a.street,
        a.number
    FROM
        customer c
    JOIN
        address a ON c.selected_address_id = a.address_id
    WHERE
        c.customer_id = $1
`, [user_id]);
```

	address_id [PK] uuid	title character varying (100)	city character varying (100)	district character varying (100)	neighbourhood character varying (100)	street character varying (255)	number character varying
1	1f8b3ded-7755-4c74-91e7-20384d2ee9...	Ev	İstanbul	Kadıköy	Caferağa	Moda	15

Bu sorgu, belirtilen kullanıcının ($1) o an aktif olarak seçtiği adresin detaylarını (şehir, ilçe, sokak vb.) getirir.

Çalışma Mantığı:

1. Bağlantı (JOIN): `customer` tablosundaki `selected_address_id` sütunu ile `address` tablosundaki `address_id` sütununu eşleştirir.
2. Filtre: Sadece ID'si verilen müşterinin verisini çeker (`WHERE c.customer_id = $1`).
3. Sonuç: Müşterinin kayıtlı tüm adreslerini değil, sadece o an teslimat için seçtiği tek bir adresi döndürür. Eğer müşterinin seçili bir adresi yoksa sorgu boş sonuç döner.

4)

```
const result = await query(
  `
    SELECT
        r.restaurant_id,
        r.name,
        r.description,
        r.rating,
        r.min_order_price,
        r.city,
        r.district
    FROM
        restaurant r
    INNER JOIN
        address a ON r.city = a.city AND r.district = a.district
    INNER JOIN
        customer c ON c.selected_address_id = a.address_id
    WHERE
        c.customer_id = $1
  `,
  [user_id]
);
```

	restaurant_id [PK] uuid	name character varying (150)	description text	rating numeric (2,1)	min_order_price numeric (10,2)	city character varying (100)	district character varying (100)
1	000d7f03-5f8b-442a-a0a8-0d7e466cd5...	döner	dxfsdf	0.0	61.00	İstanbul	Kadıköy
2	c68bfaa4-990a-4c99-b326-bec7b380c3...	Adana Kebapçısı	En güzel adana kebabç...	0.0	119.00	İstanbul	Kadıköy

Bu sorgu, kullanıcının o an seçili olan adresine hizmet veren (yani aynı konumdaki) restoranları listeler.

Çalışma Mantığı:

1. Müşteriden Adrese: `c.selected_address_id` sayesinde müşterinin şu anki teslimat adresini bulur.
2. Konum Eşleştirme (Kritik Nokta): Bulunan adresin İl ve İlçe bilgisini alır; restoranlar tablosundaki İl ve İlçe ile eşleştirir (`ON r.city = a.city AND r.district = a.district`).
3. Sonuç: Kullanıcı İstanbul/Kadıköy adresini seçtiyse, sadece İstanbul/Kadıköy'deki restoranları ekrana getirir. Başka şehir veya ilçedeki restoranlar listelenmez.

5)

```
const result = await query(`
  SELECT
  -- 1. Toplam Ciro
  COALESCE(SUM(total_price), 0) as turnover,

  -- 2. Toplam Sipariş Sayısı
  COUNT(*) as total_order_count,

  -- 3. Yeni Müşteri Sayısı (İlk Kez Sipariş Verenler)
  COUNT(DISTINCT customer_id) FILTER (
      WHERE customer_id NOT IN (
          SELECT customer_id
          FROM orders
          WHERE restaurant_id = $1
            AND created_at < CURRENT_DATE -- Bugünden eski siparişler
            AND status != 'cancelled'
      )
  ) as new_customer_count

  FROM
      orders
  WHERE
      restaurant_id = $1
      AND created_at::date = CURRENT_DATE -- Sadece bugünün kayıtları
      AND status != 'cancelled'
`, [restaurant_id]);
```

	turnover numeric	total_order_count bigint	new_customer_count bigint
1	55665.00	1	1

Bu sorgu, restoran sahibinin paneli (Dashboard) için en kritik olan "Günün Özeti" verilerini tek bir seferde hesaplar.

Sorgu, sadece bugüne ait ve iptal edilmemiş siparişleri havuza alır, ardından bu havuz içinden şu 3 veriyi çıkarır:

1. Toplam Ciro (`turnover`)

- Kod: `COALESCE(SUM(total_price), 0)`
- Anlamı: Bugün kazanılan toplam para.
- Detay: `COALESCE` fonksiyonu, eğer bugün hiç sipariş yoksa sonuç `NULL` gelmesin, ekranda `0` yazsın diye koruma sağlar.

2. Toplam Sipariş Sayısı (`total_order_count`)

- Kod: `COUNT(*)`
- Anlamı: Bugün alınan toplam geçerli sipariş adedi.

3. Yeni Müşteri Sayısı (`new_customer_count`) - *En Zeki Kısım*

- Kod: `COUNT(DISTINCT customer_id) FILTER (WHERE customer_id NOT IN (...))`
- Mantığı:
    1. Bugün sipariş veren müşterilerin ID'lerini alır.
    2. Alt Sorgu (Subquery): Restoranın geçmişteki (`created_at < CURRENT_DATE`) tüm siparişlerine bakar.
    3. Karşılaştırma: Eğer bugün sipariş veren bir müşteri, o "geçmiş listesinde" YOKSA (`NOT IN`), demek ki bu müşteri restorandan ilk kez alışveriş yapıyordur.
    4. Bunu sayar ve "Yeni Kazanılan Müşteri" olarak raporlar.

9. Veri tabanı bağlama ve uygulama geliştirme aşamalarınızı kısaca açıklayarak, kullanıcı ara yüz ekranından bir örnek veriniz. Ve geliştirdiğiniz ara yüzü anlatınız. (10 p)

Projemizde PostgreSQL veritabanını Node.js backend'e bağlamak için pg (node-postgres) kütüphanesini kullandık. Bağlantı bilgilerini .env dosyasında sakladık ve connection pool yöntemiyle veritabanına bağlandık:

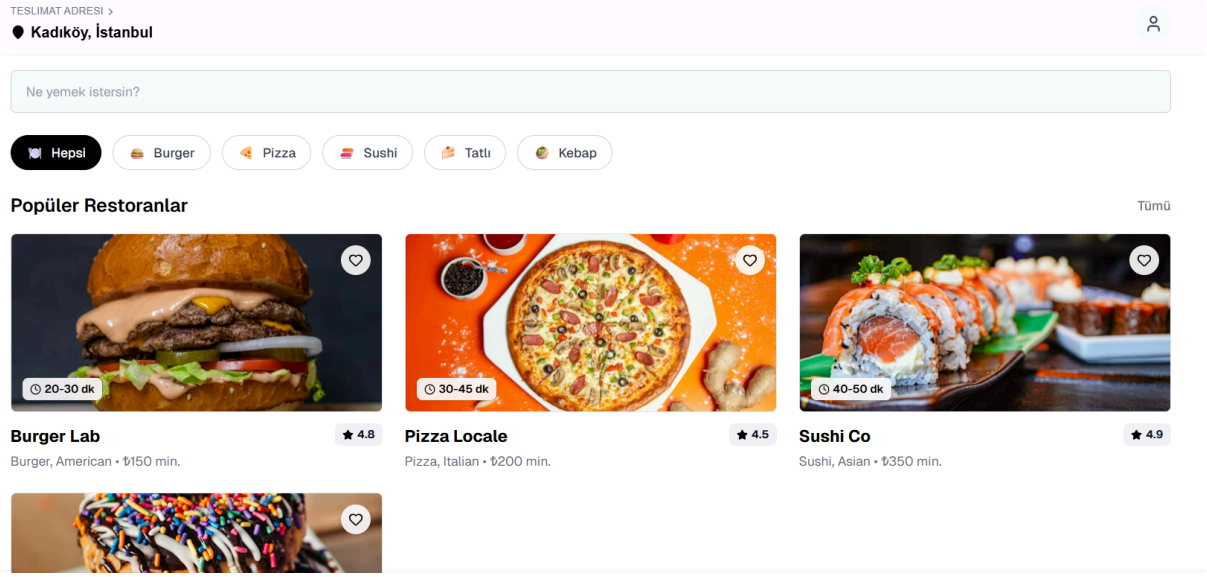
```
import pkg from "pg";
const { Pool } = pkg;

const pool = new Pool({
  connectionString: process.env.DATABASE_URL,
});

export const query = (text, params) => pool.query(text, params);
```

Veritabanı şemasını oluşturmak için node-pg-migrate kütüphanesi ile migration dosyaları yazdık. Bu sayede veritabanı değişikliklerini versiyon kontrolü altında tuttuk.

2. Backend Geliştirme:

Express.js framework'ü ile REST API geliştirdik. Modüler bir yapı kullanarak her özellik için ayrı controller, service ve repository katmanları oluşturduk. Kullanıcı kimlik doğrulaması için JWT token kullandık.

3. Frontend Geliştirme:



Next.js 15 framework'ü ile React tabanlı kullanıcı arayüzü geliştirildi. Tailwind CSS ile stil verdik. Frontend, backend API'ye HTTP istekleri göndererek veritabanı ile iletişim kurar.

Kullanıcı Arayüzü Örneği - Ana Sayfa:

Ana sayfamızda kullanıcılar kendi ilçelerindeki restoranları görebilir. Sayfa üst kısmında arama çubuğu, altında kategori filtreleri ve restoran kartları yer almaktadır. Her restoran kartında restoranın adı, puanı, minimum sipariş tutarı ve teslimat süresi bilgileri gösterilmektedir.

Kullanıcı bir restorana tıkladığında restoran detay sayfasına yönlendirilir. Burada restoranın menüsü kategorilere ayrılmış şekilde listelenir. Ürüne tıklandığında ürün detayları modal pencerede açılır ve sepete ekleme işlemi yapılabilir.

Alt navigasyon çubuğunda Ana Sayfa, Arama, Sepet ve Profil sekmeleri bulunmaktadır. Sepet sayfasında seçilen ürünler, miktarları ve toplam tutar görüntülenir. Profil sayfasından ise adres yönetimi, sipariş geçmişi ve favori restoranlar gibi özelliklere erişilebilir.

10. VTYS sistemlerinde Transaction nedir açıklayınız ve çalışmanızdan bir Transaction örneği veriniz. (10 p)

Transaction, veritabanında bir veya birden fazla SQL komutunun tek bir mantıksal birim olarak çalıştırılmasıdır. Transaction içindeki tüm işlemler ya hep birlikte başarılı olur (COMMIT) ya da hiçbiri uygulanmaz (ROLLBACK). Bu sayede veri bütünlüğü ve tutarlılığı sağlanır.

ACID:

Atomicity (Bölünmezlik): İşlemler ya hep ya hiç şeklinde çalışır.
Consistency (Tutarlılık): İşlem sonrası veritabanı tutarlı kalır.
Isolation (Yalıtım): Eş zamanlı işlemler birbirini etkilemez.

Durability (Kalıcılık): Onaylanan işlemler kalıcı olarak saklanır.

Projemizden Transaction Örneği - Sipariş Oluşturma:

Müşteri sipariş verdiğinde birden fazla tabloya aynı anda kayıt eklenmesi gerekir. Önce orders tablosuna sipariş eklenir, sonra order_details tablosuna siparişteki her ürün için kayıt eklenir ve son olarak cart tablosundan sepet silinir. Bu işlemlerden biri

bile başarısız olursa tutarsız veri oluşur. Bu yüzden transaction kullanmamız gerekir.

```
  // Transaction Başlat
  await query('BEGIN');

  try {
    // 1. Kullanıcının aktif sepetini bul
    const cartRes = await query(
      `SELECT c.cart_id, c.restaurant_id, c.address_id
       FROM cart c
       JOIN customer cust ON c.address_id = cust.selected_address_id
       WHERE cust.customer_id = $1`,
      [userId]
    );
    if (cartRes.rows.length === 0) throw new Error("Aktif sepet veya seçili adres bulunamadı.");
    const cart = cartRes.rows[0];

    // 2. Sepet ürünlerini çek
    // Not: Sepet tablosunda 'cart_items' olduğunu varsayıyoruz.
    const itemsRes = await query(
      `SELECT product_id, quantity, price, options FROM cart_items WHERE cart_id = $1`,
      [cart.cart_id]
    );
    const cartItems = itemsRes.rows;

    if (cartItems.length === 0) throw new Error("Sepetiniz boş.");

    const totalAmount = cartItems.reduce((acc, item) => acc + (Number(item.price) * item.quantity), 0);


    console.log([userId, cart.restaurant_id, cart.address_id, totalAmount])
    // 3. Siparişi Oluştur (Orders Tablosu)
    const orderRes = await query(
      `INSERT INTO orders (customer_id, restaurant_id, address_id, total_price, created_at)
       VALUES ($1, $2, $3, $4, NOW())
       RETURNING order_id`,
      [userId, cart.restaurant_id, cart.address_id, totalAmount]
    );
    const orderId = orderRes.rows[0].order_id;

    // 4. Sipariş Detaylarını Ekle (Order Details Tablosu)
    for (const item of cartItems) {
      await query(
        `INSERT INTO order_details (order_id, product_id, quantity, unit_price_at_order)
         VALUES ($1, $2, $3, $4)`,
        [
          orderId,
          item.product_id,
          item.quantity,
          item.price,
        ]
      );
    }

    // 5. Sepeti Temizle
    await query(`DELETE FROM cart_items WHERE cart_id = $1`, [cart.cart_id]);

    await query('COMMIT');
    return { success: true, orderId };

  } catch (e) {
    await query('ROLLBACK');
    console.error("Sipariş oluşturma hatası:", e);
    throw e;
  }
}
```

BEGIN;

-- 1. Sipariş oluştur
INSERT INTO orders (order_id, customer_id, restaurant_id, address_id, status, total_price)
VALUES ('aaaa-bbbb-cccc-dddd', '1111-2222-3333-4444', 'rrrr-ssss-tttt-uuuu', 'dddd-eeee-ffff-gggg', 'pending', 285.00);

-- 2. Sipariş detaylarını ekle
INSERT INTO order_details (order_id, product_id, quantity, unit_price_at_order)
VALUES ('aaaa-bbbb-cccc-dddd', 'pppp-1111-2222-3333', 2, 120.00);

INSERT INTO order_details (order_id, product_id, quantity, unit_price_at_order)
VALUES ('aaaa-bbbb-cccc-dddd', 'pppp-4444-5555-6666', 1, 45.00);

-- 3. Sepeti temizle
DELETE FROM cart_items WHERE cart_id = 'cccc-1111-2222-3333';
DELETE FROM cart WHERE cart_id = 'cccc-1111-2222-3333';

-- Tüm işlemler başarılıysa onayla
COMMIT;
Eğer bu işlemlerden herhangi biri hata verirse ROLLBACK çalışır ve hiçbir değişiklik veritabanına yansımaz. Örneğin sipariş detayı eklenirken hata oluşursa, daha önce eklenen sipariş kaydı da geri alınır. Bu sayede yarım kalmış siparişler oluşmaz ve veri bütünlüğü korunmuş olur.

11. View nedir açıklayınız ve bir adet view, bir adet saklı yordam (Stored Procedute) ifadesine ait SQL deyimlerinin sorgusunu ve cevabını yazınız. (10 p)

VİEW

```
CREATE OR REPLACE VIEW vw_active_cart_details AS
    SELECT
        cust.customer_id,
        c.cart_id,
        r.restaurant_id,
        r.name AS restaurant_name,
        r.min_order_price,
        p.product_id,
        p.name AS product_name,
        p.image_url AS product_image,
        ci.quantity,
        ci.price AS unit_price,
        (ci.quantity * ci.price) AS total_line_price,
        ci.options,
        a.title AS address_title
    FROM
```

```
        cart_items ci
    INNER JOIN cart c ON ci.cart_id = c.cart_id
    INNER JOIN address a ON c.address_id = a.address_id
    INNER JOIN customer cust ON a.customer_id = cust.customer_id
    INNER JOIN restaurant r ON c.restaurant_id = r.restaurant_id
    INNER JOIN products p ON ci.product_id = p.product_id
    WHERE
        -- En Kritik Nokta: Sadece müşterinin ŞU AN SEÇİLİ adresindeki
sepeti getirir.
        cust.selected_address_id = c.address_id;
```

PROCEDURE

```
CREATE OR REPLACE FUNCTION sp_complete_checkout(p_customer_id UUID)
        RETURNS UUID -- Oluşan Siparişin ID'sini döner
        LANGUAGE plpgsql
        AS $$
        DECLARE
            v_selected_address_id UUID;
            v_cart_id UUID;
            v_restaurant_id UUID;
            v_total_price DECIMAL(10, 2);
            v_new_order_id UUID;
        BEGIN
            -- 1. Müşterinin seçili adresini bul
            SELECT selected_address_id INTO v_selected_address_id
            FROM customer
            WHERE customer_id = p_customer_id;

            IF v_selected_address_id IS NULL THEN
                RAISE EXCEPTION 'Müşterinin seçili bir adresi yok.';
            END IF;

            -- 2. Bu adrese bağlı sepeti ve restoranı bul
            SELECT cart_id, restaurant_id INTO v_cart_id,
v_restaurant_id
            FROM cart
            WHERE address_id = v_selected_address_id;

            IF v_cart_id IS NULL THEN
                RAISE EXCEPTION 'Bu adreste aktif bir sepet
bulunamadı.';
```

```
            END IF;

            -- 3. Sepet tutarını hesapla (Sepet boşsa hata ver)
            SELECT SUM(quantity * price) INTO v_total_price
            FROM cart_items
            WHERE cart_id = v_cart_id;

            IF v_total_price IS NULL OR v_total_price = 0 THEN
                RAISE EXCEPTION 'Sepet boş, sipariş oluşturulamaz.';
            END IF;

            -- 4. ORDERS tablosuna ana kaydı at
            INSERT INTO orders (customer_id, restaurant_id, address_id,
total_price, status)
            VALUES (p_customer_id, v_restaurant_id,
v_selected_address_id, v_total_price, 'pending')
            RETURNING order_id INTO v_new_order_id;

            -- 5. ORDER_DETAILS tablosuna sepet kalemlerini aktar
            -- Not: Senin order_details tablonun create dosyasında
'options' kolonu yoktu.
            -- Eğer options'ı da saklamak istersen önce order_details
tablosuna o kolonu eklemelisin.
            INSERT INTO order_details (order_id, product_id, quantity,
unit_price_at_order)
            SELECT v_new_order_id, product_id, quantity, price
            FROM cart_items
            WHERE cart_id = v_cart_id;

            -- 6. Sepeti temizle (Cart silinmez, içindeki ürünler
silinir)
            DELETE FROM cart_items
            WHERE cart_id = v_cart_id;

            -- 7. Yeni oluşan sipariş ID'sini döndür
            RETURN v_new_order_id;
        END;
        $$;
```