

**TUGAS KELOMPOK 2**  
**QUERY DAN ACTIVE DATABASE FASOR ITS**  
**MANAJEMEN BASIS DATA F**

**Dosen: Ratih Nur Esti Anggraini, S.Kom., M.Sc., Ph.D.**

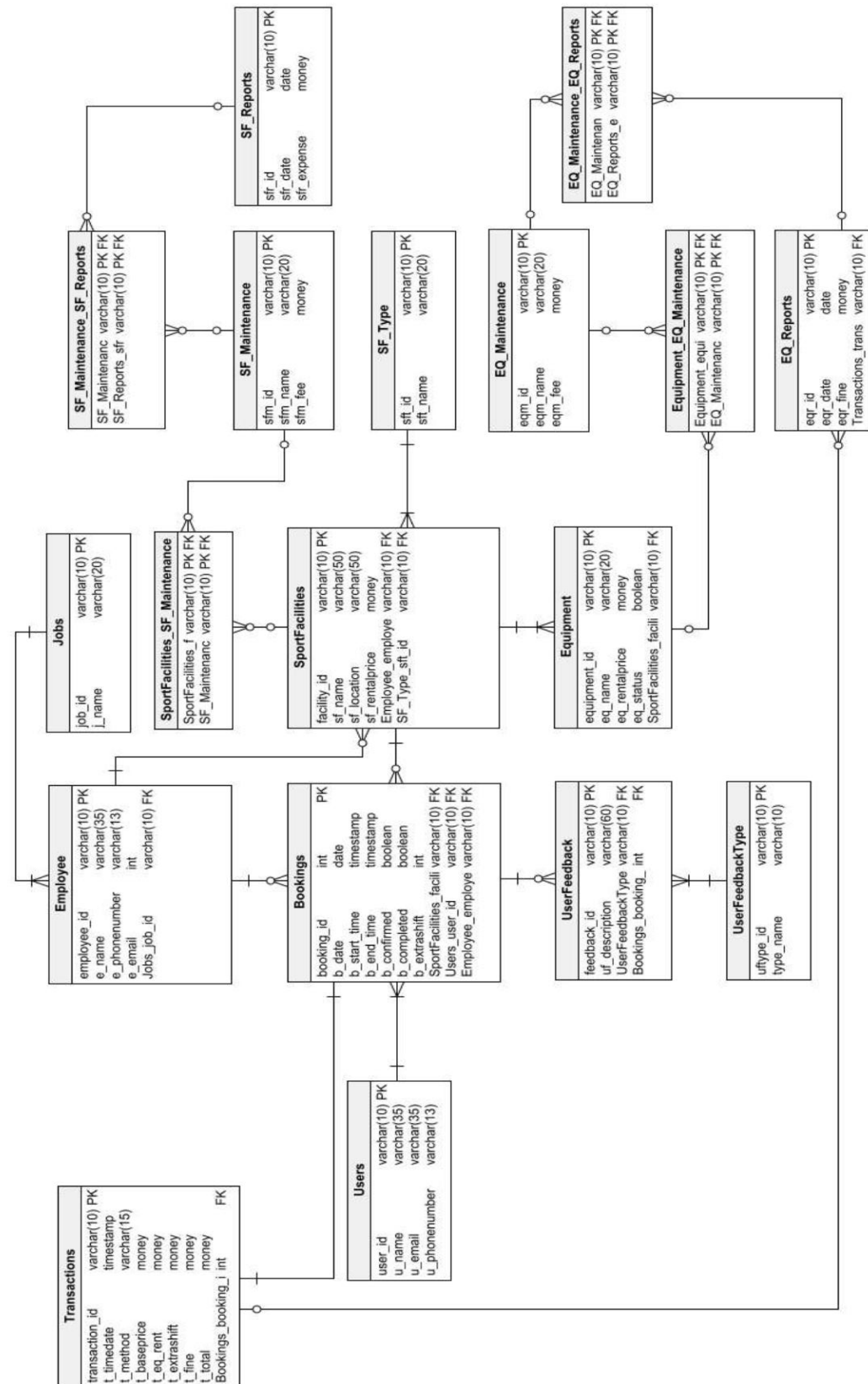


**Anggota Kelompok:**

Arif Nugraha Santosa 5025211048  
Laurivasya Gadhing Syahafidh 5025211136  
Rayhan Arvianta Bayuputra 5025211217  
Ryan Abinugraha 5025211178  
Fihriz Ilham Rabbany 5025211040  
Timothy Hosia Budianto 5025211098

**DEPARTEMEN TEKNIK INFORMATIKA**  
**FAKULTAS TEKNOLOGI ELEKTRO DAN INFORMATIKA CERDAS**  
**(FTEIC)**  
**INSTITUT TEKNOLOGI SEPULUH NOPEMBER**  
**SURABAYA**  
**2023**

## A. RANCANGAN DATABASE



## B. Data Definition Language (DDL)

### - Tabel Transaksi

```
1 CREATE TABLE Transactions(  
2 transaction_id VARCHAR(10) PRIMARY KEY NOT NULL,  
3 t_timedate TIMESTAMP NOT NULL,  
4 t_method VARCHAR(15) NOT NULL,  
5 t_baseprice MONEY NOT NULL,  
6 t_eq_rent MONEY,  
7 t_extrashift MONEY,  
8 t_fine MONEY,  
9 t_total MONEY NOT NULL,  
10 b_booking_id VARCHAR(10) NOT NULL REFERENCES Bookings(booking_id)  
11 );
```

---

### - Tabel Users

```
13 CREATE TABLE Users(  
14 user_id VARCHAR(10) PRIMARY KEY NOT NULL,  
15 u_name VARCHAR(35) NOT NULL,  
16 u_email VARCHAR(35),  
17 u_phonenumber VARCHAR(15) NOT NULL  
18 );
```

### - Tabel Employee

```
20 CREATE TABLE Employee(  
21 employee_id VARCHAR(10) PRIMARY KEY NOT NULL,  
22 e_name VARCHAR(35) NOT NULL,  
23 e_phonenumber VARCHAR(15) NOT NULL,  
24 e_email VARCHAR(35),  
25 j_jobs_id VARCHAR(10) NOT NULL REFERENCES Jobs(jobs_id)  
26 );
```

### - Tabel Bookings

```
28 CREATE TABLE Bookings(  
29 booking_id VARCHAR(10) PRIMARY KEY NOT NULL,  
30 b_date DATE,  
31 b_start_time TIMESTAMP,  
32 b_end_time TIMESTAMP,  
33 b_confirmed BOOL,  
34 b_completed BOOL,  
35 b_extrashift INT,  
36 sf_facility_id VARCHAR(10) NOT NULL REFERENCES SportsFacilities(facility_id),  
37 e_employee_id VARCHAR(10) NOT NULL REFERENCES Employee(employee_id),  
38 u_user_id VARCHAR(10) NOT NULL REFERENCES Users(user_id)  
39 );
```

### - Tabel UserFeedback

```
41 CREATE TABLE UserFeedback(  
42 feedback_id VARCHAR(10) PRIMARY KEY NOT NULL,  
43 uf_description VARCHAR(60) NOT NULL,  
44 uft_uftype_id VARCHAR(10) NOT NULL REFERENCES UserFeedbackType(uftype_id),  
45 b_booking_id VARCHAR(10) NOT NULL REFERENCES Bookings(booking_id)  
46 );
```

#### - Tabel UserFeedbackType

```
48 CREATE TABLE UserFeedbackType(  
49  uftype_id VARCHAR(10) PRIMARY KEY NOT NULL,  
50  type_name VARCHAR(20) NOT NULL  
51 );
```

#### - Tabel Jobs

```
53 CREATE TABLE Jobs(  
54  jobs_id VARCHAR(10) PRIMARY KEY NOT NULL,  
55  j_name VARCHAR(20) NOT NULL  
56 );
```

#### - Tabel SportFacilities

```
66 CREATE TABLE SportsFacilities(  
67  facility_id VARCHAR(10) PRIMARY KEY NOT NULL,  
68  sf_name VARCHAR(50) NOT NULL,  
69  sf_location VARCHAR(50) NOT NULL,  
70  sf_rentalprice MONEY NOT NULL,  
71  e_employee_id VARCHAR(10) NOT NULL REFERENCES Employee(employee_id),  
72  sft_sft_id VARCHAR(10) NOT NULL REFERENCES SF_Type(sft_id)  
73 );
```

#### - Tabel Equipment

```
75 CREATE TABLE Equipment(  
76  equipment_id VARCHAR(10) PRIMARY KEY NOT NULL,  
77  eq_name VARCHAR(20) NOT NULL,  
78  eq_rentalprice MONEY NOT NULL,  
79  eq_status BOOL,  
80  sf_facility_id VARCHAR(10) NOT NULL REFERENCES SportsFacilities(facility_id)  
81 );
```

#### - Tabel SF\_Maintenance

```
91 CREATE TABLE SF_Maintenance(  
92  sfm_id VARCHAR(10) PRIMARY KEY NOT NULL,  
93  sfm_name VARCHAR(20) NOT NULL,  
94  sfm_fee MONEY  
95 );
```

#### - Tabel SF\_Type

```
97 CREATE TABLE SF_Type(  
98  sft_id VARCHAR(10) PRIMARY KEY NOT NULL,  
99  sft_name VARCHAR(20) NOT NULL  
100 );
```

#### - Tabel EQ\_Maintenance

```
102 CREATE TABLE EQ_Maintenance(  
103  eqm_id VARCHAR(10) PRIMARY KEY NOT NULL,  
104  eqm_name VARCHAR(20) NOT NULL,  
105  eqm_fee MONEY  
106 );
```

#### - Tabel EQ\_Reports

```
116 CREATE TABLE EQ_Reports(  
117  eqr_id VARCHAR(10) PRIMARY KEY NOT NULL,  
118  eqr_date DATE NOT NULL,  
119  eqr_fine MONEY,  
120  transaction_id VARCHAR(10) NOT NULL REFERENCES Transactions(transaction_id)  
121 );
```

#### - Tabel SF\_Reports

```
123 CREATE TABLE SF_Reports(  
124   sfr_id VARCHAR(10) PRIMARY KEY NOT NULL,  
125   sfr_date DATE NOT NULL,  
126   sfr_expense MONEY  
127 );
```

## C. Data Manipulation Language (DML)

### Tabel Transaksi

**INSERT INTO Transactions (transaction\_id, t\_timedate, t\_method, t\_baseprice, t\_eq\_rent, t\_extrashift, t\_fine, t\_total, b\_booking\_id)**

**VALUES**

```
('T_001', CURRENT_TIMESTAMP, 'Credit Card', 150000, NULL, NULL, 650000, 800000,  
'B_001'),  
( 'T_002', CURRENT_TIMESTAMP, 'Debit Card', 150000, NULL, 150000, 550000, 850000,  
'B_002'),  
( 'T_003', CURRENT_TIMESTAMP, 'Cash', 100000, NULL, NULL, NULL, 100000, 'B_003'),  
( 'T_004', CURRENT_TIMESTAMP, 'Cash', 200000, NULL, NULL, NULL, 100000, 'B_004');
```

### Tabel Users

**INSERT INTO Users (user\_id, u\_name, u\_email, u\_phonenumber)**

**VALUES**

```
('U_001', 'Anita Widjaja', 'anita.widjaja@gmail.com', '+6281234567890'),  
( 'U_002', 'Bambang Susanto', 'bambang.susanto@yahoo.com', '+6282345678901'),  
( 'U_003', 'Cindy Sari', 'cindy.sari@hotmail.com', '+6283456789012'),  
( 'U_004', 'Dedi Kurniawan', 'dedi.kurniawan@gmail.com', '+6284567890123'),  
( 'U_005', 'Eka Pratiwi', 'eka.pratiwi@yahoo.com', '+6285678901234'),  
( 'U_006', 'Fajar Rahman', 'fajar.rahman@hotmail.com', '+6286789012345'),  
( 'U_007', 'Gita Wulandari', 'gita.wulandari@gmail.com', '+6287890123456'),  
( 'U_008', 'Hendra Prasetya', 'hendra.prasetya@yahoo.com', '+6288901234567'),  
( 'U_009', 'Indra Nugraha', 'indra.nugraha@hotmail.com', '+6289012345678'),  
( 'U_010', 'Joko Santoso', 'joko.santoso@gmail.com', '+6280123456789');
```

### Tabel Employee

**INSERT INTO Employee (employee\_id, e\_name, e\_phonenumber, e\_email, j\_jobs\_id)**

**VALUES**

```
('E_001', 'Aris Prasetyo', '+6281111111111', 'aris.prasetyo@example.com', 'J_001'),  
( 'E_002', 'Budi Santoso', '+6282222222222', 'budi.santoso@example.com', 'J_002'),  
( 'E_003', 'Cahya Kusuma', '+6283333333333', 'cahya.kusuma@example.com', 'J_002'),  
( 'E_004', 'Dewi Sari', '+6284444444444', 'dewi.sari@example.com', 'J_003'),  
( 'E_005', 'Eko Sutrisno', '+6285555555555', 'eko.sutrisno@example.com', 'J_003'),  
( 'E_006', 'Fanny Wijaya', '+6286666666666', 'fanny.wijaya@example.com', 'J_004'),  
( 'E_007', 'Gina Yuliana', '+6287777777777', 'gina.yuliana@example.com', 'J_004');
```

### **Tabel Bookings**

```
INSERT INTO Bookings (booking_id, b_date, b_start_time, b_end_time,
b_confirmed, b_completed, b_extrashift, sf_facility_id, e_employee_id,
u_user_id)
VALUES
('B_001', '2023-04-12', '2023-04-15 18:00:00', '2023-04-15 19:00:00', TRUE,
TRUE, 0, 'F_001', 'E_002', 'U_001'),
('B_002', '2023-04-13', '2023-04-16 10:00:00', '2023-04-16 11:00:00', TRUE,
TRUE, 1, 'F_002', 'E_002', 'U_002'),
('B_003', '2023-04-17', '2023-04-17 14:00:00', '2023-04-17 16:00:00', TRUE,
TRUE, 0, 'F_003', 'E_003', 'U_003'),
('B_004', '2023-04-18', '2023-04-18 09:00:00', '2023-04-18 11:00:00', TRUE,
TRUE, 0, 'F_005', 'E_003', 'U_004'),
('B_005', '2023-04-19', '2023-04-19 16:00:00', '2023-04-19 17:00:00', TRUE,
FALSE, 0, 'F_004', 'E_003', 'U_005'),
('B_006', '2023-04-20', '2023-04-20 11:00:00', '2023-04-20 12:00:00', FALSE,
FALSE, 1, 'F_002', 'E_002', 'U_006'),
('B_007', '2023-04-21', '2023-04-21 19:00:00', '2023-04-21 20:00:00', FALSE,,
FALSE, 0, 'F_003', 'E_002', 'U_007'),
('B_008', '2023-04-22', '2023-04-22 16:00:00', '2023-04-22 17:00:00', TRUE,
FALSE, 0, 'F_005', 'E_003', 'U_008'),
('B_009', '2023-04-23', '2023-04-23 18:00:00', '2023-04-23 19:00:00', TRUE,
FALSE, 0, 'F_001', 'E_003', 'U_009'),
('B_010', '2023-04-24', '2023-04-24 10:00:00', '2023-04-24 11:00:00', TRUE,
FALSE, 1, 'F_004', 'E_002', 'U_010'),
('B_011', '2023-05-15', '2023-05-15 18:00:00', '2023-04-15 19:00:00', TRUE,
TRUE, 0, 'F_001', 'E_002', 'U_001'),
('B_012', '2023-05-16', '2023-05-16 10:00:00', '2023-04-16 11:00:00', TRUE,
TRUE, 1, 'F_002', 'E_002', 'U_002'),
('B_013', '2023-05-17', '2023-05-17 14:00:00', '2023-04-17 16:00:00', TRUE,
TRUE, 0, 'F_003', 'E_003', 'U_003'),
('B_014', '2023-05-16', '2023-05-16 10:00:00', '2023-04-16 11:00:00', TRUE,
TRUE, 1, 'F_002', 'E_002', 'U_002'),
('B_015', '2023-05-25', '2023-05-29 14:00:00', '2023-04-29 16:00:00', TRUE,
TRUE, 0, 'F_003', 'E_003', 'U_003'),
('B_016', '2023-05-16', '2023-05-16 10:00:00', '2023-04-16 11:00:00', TRUE, TRUE,
1, 'F_002', 'E_002', 'U_002'),
('B_017', '2023-05-20', '2023-05-28 14:00:00', '2023-04-28 16:00:00', TRUE,
TRUE, 0, 'F_003', 'E_003', 'U_003'),
('B_018', '2023-05-24', '2023-05-24 10:00:00', '2023-04-24 11:00:00', TRUE,
FALSE, 1, 'F_004', 'E_002', 'U_010');
```

#### **Tabel UserFeedback**

**INSERT INTO UserFeedback (feedback\_id, uf\_description, uft\_uftype\_id, b\_booking\_id)**

#### **VALUES**

('FB\_001', 'Sangat puas dengan layanan yang diberikan', 'uftype\_002', 'B\_001'),  
('FB\_002', 'Kurang puas dengan kondisi lapangan futsal yang kecil', 'uftype\_001', 'B\_002'),  
('FB\_003', 'Pelayanan karyawan sangat ramah', 'uftype\_002', 'B\_003'),  
('FB\_004', 'Lapangan basketnya kotor tidak dibersihkan', 'uftype\_001', 'B\_004');

#### **Tabel UserFeedbackType**

**INSERT INTO UserFeedbackType (uftype\_id, type\_name)**

#### **VALUES**

('uftype\_001', 'kritik dan saran'),  
('uftype\_002', 'apresiasi');

#### **Tabel Jobs**

**INSERT INTO Jobs (jobs\_id, j\_name)**

#### **VALUES**

('J\_001', 'Manajer'),  
('J\_002', 'Petugas Administrasi'),  
('J\_003', 'Petugas Lapangan'),  
('J\_004', 'Petugas Kebersihan');

#### **Tabel SportFacilities**

**INSERT INTO SportsFacilities (facility\_id, sf\_name, sf\_location, sf\_rentalprice, e\_employee\_id, sft\_sft\_id)**

#### **VALUES**

('F\_001', 'Lapangan Futsal A', 'Jl. Ahmad Yani No.1, Surabaya', 150000, 'E\_004', 'SFT\_001'),  
('F\_002', 'Lapangan Futsal B', 'Jl. Diponegoro No.10, Surabaya', 150000, 'E\_004', 'SFT\_001'),  
('F\_003', 'Lapangan Badminton A', 'Jl. Gajah Mada No.50, Surabaya', 100000, 'E\_004', 'SFT\_002'),  
('F\_004', 'Lapangan Basket A', 'Jl. Basuki Rahmat No.100, Surabaya', 200000, 'E\_005', 'SFT\_003'),  
('F\_005', 'Lapangan Basket B', 'Jl. Manyar Kertoarjo No.20, Surabaya', 200000, 'E\_005', 'SFT\_003');

#### **Tabel Equipment**

**INSERT INTO Equipment (equipment\_id, eq\_name, eq\_rentalprice, eq\_status, sf\_facility\_id)**

#### **VALUES**

('EQ\_001', 'Bola Basket Nike', 20000, TRUE, 'F\_004'),  
('EQ\_002', 'Bola Basket Adidas', 20000, TRUE, 'F\_004'),  
('EQ\_003', 'Bola Futsal Putih', 15000, TRUE, 'F\_001'),  
('EQ\_004', 'Bola Futsal Hitam', 15000, TRUE, 'F\_001'),  
('EQ\_005', 'Shuttlecock JP Gold', 25000, TRUE, 'F\_003'),  
('EQ\_006', 'Shuttlecock Mavis', 25000, TRUE, 'F\_003'),  
('EQ\_007', 'Raket Yonex Merah', 35000, TRUE, 'F\_003');

```
('EQ_008', 'Raket Yonex Putih', 35000, TRUE, 'F_003'),
('EQ_009', 'Raket Lining Hitam', 30000, TRUE, 'F_003'),
('EQ_010', 'Raket Lining Merah', 30000, TRUE, 'F_003'),
('EQ_011', 'Bola Basket Nike', 20000, TRUE, 'F_005'),
('EQ_012', 'Bola Basket Adidas', 20000, TRUE, 'F_005'),
('EQ_013', 'Bola Futsal Putih', 15000, TRUE, 'F_002'),
('EQ_014', 'Bola Futsal Hitam', 15000, TRUE, 'F_002');
```

#### **Tabel SF\_Maintenance**

```
INSERT INTO SF_Maintenance (sfm_id, sfm_name, sfm_fee)
VALUES
```

```
('SFM_001', 'Lapangan Futsal A', 750000),
('SFM_002', 'Lapangan Futsal B', 750000),
('SFM_003', 'Lapangan Basket A', 600000),
('SFM_004', 'Lapangan Basket B', 600000),
('SFM_005', 'Lapangan Badminton A', 550000);
```

#### **Tabel SF\_Type**

```
INSERT INTO SF_Type (sft_id, sft_name)
VALUES
```

```
('SFT_001', 'Lapangan Futsal'),
('SFT_002', 'Lapangan Badminton'),
('SFT_003', 'Lapangan Basket');
```

#### **Tabel EQ\_Maintenance**

```
INSERT INTO EQ_Maintenance (eqm_id, eqm_name, eqm_fee)
VALUES
```

```
('EQM_001', 'Bola Basket', 650000),
('EQM_002', 'Bola Futsal', 550000),
('EQM_003', 'Shuttlecock', 50000),
('EQM_004', 'Raket', 150000);
```

#### **Tabel EQ\_Reports**

```
INSERT INTO EQ_Reports (eqr_id, eqr_date, eqr_fine, transaction_id)
VALUES
```

```
('EQR_001', '2023-04-10', 650000, 'T_001'),
('EQR_002', '2023-04-10', 550000, 'T_002');
```

#### **Tabel SF\_Reports**

```
INSERT INTO SF_Reports (sfr_id, sfr_date, sfr_expense)
VALUES
```

```
('SFR_001', '2023-04-09', 750000),
('SFR_002', '2023-04-10', 750000),
```



```
('SFR_003', '2023-04-11', 600000),  
( 'SFR_004', '2023-04-12', 600000),  
( 'SFR_005', '2023-04-13', 550000);
```

## D. QUERY

### 1. Aggregate

#### a. Mendapatkan harga peminjaman rata-rata dari semua fasilitas olahraga

```
SELECT ('Rp' || ROUND(AVG(sf_rentalprice::NUMERIC), 0))::MONEY AS AverageRentalPrice  
FROM SportsFacilities;
```

Query ini digunakan untuk menghitung rata-rata harga sewa fasilitas olahraga dari tabel SportsFacilities, kemudian mengonversi hasilnya menjadi tipe data uang (MONEY) dengan menambahkan simbol rupiah ('Rp') di depan angka dan menghilangkan tanda desimal. Hasil query berupa satu baris dengan satu kolom yang dinamai AverageRentalPrice.

#### b. Menghitung total booking yang dilakukan oleh setiap user

```
SELECT u.u_name, COUNT(b.booking_id) AS TotalBookings  
FROM Users u  
LEFT JOIN Bookings b ON u.user_id = b.u_user_id  
GROUP BY u.u_name;
```

Query ini menggunakan operasi JOIN dengan menggabungkan tabel Users dan Bookings berdasarkan kolom user\_id pada tabel Users dan kolom u\_user\_id pada tabel Bookings. Namun, query ini menggunakan LEFT JOIN agar menampilkan seluruh data dari tabel Users walaupun tidak ada data yang cocok di tabel Bookings.

Selanjutnya, hasil JOIN tersebut dihitung jumlahnya menggunakan fungsi agregat COUNT() pada kolom booking\_id dari tabel Bookings. Data yang dihasilkan dikelompokkan berdasarkan nama user dengan menggunakan GROUP BY pada kolom u\_name dari tabel Users.

#### c. Menampilkan nama dari fasilitas olahraga yang memiliki frekuensi paling banyak pada bulan ini beserta dengan jumlah frekuensinya.

```

283 -- 3. Get the sf that had the most uses this month, as well as the frequency
284 SELECT sf.sf_name, COUNT(*) AS Frequency
285 FROM SportsFacilities sf
286 JOIN Bookings b ON b.sf_facility_id = sf.facility_id
287 WHERE DATE_TRUNC('month', b.b_date) = DATE_TRUNC('month', CURRENT_DATE)
288 GROUP BY sf.sf_name
289 HAVING COUNT(*) = (
290     SELECT MAX(freq) AS Frequency
291     FROM (
292         SELECT COUNT(*) AS freq
293         FROM SportsFacilities sf
294         JOIN Bookings b ON b.sf_facility_id = sf.facility_id
295         WHERE DATE_TRUNC('month', b.b_date) = DATE_TRUNC('month', CURRENT_DATE)
296         GROUP BY sf.sf_name
297     ) subquery
298 )
299 ORDER BY Frequency DESC;

```

Pertama-tama, query ini melakukan operasi JOIN antara tabel SportsFacilities dan Bookings menggunakan kolom sf\_facility\_id pada tabel Bookings dan kolom facility\_id pada tabel SportsFacilities. Kemudian, query ini membatasi data pada bulan ini menggunakan fungsi DATE\_TRUNC() pada kolom b\_date dari tabel Bookings.

Setelah itu, query ini menghitung jumlah frekuensi untuk setiap fasilitas olahraga menggunakan fungsi agregat COUNT(\*) pada tabel Bookings dan mengelompokkannya berdasarkan nama fasilitas olahraga dengan menggunakan GROUP BY.

Selanjutnya, query ini menggunakan subquery pada HAVING untuk memfilter hasil yang hanya menampilkan fasilitas olahraga dengan jumlah frekuensi terbanyak. Subquery tersebut juga menghitung jumlah frekuensi terbanyak pada tabel Bookings dan mengelompokkannya berdasarkan nama fasilitas olahraga dengan menggunakan GROUP BY dan ORDER BY. Hasil dari subquery tersebut diambil hanya satu data teratas dengan menggunakan MAX.

Terakhir, query ini mengurutkan hasilnya berdasarkan frekuensi yang terbanyak dengan menggunakan ORDER BY.

Data Output Messages Notifications		
	sf_name character varying (50)	frequency bigint
1	Lapangan Badminton A	3
2	Lapangan Futsal B	3

Apabila ada 2 lapangan yang memiliki frekuensi yang sama, maka akan tetap ditampilkan keduanya

## 2. Outer Join

### a. Menampilkan data pengguna beserta ID pemesanan yang terkait

```
SELECT u.u_name, b.booking_id
FROM Users u
LEFT OUTER JOIN Bookings b ON b.u_user_id = u.user_id;
```

Query ini melakukan operasi LEFT OUTER JOIN antara tabel Users dan Bookings dengan menggunakan kolom u\_user\_id pada tabel Bookings dan kolom user\_id pada tabel Users.

Pada operasi LEFT OUTER JOIN, semua baris pada tabel Users akan ditampilkan, termasuk jika tidak memiliki pasangan data pada tabel Bookings. Jika pada tabel Bookings tidak ditemukan pasangan data untuk baris pada tabel Users, maka nilai kolom pada tabel Bookings akan diisi dengan nilai NULL.

### b. Menampilkan nama employee serta sport facility yang mereka naungi

```
SELECT e.e_name, sf.sf_name
FROM Employee e
RIGHT OUTER JOIN SportsFacilities sf ON sf.e_employee_id = e.employee_id;
```

Query di atas melakukan RIGHT OUTER JOIN antara dua tabel yaitu Employee dan SportsFacilities dengan membandingkan field employee\_id pada tabel Employee dengan field e\_employee\_id pada tabel SportsFacilities.

Pada hasil JOIN, akan ditampilkan data nama karyawan (e\_name) dan nama fasilitas olahraga (sf\_name) dimana data nama karyawan diambil dari tabel Employee dan data nama fasilitas

olahraga diambil dari tabel SportsFacilities. Namun, karena query tersebut menggunakan RIGHT OUTER JOIN, maka semua data yang ada di tabel SportsFacilities akan ditampilkan, bahkan jika tidak ada data yang cocok pada tabel Employee. Jika tidak ada data yang cocok pada tabel Employee, maka nilai NULL akan ditampilkan pada kolom e\_name.

- c. Menampilkan total jumlah pemesanan dan total pendapatan untuk setiap fasilitas olahraga pada bulan ini.

```
-- 3. Retrieve the sports facility names, the total number of bookings for each sports facility, and the total revenue generated by
SELECT sf.sf_name, COUNT(b.booking_id) AS TotalBookings, COALESCE(CAST(SUM(t.t_total) AS pg_catalog.money), '0.00') AS TotalRevenue
FROM SportsFacilities sf
LEFT JOIN Bookings b ON b.sf_facility_id = sf.facility_id
LEFT JOIN Transactions t ON t.b_booking_id = b.booking_id
GROUP BY sf.sf_name;
```

Pada bagian SELECT, kita memilih kolom sf\_name dari tabel SportsFacilities, menghitung jumlah booking dari tabel Bookings dengan menggunakan COUNT(), dan menghitung total pendapatan dari tabel Transactions dengan menggunakan SUM(). Kemudian kita menggunakan full outer join antara tabel SportsFacilities dengan tabel Bookings dan Transactions. Lalu kita filter untuk hanya mengambil data dari bulan yang sedang berjalan dan hanya memperhitungkan pemesanan yang sudah selesai (completed = TRUE) atau yang belum selesai (completed = NULL). Selanjutnya, data dikelompokkan berdasarkan sf\_name.

Data Output Messages Notifications			
	sf_name character varying (50)	totalbookings bigint	totalrevenue money
1	Lapangan Badminton A	5	Rp100.000
2	Lapangan Futsal A	3	Rp800.000
3	Lapangan Futsal B	5	Rp850.000
4	Lapangan Basket A	3	Rp0
5	Lapangan Basket B	2	Rp100.000

### 3. Nested Query

- a. Menampilkan pengguna yang telah melakukan pemesanan minimal satu kali untuk fasilitas olahraga yang memiliki harga sewa lebih besar dari harga sewa rata-rata semua fasilitas olahraga.

```
SELECT *
FROM Users
WHERE user_id IN (
    SELECT DISTINCT u_user_id
    FROM Bookings
    WHERE sf_facility_id IN (
        SELECT facility_id
        FROM SportsFacilities
        WHERE CAST(sf_rentalprice AS numeric) > (
            SELECT AVG(CAST(sf_rentalprice AS numeric))
            FROM SportsFacilities
        )
    )
);
```

Subquery utama menggunakan operator IN untuk mencari user\_id yang terdapat dalam tabel Bookings. Subquery ini mencari u\_user\_id yang terdapat dalam tabel Bookings dengan syarat bahwa sf\_facility\_id-nya terdapat dalam subquery lainnya.

Subquery yang kedua digunakan untuk mencari facility\_id yang memiliki harga sewa di atas rata-rata. Subquery ini menggunakan operator IN untuk mencari facility\_id yang terdapat dalam tabel SportsFacilities dengan syarat bahwa sf\_rentalprice-nya di atas rata-rata harga sewa dari seluruh fasilitas yang ada.

Dalam subquery kedua ini, digunakan CAST() function untuk mengubah tipe data kolom sf\_rentalprice menjadi numerik agar dapat dibandingkan dengan AVG() function yang akan menghasilkan nilai numerik.

Setelah subquery kedua memberikan hasil, subquery pertama akan mengambil nilai u\_user\_id yang memenuhi syarat dan menampilkan seluruh informasi yang terkait dari tabel Users.

#### b. Menampilkan sport facility yang dibook paling lama pada bulan ini

```
SELECT sf_name, MAX(duration) AS MaxDuration
FROM (
    SELECT sf.sf_name, b.booking_id, (b.b_end_time - b.b_start_time) AS duration
    FROM SportsFacilities sf
    FULL OUTER JOIN Bookings b ON b.sf_facility_id = sf.facility_id
    WHERE DATE_TRUNC('month', COALESCE(b.b_date, b.b_end_time)) = DATE_TRUNC('month', CURRENT_DATE)
    AND b.b_completed = TRUE
) AS subquery
GROUP BY sf_name;
```

Pertama-tama dilakukan join antara tabel SportsFacilities dan Bookings menggunakan full outer join, dimana semua data dari kedua tabel akan diikutsertakan dalam hasil output. Kemudian, dilakukan seleksi dengan WHERE untuk memfilter hanya data bookings yang selesai dan berada di bulan ini. Setelah itu, dilakukan perhitungan durasi booking dengan mengurangi waktu akhir booking dengan waktu awal booking.

Hasil subquery tersebut kemudian dikelompokkan berdasarkan nama fasilitas olahraga (sf\_name) dan dilakukan agregasi dengan menggunakan fungsi MAX untuk mencari durasi paling lama dari setiap fasilitas. Hasil akhirnya adalah daftar fasilitas olahraga beserta durasi booking paling lama yang terjadi pada bulan ini.

## E. ACTIVE DATABASE

### 1. VIEW

#### -----View Detail Fasilitas Olahraga-----

View detail fasilitas olahraga ini terdiri dari hasil penggabungan beberapa tabel yaitu, SportFacilities, Employee, SF\_Reports, dan Bookings. View ini akan menampilkan detail informasi tentang fasilitas olahraga yang berisi id lapangan, nama lapangan, harga sewa, nama pegawai yang bertanggung jawab, termasuk juga status ketersediaannya pada hari ini.

Pada bagian SELECT, view ini memilih beberapa kolom dari tabel SportsFacilities (sf) seperti facility\_id, sf\_name, dan sf\_rentalprice. Kemudian ditambahkan kolom e\_name dari tabel Employee menggunakan INNER JOIN.

Selain itu, terdapat juga sebuah CASE statement yang akan mengevaluasi kondisi ketersediaan fasilitas olahraga. Jika pada hari ini telah terdapat laporan kerusakan fasilitas di tabel SF\_Reports dan SportsFacilities\_SF\_Reports yang terkait, maka fasilitas tersebut akan ditandai sebagai "Tidak Tersedia". Jika pada hari ini terdapat booking (pemesanan) di tabel Bookings pada fasilitas tersebut, maka fasilitas tersebut akan ditandai sebagai "Terpakai". Namun, jika tidak ada kerusakan maupun booking pada fasilitas tersebut pada hari ini, maka fasilitas akan ditandai sebagai "Tersedia".

```

268 CREATE VIEW DetailSportsFacilities AS
269 SELECT sf.facility_id, sf.sf_name, sf.sf_rentalprice, e.e_name,
270        CASE
271            WHEN EXISTS (
272                SELECT 1
273                FROM SF_Reports sfr, SportsFacilities_SF_Reports sfsf
274                WHERE sfsf.sf_facility_id = sf.facility_id
275                AND sfsf.sf_sfm_id = sfr.sfr_id
276                AND sfr.sfr_date = CURRENT_DATE
277            )
278            THEN 'Tidak Tersedia'
279            WHEN EXISTS (
280                SELECT 1
281                FROM Bookings b
282                WHERE b.sf_facility_id = sf.facility_id
283                AND b.b_date = CURRENT_DATE
284            )
285            THEN 'Terpakai'
286            ELSE 'Tersedia'
287        END AS status
288 FROM SportsFacilities sf
289 INNER JOIN Employee e ON sf.e_employee_id = e.employee_id

```

	facility_id character varying (10) 🔒	sf_name character varying (50) 🔒	sf_rentalprice money 🔒	e_name character varying (35) 🔒	status text 🔒
1	F_004	Lapangan Basket A	\$200,000.00	Eko Sutrisno	Tersedia
2	F_005	Lapangan Basket B	\$200,000.00	Eko Sutrisno	Tersedia
3	F_001	Lapangan Futsal A	\$150,000.00	Dewi Sari	Terpakai
4	F_002	Lapangan Futsal B	\$150,000.00	Dewi Sari	Tersedia
5	F_003	Lapangan Badminton A	\$100,000.00	Dewi Sari	Tersedia

### -----View User Bookings-----

View user bookings ini menampilkan informasi booking dari pengguna beserta statusnya. View ini terdiri dari gabungan beberapa kolom yaitu, booking\_id , u\_name, b\_date, b\_start\_time , b\_end\_time, sf\_name, dan status.

Perintah SELECT di dalam view ini mengambil data dari tiga tabel yaitu Bookings, Users, dan SportsFacilities yang dihubungkan melalui relasi foreign key. Kolom-kolom yang diambil dari tabel-tabel tersebut antara lain booking\_id, u\_name, b\_date, b\_start\_time, b\_end\_time, dan sf\_name.

Kolom terakhir, "**status**", dihasilkan melalui penggunaan CASE WHEN statement. Jika b\_confirmed dan b\_completed bernilai TRUE, maka status



booking adalah "Selesai". Jika b\_confirmed TRUE tetapi b\_completed FALSE, maka status booking adalah "Sudah Dikonfirmasi". Jika b\_confirmed FALSE, maka status booking adalah "Belum dikonfirmasi".

Dengan adanya view ini, pengguna dapat melihat daftar booking yang dilakukan beserta statusnya, sehingga memudahkan pengguna dalam mengelola jadwal dan memantau status booking mereka.

```

294 CREATE VIEW UserBookings AS
295 SELECT b.booking_id, u.u_name, b.b_date, b.b_start_time, b.b_end_time, sf.sf_name,
296        CASE
297            WHEN b.b_confirmed = TRUE AND b.b_completed = TRUE THEN 'Selesai'
298            WHEN b.b_confirmed = TRUE AND b.b_completed = FALSE THEN 'Sudah Dikonfirmasi'
299            WHEN b.b_confirmed = FALSE THEN 'Belum dikonfirmasi'
300        END AS status
301 FROM Bookings b
302 INNER JOIN Users u ON b.u_user_id = u.user_id
303 INNER JOIN SportsFacilities sf ON b.sf_facility_id = sf.facility_id

```

	booking_id character var	u_name character varying (35)	b_date date	b_start_time timestamp without time	b_end_time timestamp without time	sf_name character varying (50)	status text
1	B_005	Eka Pratiwi	2023-04-19	2023-04-19 16:00:00	2023-04-19 17:00:00	Lapangan Basket A	Sudah Dikonfirmasi
2	B_008	Hendra Prasetya	2023-04-22	2023-04-22 16:00:00	2023-04-22 17:00:00	Lapangan Basket B	Sudah Dikonfirmasi
3	B_009	Indra Nugraha	2023-04-23	2023-04-23 18:00:00	2023-04-23 19:00:00	Lapangan Futsal A	Sudah Dikonfirmasi
4	B_010	Joko Santoso	2023-04-24	2023-04-24 10:00:00	2023-04-24 11:00:00	Lapangan Basket A	Sudah Dikonfirmasi
5	B_003	Cindy Sari	2023-04-17	2023-04-17 14:00:00	2023-04-17 16:00:00	Lapangan Badminton A	Selesai
6	B_004	Dedi Kurniawan	2023-04-18	2023-04-18 09:00:00	2023-04-18 11:00:00	Lapangan Basket B	Selesai
7	B_001	Anita Widjaja	2023-04-12	2023-04-15 18:00:00	2023-04-15 19:00:00	Lapangan Futsal A	Selesai
8	B_002	Bambang Susanto	2023-04-13	2023-04-16 10:00:00	2023-04-16 11:00:00	Lapangan Futsal B	Selesai
9	B_006	Fajar Rahman	2023-04-20	2023-04-20 11:00:00	2023-04-20 12:00:00	Lapangan Futsal B	Belum dikonfirmasi
10	B_007	Gita Wulandari	2023-04-21	2023-04-21 19:00:00	2023-04-21 20:00:00	Lapangan Badminton A	Belum dikonfirmasi

## 2. Function

### a. Fungsi untuk menampilkan jumlah equipment yang tersedia pada suatu sports facility

```

-- Function untuk menampilkan jumlah equipment yang tersedia pada suatu sports facility:
CREATE OR REPLACE FUNCTION get_available_equipment_count(sf_id VARCHAR)
RETURNS TABLE (eq_count INTEGER) AS $$
BEGIN
    SELECT COUNT(*) INTO eq_count FROM Equipment WHERE sf_facility_id = sf_id AND eq_status = TRUE;
    RETURN QUERY SELECT eq_count;
END;
$$ LANGUAGE plpgsql;

```

Kode membuat fungsi bernama get\_available\_equipment\_count dengan satu parameter input sf\_id yang bertipe VARCHAR dan mengembalikan tabel satu



kolom bernama eq\_count yang bertipe INTEGER. Fungsi ini menggunakan bahasa plpgsql.


Di dalam fungsi, pertama-tama ia mendeklarasikan variabel eq\_count bertipe INTEGER. Ini kemudian menjalankan pernyataan SELECT untuk menghitung jumlah baris dari tabel Peralatan di mana sf\_facility\_id sama dengan parameter input sf\_id dan eq\_status benar. Hitungan yang dihasilkan disimpan dalam variabel eq\_count menggunakan klausa INTO.

Terakhir, fungsi mengembalikan kueri yang memilih nilai eq\_count menggunakan RETURN QUERY. Pernyataan SELECT digunakan untuk mengembalikan nilai eq\_count sebagai satu baris dan kolom.

Fungsi tersebut kemudian dipanggil menggunakan SELECT get\_available\_equipment\_count('F\_004') di mana F\_004 adalah nilai parameter masukan. Keluaran dari fungsi ini adalah satu baris dengan satu kolom bernama eq\_count yang berisi jumlah peralatan yang tersedia untuk ID fasilitas olahraga yang ditentukan.

Sebagai contoh kita akan melihat peralatan yang tersedia di fasilitas F\_004.

```
SELECT get_available_equipment_count('F_004');
```

get_available_equipment_count 	
integer	
1	2

## **b. Fungsi untuk menghitung jumlah transaksi berdasarkan metode pembayaran tertentu**

```
--Function untuk menghitung jumlah transaksi berdasarkan metode pembayaran tertentu
CREATE OR REPLACE FUNCTION count_transactions_by_method(t_method VARCHAR)
RETURNS INTEGER AS $$
BEGIN
    RETURN (SELECT COUNT(*) FROM Transactions WHERE Transactions.t_method = $1);
END;
$$ LANGUAGE plpgsql;
```

Kode di atas membuat fungsi yang disebut "count\_transactions\_by\_method" yang mengambil satu parameter input bertipe VARCHAR dan mengembalikan INTEGER. Fungsi mengambil jumlah semua transaksi dari tabel Transaksi di

mana bidang t\_method sama dengan parameter masukan yang diberikan ke fungsi.

Fungsi didefinisikan menggunakan bahasa PL/pgSQL. Badan fungsi hanya berisi pernyataan RETURN yang mengembalikan nilai hitungan sebagai hasil fungsi. Parameter input fungsi direferensikan menggunakan sintaks "\$1".

Fungsi ini dapat digunakan untuk mendapatkan dengan cepat jumlah transaksi yang dilakukan menggunakan metode pembayaran tertentu dengan memanggilnya dengan metode yang diinginkan sebagai parameter input. Misalnya, memanggil "count\_transactions\_by\_method('Cash')" akan mengembalikan hitungan semua transaksi yang menggunakan uang tunai sebagai metode pembayaran.

Sebagai contoh kita akan menghitung transaksi yang menggunakan metode credit card dan metode cash.

```
SELECT count_transactions_by_method('Credit Card');
```

	count_transactions_by_method integer
1	1

```
SELECT count_transactions_by_method('Cash');
```

	count_transactions_by_method integer
1	2

### 3. Procedure

Prosedur dengan nama `updateStokAlat` mengambil 3 buah parameter sebagai input yaitu `id equipment`, jumlah dipinjam, dan jumlah dikembalikan. Karena pada tabel `Equipemnt` kami belum terdapat kolom stok maka pertama-tama kita akan menambahkan kolom baru dengan nama 'stok' dan melakukan insert value. Tujuan prosedur ini untuk melakukan update stok pada tabel `Equipment` sesuai id `Equipement` yang diberikan berdasarkan parameter `equipemnt` yang dipinjam dan dikembalikan oleh user.

Prosedur akan melakukan update dengan mengurangi stok dari `equipemnt` yang tersedia dengan banyaknya `equipement` yang dipinjam kemudian ditambahkan dengan `equipment` yang sudah dikembalikan. Lalu kami menggunakan `RAISE INFO` untuk menandakan update dari jumlah stok berhasil dengan keterangan 'Stok Berhasil Diupdate'.

Procedure Query :

```
CREATE OR REPLACE PROCEDURE updateStokAlat (
    IN IDEquipment VARCHAR(10),
    IN dipinjam INT,
    IN dikembalikan INT
)
LANGUAGE plpgsql AS $$
BEGIN
    UPDATE Equipment
    SET stok = stok - dipinjam + dikembalikan
    WHERE IDEquipment = equipment_id;
    RAISE INFO 'Stok Berhasil Diupdate';
END $$;
```

Kemudian prosedur tersebut dipanggil dengan fungsi `CALL`. Maka stok `equipement` dengan `id_equipement` 'EQ\_010' akan dikurangi 3 dan ditambah 1.



```
CALL updateStokAlat('EQ_010',3,1);
```

Testing:

```
ALTER TABLE Equipment
ADD COLUMN stok INT;

UPDATE Equipment
SET stok = CASE
    WHEN equipment_id = 'EQ_001' THEN 10
    WHEN equipment_id = 'EQ_002' THEN 20
    WHEN equipment_id = 'EQ_003' THEN 15
    WHEN equipment_id = 'EQ_004' THEN 10
    WHEN equipment_id = 'EQ_005' THEN 25
    WHEN equipment_id = 'EQ_006' THEN 30
    WHEN equipment_id = 'EQ_007' THEN 20
    WHEN equipment_id = 'EQ_008' THEN 15
    WHEN equipment_id = 'EQ_009' THEN 10
    WHEN equipment_id = 'EQ_010' THEN 5
    WHEN equipment_id = 'EQ_011' THEN 20
    WHEN equipment_id = 'EQ_012' THEN 15
    WHEN equipment_id = 'EQ_013' THEN 10
    WHEN equipment_id = 'EQ_014' THEN 5
    ELSE 0
END;

SELECT * FROM Equipment;
```

	equipment_id [PK] character varying (10) 	eq_name character varying (20) 	eq_rentalprice money 	eq_status boolean 	sf_facility_id character varying (10) 	stok integer 
1	EQ_001	Bola Basket Nike	\$20,000.00	true	F_004	10
2	EQ_002	Bola Basket Adidas	\$20,000.00	true	F_004	20
3	EQ_003	Bola Futsal Putih	\$15,000.00	true	F_001	15
4	EQ_004	Bola Futsal Hitam	\$15,000.00	true	F_001	10
5	EQ_005	Shuttlecock JP Gold	\$25,000.00	true	F_003	25
6	EQ_006	Shuttlecock Mavis	\$25,000.00	true	F_003	30
7	EQ_007	Raket Yonex Merah	\$35,000.00	true	F_003	20
8	EQ_008	Raket Yonex Putih	\$35,000.00	true	F_003	15
9	EQ_009	Raket Lining Hitam	\$30,000.00	true	F_003	10
10	EQ_011	Bola Basket Nike	\$20,000.00	true	F_005	20
11	EQ_012	Bola Basket Adidas	\$20,000.00	true	F_005	15
12	EQ_013	Bola Futsal Putih	\$15,000.00	true	F_002	10
13	EQ_014	Bola Futsal Hitam	\$15,000.00	true	F_002	5
14	EQ_010	Raket Lining Merah	\$30,000.00	true	F_003	5

## 4. Trigger

Pada kasus ini, kita akan melakukan pengecekan pada nama user yang memiliki nama yang sama. Apabila ada nama user yang sama maka nama user yang diinputkan setelahnya akan diberikan tambahan (number).

```
CREATE OR REPLACE FUNCTION add_number_to_duplicate_names() RETURNS TRIGGER AS $$
DECLARE
    new_name VARCHAR(35);
    suffix INT;
BEGIN
    SELECT COUNT(*) INTO suffix FROM Users WHERE u_name = NEW.u_name;

    IF suffix > 0 THEN
        new_name := NEW.u_name || '(' || suffix || ')';
        LOOP
            EXIT WHEN NOT EXISTS (SELECT 1 FROM Users WHERE u_name = new_name);
            suffix := suffix + 1;
            new_name := NEW.u_name || '(' || suffix || ')';
        END LOOP;
        NEW.u_name := new_name;
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

Fungsi ini disebut `add_number_to_duplicate_names()`, yang berarti ia bertugas menambahkan angka pada nama pengguna (user) yang duplikat untuk membedakan antara satu pengguna dengan pengguna lain yang memiliki nama yang sama.

Fungsi ini memiliki satu argumen `NEW` yang merupakan row baru yang akan dimasukkan ke dalam tabel `Users` melalui operasi `INSERT` atau diupdate melalui operasi `UPDATE`. Fungsi ini mengambil nama pengguna baru dari argumen `NEW.u_name`, lalu mencari jumlah pengguna lain yang telah memiliki nama yang sama dalam tabel `Users`. Jika fungsi menemukan pengguna lain yang memiliki nama yang sama, maka akan menambahkan angka pada akhir nama pengguna baru untuk membuatnya unik, kemudian nama yang baru dibuat akan dimasukkan kembali ke `NEW.u_name`.

Fungsi ini kemudian mengembalikan `NEW` yang telah diubah atau diupdate dengan nama baru yang unik tersebut. Fungsi ini ditulis dalam bahasa `plpgsql`, yaitu bahasa pemrograman yang digunakan untuk membuat fungsi dan prosedur di dalam database PostgreSQL.

```
CREATE TRIGGER add_number_to_duplicate_names_trigger
BEFORE INSERT ON Users
FOR EACH ROW
EXECUTE FUNCTION add_number_to_duplicate_names();
```





Disini trigger akan ditambahkan pada table “Users”. Trigger akan dijalankan sebelum record baru ditambahkan. Setiap ada record baru yang ditambahkan, maka trigger ini akan berjalan dan menjalankan fungsi diatas.

```
INSERT INTO Users (user_id, u_name, u_email, u_phonenumber)
VALUES
('U_011', 'Anton', 'anton.widjaja@gmail.com', '+6281234567899'),
('U_012', 'Anton', 'anton.susanto@yahoo.com', '+6282345678909'),
('U_013', 'Anton', 'anton.susanti@yahoo.com', '+6282345678909');
```

```
INSERT INTO Users (user_id, u_name, u_email, u_phonenumber)
VALUES
('U_014', 'Anton', 'anton.widjajo@gmail.com', '+6281234567899');
```

```
INSERT INTO Users (user_id, u_name, u_email, u_phonenumber)
VALUES
('U_015', 'Anton', 'anton.widjojo@gmail.com', '+6281234567899');
```

Untuk melakukan testing dimasukan data berikut untuk menguji nama user yang sama. Pada hal ini ditemukan nama “Anton” yang sama.

	user_id [PK] character varying (10) 	u_name character varying (35) 	u_email character varying (35) 	u_phonenumber character varying (15) 
1	U_001	Anita Widjaja	anita.widjaja@gmail.com	+6281234567890
2	U_002	Bambang Susanto	bambang.susanto@yahoo....	+6282345678901
3	U_003	Cindy Sari	cindy.sari@hotmail.com	+6283456789012
4	U_004	Dedi Kurniawan	dedi.kurniawan@gmail.com	+6284567890123
5	U_005	Eka Pratiwi	eka.pratiwi@yahoo.com	+6285678901234
6	U_006	Fajar Rahman	fajar.rahman@hotmail.com	+6286789012345
7	U_007	Gita Wulandari	gita.wulandari@gmail.com	+6287890123456
8	U_008	Hendra Prasetya	hendra.prasetya@yahoo.co...	+6288901234567
9	U_009	Indra Nugraha	indra.nugraha@hotmail.com	+6289012345678
10	U_010	Joko Santoso	joko.santoso@gmail.com	+6280123456789
11	U_011	Anton	anton.widjaja@gmail.com	+6281234567899
12	U_012	Anton(1)	anton.susanto@yahoo.com	+6282345678909
13	U_013	Anton(2)	anton.susanti@yahoo.com	+6282345678909
14	U_014	Anton(3)	anton.widjajo@gmail.com	+6281234567899
15	U_015	Anton(4)	anton.widjojo@gmail.com	+6281234567899

Dapat kita lihat bahwa nama anton dengan id U\_012 hingga U\_015 akan ditambahkan karakter (1) dan akan terus melakukan increment pada akhir kalimatnya.

## 5. Sequence

**Users Sequence, membuat AUTO\_NUMBER pada tabel Users.**

Pada kasus ini, sequence akan digunakan untuk mengisi ID secara otomatis, ketika terjadi penambahan data baru pada tabel Users.

### - Sequence

```
DROP SEQUENCE IF EXISTS users_seq;  
CREATE SEQUENCE users_seq  
    INCREMENT BY 1  
    NO MINVALUE  
    NO MAXVALUE  
    CACHE 1;
```

Perintah di atas merupakan perintah SQL untuk membuat sequence dengan nama "users\_seq". Sequence dimulai dari angka 1 dan setiap kali digunakan, nilai angka akan bertambah 1. Tidak ada nilai minimum atau maksimum yang ditetapkan, dan database akan menyimpan 1 nilai dalam cache setiap kali sequence digunakan.

### - Function

```
SELECT SETVAL('users_seq', (TRIM(LEADING 'U_' FROM user_id)::INTEGER)) FROM users ORDER BY user_id DESC LIMIT 1;  
CREATE OR REPLACE FUNCTION add_new_users()  
    RETURNS TRIGGER  
    AS $add_new_users$  
DECLARE  
    next_val INT = NEXTVAL('users_seq');  
    digits INTEGER := CEIL(LOG(100, next_val + 1))+2;  
BEGIN  
    IF NEW.user_id IS NULL THEN  
        NEW.user_id := 'U_' || LPAD(next_val::text, digits, '0');  
    END IF;  
    RETURN NEW;  
END;  
$add_new_users$ LANGUAGE plpgsql;
```

Dengan menggunakan perintah tersebut, sebuah fungsi baru dengan nama "add\_new\_users" akan dibuat. Fungsi tersebut akan mengembalikan tipe data trigger. Fungsi ini akan mengecek jika kolom "user\_id" pada record baru (NEW) bernilai NULL, jika iya maka fungsi akan mengisi nilai tersebut dengan format string 'U\_' dan angka sequence dari sequence 'users\_seq' dengan format 3 digit angka, yang depannya akan diisi dengan angka 0. Kemudian, fungsi akan mengembalikan record baru (NEW), yang sudah diubah oleh fungsi. Terakhir,



`\$add\_new\_users\$ LANGUAGE plpgsql` digunakan untuk mengeaskan bahwa fungsi ini ditulis dalam bahasa plpgsql.

#### - Trigger

```
DROP TRIGGER IF EXISTS add_new_users ON users;  
CREATE TRIGGER add_new_users BEFORE INSERT ON Users  
FOR EACH ROW EXECUTE PROCEDURE add_new_users();|
```

Sebuah trigger dengan nama "add\_new\_users" akan dibuat pada table "Users". Trigger ini akan dijalankan sebelum record baru (NEW) ditambahkan ke table "Users". Setiap kali sebuah record baru (NEW) ditambahkan, trigger ini akan diaktifkan dan akan menjalankan fungsi "add\_new\_users" di atas.

#### - Insert

```
1 INSERT INTO Users (u_name, u_email, u_phonenumber)  
2 VALUES  
3 ('Anita Widjaja', 'anita.widjaja@gmail.com', '+6281234567890'),  
4 ('Bambang Susanto', 'bambang.susanto@yahoo.com', '+6282345678901'),  
5 ('Cindy Sari', 'cindy.sari@hotmail.com', '+6283456789012'),  
6 ('Dedi Kurniawan', 'dedi.kurniawan@gmail.com', '+6284567890123'),  
7 ('Eka Pratiwi', 'eka.pratiwi@yahoo.com', '+6285678901234'),  
8 ('Fajar Rahman', 'fajar.rahman@hotmail.com', '+6286789012345'),  
9 ('Gita Wulandari', 'gita.wulandari@gmail.com', '+6287890123456'),  
10 ('Hendra Prasetya', 'hendra.prasetya@yahoo.com', '+6288901234567'),  
11 ('Indra Nugraha', 'indra.nugraha@hotmail.com', '+6289012345678'),  
12 ('Joko Santoso', 'joko.santoso@gmail.com', '+6280123456789');
```

Untuk memasukkan data, tidak perlu mengisi ID dari user, karena sudah ditangani oleh perintah di atas.

#### - Output

```
SELECT user_id "User ID", u_name "Nama", u_email "E-Mail", u_phonenumber "No Hp" FROM Users;  
SELECT user_id "User ID", u_name "Nama", u_email "E-Mail", u_phonenumber  
"No Hp" FROM Users;
```



18	U_018	Hendra Prasetya	hendra.prasetya@yahoo.co...	+6288901234567
19	U_019	Indra Nugraha	indra.nugraha@hotmail.com	+6289012345678
20	U_020	Joko Santoso	joko.santoso@gmail.com	+6280123456789
21	U_999	Anita Widjaja	anita.widjaja@gmail.com	+6281234567890
22	U_1000	Anita Widjaja	anita.widjaja@gmail.com	+6281234567890
23	U_1001	Bambang Susanto	bambang.susanto@yahoo....	+6282345678901
24	U_1002	Cindy Sari	cindy.sari@hotmail.com	+6283456789012
25	U_1003	Dedi Kurniawan	dedi.kurniawan@gmail.com	+6284567890123
26	U_1004	Eka Pratiwi	eka.pratiwi@yahoo.com	+6285678901234
27	U_1005	Fajar Rahman	fajar.rahman@hotmail.com	+6286789012345
28	U_1006	Gita Wulandari	gita.wulandari@gmail.com	+6287890123456
29	U_1007	Hendra Prasetya	hendra.prasetya@yahoo.co...	+6288901234567
30	U_1008	Indra Nugraha	indra.nugraha@hotmail.com	+6289012345678
31	U_1009	Joko Santoso	joko.santoso@gmail.com	+6280123456789