AFRICAN CENTRE OF EXCELLENCE IN DATA SCIENCE

COLLEGE OF BUSINESS & ECONOMICS

UNIVERSITY OF RWANDA (UR GIKONDO CAMPUS)

COLLEGE OF BUSINESS AND ECONOMICS (CBE)

NAMES: KABWALI MASUDI Dischon

REGISTRATION NUMBER: 223027551

SCHOOL: ACE-DS, COHORT 6 PROGRAM:

MSc IN DATA SCIENCE IN DATA MINING

ACADEMIC YEAR: 2024-2025

SPECIALIZATION MODULE: ADVANCED DATABASED TECHNOLOGY

ASSIGNMENT 2: SQL DATABASES (ORACLES DEVELOPER/POSTGRESQL)

1. SQL case study title: Smart Parking Management and Ticketing System

1.1. Case Study Description

The Parking Management System monitors parking spaces, vehicles, tickets, staff, and payments. It ensures efficient parking allocation, revenue management, and occupancy monitoring. This subject combines the concepts of managing parking spaces intelligently and handling the ticketing process, such as payments and access.

1.2. Tables to create

1. ParkingLot(LotID, Name, Location, Capacity, Status)

2. Space(SpaceID, LotID, SpaceNo, Status, Type)

3. Vehicle(VehicleID, PlateNo, Type, OwnerName, Contact)

4. Ticket(TicketID, SpaceID, VehicleID, EntryTime, ExitTime, Status)

5. Staff(StaffID, FullName, Role, Contact, Shift)

6. Payment(PaymentID, TicketID, Amount, PaymentDate, Method)

1.3. Relationships of entities

No	Entities related	Cardinality	Relationship name	Meaning
1	Parking → Space	1 : N	contains or has	a parking lot contains many spaces
2	Space → Ticket	1 : N	allocates or is assigned to	a space is allocated to many tickets
3	Vehicle → Ticket	1 : N	is linked to or issues	a vehicle may have many tickets
4	Ticket → Payment	1 : 1	generates or is settled by	each ticket generates one payment
5	Staff → Ticket	1 : N	processes or manages	a staff member processes many tickets

1.4. Tasks to Perform

1. Define all six tables with PK, FK, CHECK constraints.

2. Apply CASCADE DELETE between Ticket → Payment.

3. Insert 5 parking lots and 10 vehicles.

4. Retrieve all occupied spaces with vehicle details.

5. Update payment status upon vehicle exit.

6. Identify parking lots nearing full capacity.

7. Create a view showing total revenue per lot.

8. Implement a trigger to mark space as available after payment completion

2. PostgreSQL solutions to tasks to perform + screenshots

2.1. Database Overview

The parkingticketingsystem database, developed using PostgreSQL, is designed to efficiently manage the core operations of a smart parking environment. Its primary goal is to handle and automate the management of parking lots, vehicles, tickets, staff, and payments while ensuring accurate tracking, streamlined workflows, and optimized resource utilization. Through its structured design, the system supports efficient monitoring of parking activities, enhances operational decision-making, and improves overall service delivery.

2.2. SQL implementation on postgreSQL shell

2.2.1. Create Database

2.2.1.1. Create Database

CREATE DATABASE parkingticketingsystem;

2.2.1.2. Connect to parkingticketingsystem database: \c parkingticketingsystem;

```
postgres=# \c parkingticketingsystem;
You are now connected to database "parkingticketingsystem" as user "postgres".
parkingticketingsystem=#
```

2.2.2. Create Tables with PK, FK, CHECK Constraints

2.2.2.1. Enable UUID (Universal Unique Identifier for unique identifiers)

```
CREATE EXTENSION IF NOT EXISTS "uuid-ossp";
```

2.2.2.2. Creation of the tables in parkingticketingsystem database

2.2.2.2.1. ParkingLot Table

```
CREATE TABLE ParkingLot (
    LotID UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    Name VARCHAR(100) NOT NULL,
    Location VARCHAR(150) NOT NULL,
    Capacity INT NOT NULL CHECK (Capacity > 0),
    Status VARCHAR(20) DEFAULT 'Open' CHECK (Status IN ('Open', 'Closed'))
);
```

2.2.2.2.2. Space Table

```
CREATE TABLE Space (
    SpaceID UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    LotID UUID NOT NULL REFERENCES ParkingLot(LotID) ON DELETE CASCADE,
    SpaceNo VARCHAR(10) NOT NULL,
    Status VARCHAR(20) DEFAULT 'Available' CHECK (Status IN ('Available', 'Occupied')),
    Type VARCHAR(20) CHECK (Type IN ('Compact', 'Large', 'Electric', 'Handicap'))
);
```

2.2.2.2.3. Vehicle Table

```
CREATE TABLE Vehicle (
    VehicleID UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    PlateNo VARCHAR(20) UNIQUE NOT NULL,
    Type VARCHAR(20) CHECK (Type IN ('Car', 'Motorcycle', 'Truck', 'Bus')),
    OwnerName VARCHAR(100) NOT NULL,
    Contact VARCHAR(15)
);
```

2.2.2.2.4. Staff Table

```
CREATE TABLE Staff (
    StaffID UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    FullName VARCHAR(100) NOT NULL,
    Role VARCHAR(50) CHECK (Role IN ('Attendant', 'Supervisor', 'Manager')),
    Contact VARCHAR(15),
    Shift VARCHAR(20) CHECK (Shift IN ('Morning', 'Evening', 'Night'))
);
```

2.2.2.2.5. Ticket Table

```
CREATE TABLE Ticket (
    TicketID UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    SpaceID UUID NOT NULL REFERENCES Space(SpaceID) ON DELETE CASCADE,
    VehicleID UUID NOT NULL REFERENCES Vehicle(VehicleID) ON DELETE CASCADE,
    StaffID UUID REFERENCES Staff(StaffID),
    EntryTime TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP,
    ExitTime TIMESTAMP,
    Status VARCHAR(20) DEFAULT 'Active' CHECK (Status IN ('Active', 'Closed'))
);
```

2.2.2.2.6. Payment Table

```
CREATE TABLE Payment (
    PaymentID UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    TicketID UUID UNIQUE NOT NULL REFERENCES Ticket(TicketID) ON DELETE
CASCADE,
    Amount NUMERIC(10,2) CHECK (Amount >= 0),
    PaymentDate TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    Method VARCHAR(20) CHECK (Method IN ('Cash', 'Card', 'Mobile'))
);
```

2.2.2.2.7. To look at created tables of parkingticketingsystem database

parkingticketingsystem=# \dt or parkingticketingsystem=# \d

```
parkingticketingsystem=# \d
              List of relations
 Schema |     Name     | Type  |  Owner
--------+--------------+-------+----------
 public | lotrevenue   | view  | postgres
 public | parkinglot   | table | postgres
 public | payment      | table | postgres
 public | space        | table | postgres
 public | staff        | table | postgres
 public | ticket       | table | postgres
 public | vehicle      | table | postgres
(7 rows)
```

2.3. Insert Sample Data

2.3.1. Insert Sample Data for each created table

2.3.1.1. Parking Lots

INSERT INTO ParkingLot (Name, Location, Capacity, Status)

VALUES

('Lot A', 'Downtown', 50, 'Closed'),

('Lot B', 'Airport Road', 100, 'Open'),

('Lot C', 'City Mall', 80, 'Open'),

('Lot D', 'University Campus', 70, 'Closed'),

('Lot E', 'Hospital', 60, 'Open');

2.3.1.2. Spaces

INSERT INTO Space (LotID, SpaceNo, Status, Type)

SELECT LotID, 'A' || i, 'Available', 'Compact'

FROM ParkingLot, generate_series(1,2) AS s(i)

LIMIT 10;

2.3.1.3. Vehicles

INSERT INTO Vehicle (PlateNo, Type, OwnerName, Contact)

VALUES

('RBC-101', 'Bus', 'John Doe', '0788000001'),

('RBC-102', 'Car', 'Jane Smith', '0788000002'),

('RBC-103', 'Truck', 'Paul Adams', '0788000003'),

('RBC-104', 'Motorcycle', 'Alice Brown', '0788000004'),

('RBC-105', 'Car', 'Kevin White', '0788000005'),

('RBC-106', 'Car', 'Maria Green', '0788000006'),

('RBC-107', 'Bus', 'Robert Black', '0788000007'),

('RBC-108', 'Truck', 'David Lee', '0788000008'),

('RBC-109', 'Car', 'Linda Young', '0788000009'),

('RBC-110', 'Motorcycle', 'Chris Hall', '0788000010');

2.3.1.4. Staff

INSERT INTO Staff (FullName, Role, Contact, Shift)

VALUES

('Eric Ndayisaba', 'Attendant', '0788000101', 'Morning'),

('Martha Uwimana', 'Supervisor', '0788000102', 'Evening');

2.3.1.5. Tickets

INSERT INTO Ticket (SpaceID, VehicleID, StaffID, EntryTime, Status)

SELECT s.SpaceID, v.VehicleID, (SELECT StaffID FROM Staff LIMIT 1), NOW(), 'Active'

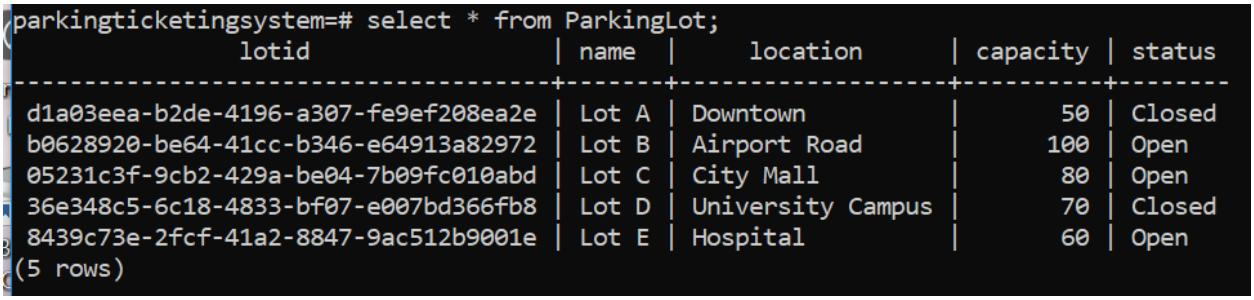
FROM Space s JOIN Vehicle v ON TRUE LIMIT 5;

2.3.1.6. Payments (initially empty)

Note: Payments table/Payments are being inserted after vehicles exit.

2.3.1.7. To look at the structure/contents of each created table

parkingticketingsystem=# select * from ParkingLot;



```
parkingticketingsystem=# select * from ParkingLot;
                lotid                 | name  |     location      | capacity | status
--------------------------------------+-------+-------------------+----------+--------
 d1a03eea-b2de-4196-a307-fe9ef208ea2e | Lot A | Downtown          |       50 | Closed
 b0628920-be64-41cc-b346-e64913a82972 | Lot B | Airport Road      |      100 | Open
 05231c3f-9cb2-429a-be04-7b09fc010abd | Lot C | City Mall         |       80 | Open
 36e348c5-6c18-4833-bf07-e007bd366fb8 | Lot D | University Campus |       70 | Closed
 8439c73e-2fcf-41a2-8847-9ac512b9001e | Lot E | Hospital          |       60 | Open
(5 rows)
```

parkingticketingsystem=# select * from Space;

```
parkingticketingsystem=# select * from Space;
              spaceid               |               lotid                | spaceno |  status   |  type
------------------------------------+------------------------------------+---------+-----------+---------
 29e54dc1-9344-4139-8ce0-81c7e2f01fa2 | d1a03eea-b2de-4196-a307-fe9ef208ea2e | A1      | Available | Compact
 7a817b48-3b18-473d-a66e-639185ea14b1 | b0628920-be64-41cc-b346-e64913a82972 | A1      | Available | Compact
 804ee2b4-4490-49ae-8d32-9a707d5456b0 | 05231c3f-9cb2-429a-be04-7b09fc010abd | A1      | Available | Compact
 3584d914-49ff-4117-aa36-9011adfc91cf | 36e348c5-6c18-4833-bf07-e007bd366fb8 | A1      | Available | Compact
 bf4fa0a7-b59f-49f8-8f3a-302f65cfd7c9 | 8439c73e-2fcf-41a2-8847-9ac512b9001e | A1      | Available | Compact
 e8d7b59d-80de-4927-8a5a-996b481805da | d1a03eea-b2de-4196-a307-fe9ef208ea2e | A2      | Available | Compact
 c0d69528-e549-4448-b112-eb36aed45272 | b0628920-be64-41cc-b346-e64913a82972 | A2      | Available | Compact
 8a941cf8-2547-48f3-93f2-c094dc1ea31a | 05231c3f-9cb2-429a-be04-7b09fc010abd | A2      | Available | Compact
 7e7c1e10-872b-40b5-a780-58c44ccb13d5 | 36e348c5-6c18-4833-bf07-e007bd366fb8 | A2      | Available | Compact
 f4286901-75d4-4b88-a4fe-50ee12140298 | 8439c73e-2fcf-41a2-8847-9ac512b9001e | A2      | Available | Compact
(10 rows)
```

parkingticketingsystem=# select * from Vehicle;

```
parkingticketingsystem=# select * from Vehicle;
              vehicleid               | plateno |    type    |  ownername   |  contact
--------------------------------------+---------+------------+--------------+------------
 414c11fb-0822-403f-82cb-848f27bfe663 | RBC-101 | Bus        | John Doe     | 0788000001
 9a99b4c1-04e4-4d60-af3a-02230ee7c23e | RBC-102 | Car        | Jane Smith   | 0788000002
 7e00dae2-d3ab-4498-a72c-74abbf959d36 | RBC-103 | Truck      | Paul Adams   | 0788000003
 740cda15-30fe-4602-ac19-7e81b17deabc | RBC-104 | Motorcycle | Alice Brown  | 0788000004
 9544609a-a005-4301-b681-d0d715295b82 | RBC-105 | Car        | Kevin White  | 0788000005
 bb5bd92a-bf06-4927-8636-bf24b126e479 | RBC-106 | Car        | Maria Green  | 0788000006
 2836eddf-f37b-4685-9354-9a0b7359fae5 | RBC-107 | Bus        | Robert Black | 0788000007
 50288c6c-ae8f-45bf-b3bb-9a0d888c86fc | RBC-108 | Truck      | David Lee    | 0788000008
 c9e7b9c8-59cc-4c7a-984f-da78a3d116f2 | RBC-109 | Car        | Linda Young  | 0788000009
 732cc593-5249-498e-a35c-8d104801ea70 | RBC-110 | Motorcycle | Chris Hall   | 0788000010
(10 rows)
```

parkingticketingsystem=# select * from Staff;

```
parkingticketingsystem=# select * from Staff;
              staffid               |    fullname    |    role    |  contact   |  shift
------------------------------------+----------------+------------+------------+---------
 376d1533-a9c8-4a50-9036-8779df86da8d | Eric Ndayisaba | Attendant  | 0788000101 | Morning
 a3ff7030-e306-4450-9432-897c85b9e2d2 | Martha Uwimana | Supervisor | 0788000102 | Evening
(2 rows)
```

parkingticketingsystem=# select * from Ticket;

```
parkingticketingsystem=# select * from Ticket;
              ticketid              |              spaceid               |              vehicleid             |              staffid               |        entryti
me         |         exittime          | status
--------------------------------------+------------------------------------+------------------------------------+------------------------------------+---------------
-----------+---------------------------+--------
 080e7f23-f010-439c-99ab-c9f82028261d | 29e54dc1-9344-4139-8ce0-81c7e2f01fa2 | 9a99b4c1-04e4-4d60-af3a-02230ee7c23e | 376d1533-a9c8-4a50-9036-8779df86da8d | 2025-10-14 16:4
1:55.570603 |                           | Active
 73e48d96-aec5-40da-82d0-57034569db80 | 29e54dc1-9344-4139-8ce0-81c7e2f01fa2 | 7e00dae2-d3ab-4498-a72c-74abbf959d36 | 376d1533-a9c8-4a50-9036-8779df86da8d | 2025-10-14 16:4
1:55.570603 |                           | Active
 41be254c-139a-4cb2-820e-d956c63290c2 | 29e54dc1-9344-4139-8ce0-81c7e2f01fa2 | 740cda15-30fe-4602-ac19-7e81b17deabc | 376d1533-a9c8-4a50-9036-8779df86da8d | 2025-10-14 16:4
1:55.570603 |                           | Active
 52e7ffdc-4294-4d10-9159-10eaf5fa000e | 29e54dc1-9344-4139-8ce0-81c7e2f01fa2 | 414c11fb-0822-403f-82cb-848f27bfe663 | 376d1533-a9c8-4a50-9036-8779df86da8d | 2025-10-14 16:4
1:55.570603 | 2025-10-14 17:16:36.025014 | Closed
 24467741-1259-4ff6-be19-c34af9a8b416 | 29e54dc1-9344-4139-8ce0-81c7e2f01fa2 | 9544609a-a005-4301-b681-d0d715295b82 | 376d1533-a9c8-4a50-9036-8779df86da8d | 2025-10-14 16:4
1:55.570603 | 2025-10-14 17:20:17.029283 | Closed
(5 rows)
```

parkingticketingsystem=# select * from Payment; (This field must be updated at exit time)

```
parkingticketingsystem=# select * from Payment;
              paymentid              |              ticketid              |  amount  |        paymentdate         | method
-------------------------------------+-------------------------------------+----------+----------------------------+--------
 c127b5ef-6bdf-458c-b674-4af11164641d | 52e7ffdc-4294-4d10-9159-10eaf5fa000e |  5000.00 | 2025-10-14 17:16:53.047356 | Card
 645c25e4-3d81-421d-83e6-e5c2d2b99865 | 24467741-1259-4ff6-be19-c34af9a8b416 | 15000.00 | 2025-10-14 17:20:17.03766  | Cash
(2 rows)
```

2.4. Retrieve all occupied spaces with vehicle details

```
parkingticketingsystem=# SELECT s.SpaceNo, s.Type, v.PlateNo, v.OwnerName, t.EntryTime
                         FROM Space s
                         JOIN Ticket t ON s.SpaceID = t.SpaceID
                         JOIN Vehicle v ON t.VehicleID = v.VehicleID
                         WHERE s.Status = 'Occupied';
```

```
parkingticketingsystem=# SELECT s.SpaceNo, s.Type, v.PlateNo, v.OwnerName, t.EntryTime
parkingticketingsystem-#     FROM Space s
parkingticketingsystem-#     JOIN Ticket t ON s.SpaceID = t.SpaceID
parkingticketingsystem-#     JOIN Vehicle v ON t.VehicleID = v.VehicleID
parkingticketingsystem-#     WHERE s.Status = 'Occupied';
 spaceno | type | plateno | ownername | entrytime
---------+------+---------+-----------+-----------
(0 rows)
```

2.5. Update payment status upon vehicle exit

2.5.1. Update payment status upon vehicle exit

```
parkingticketingsystem=#  UPDATE Ticket
                          SET ExitTime = NOW(), Status = 'Closed'
                          WHERE TicketID = '<<insert-ticket-id-here>>';
parkingticketingsystem=#  INSERT INTO Payment (TicketID, PaymentDate ,Amount,Method)
                          VALUES ('<<insert-ticket-id-here>>', 5000, 'Card');
```

2.5.2. Example to update payment status upon vehicle exits

```
parkingticketingsystem=#  UPDATE Ticket SET ExitTime = NOW(),Status = 'Closed'
                    WHERE TicketID = '24467741-1259-4ff6-be19-c34af9a8b416';

parkingticketingsystem=#  INSERT INTO Payment (TicketID, Amount, PaymentDate,Method)
                    VALUES ('24467741-1259-4ff6-be19-c34af9a8b416', 15000, NOW(), 'Cash');
```

2. 6. Identify parking lots nearing full capacity (Example of occupancy rate above 80%)

parkingticketingsystem=# SELECT pl.Name, pl.Capacity,COUNT(s.SpaceID) AS OccupiedSpaces,ROUND((COUNT(s.SpaceID)::DECIMAL / pl.Capacity) * 100, 2) AS OccupancyRate
FROM ParkingLot pl
JOIN Space s ON pl.LotID = s.LotID
WHERE s.Status = 'Occupied'
GROUP BY pl.LotID
HAVING (COUNT(s.SpaceID)::DECIMAL / pl.Capacity) * 100 >= 80;

```
parkingticketingsystem=#  SELECT pl.Name, pl.Capacity,COUNT(s.SpaceID) AS OccupiedSpaces,ROUND((COUNT(s.SpaceID)::DECIMAL / pl.Capacity) * 100, 2) AS OccupancyRate
parkingticketingsystem-# FROM ParkingLot pl
parkingticketingsystem-# JOIN Space s ON pl.LotID = s.LotID
parkingticketingsystem-# WHERE s.Status = 'Occupied'
parkingticketingsystem-# GROUP BY pl.LotID
parkingticketingsystem-# HAVING (COUNT(s.SpaceID)::DECIMAL / pl.Capacity) * 100 >= 80;
 name | capacity | occupiedspaces | occupancyrate
------+----------+----------------+---------------
(0 rows)
```

2.7. Create a view showing total revenue per lot

2.7.1. Create a view showing total revenue per lot

parkingticketingsystem=# CREATE OR REPLACE VIEW LotRevenue AS SELECT pl.Name AS LotName,SUM(p.Amount) AS TotalRevenue
FROM Payment p
JOIN Ticket t ON p.TicketID = t.TicketID
JOIN Space s ON t.SpaceID = s.SpaceID
JOIN ParkingLot pl ON s.LotID = pl.LotID
GROUP BY pl.Name;

```
parkingticketingsystem=# CREATE OR REPLACE VIEW LotRevenue AS SELECT pl.Name AS LotName,SUM(p.Amount) AS TotalRevenue
parkingticketingsystem-# FROM Payment p
parkingticketingsystem-# JOIN Ticket t ON p.TicketID = t.TicketID
parkingticketingsystem-# JOIN Space s ON t.SpaceID = s.SpaceID
parkingticketingsystem-# JOIN ParkingLot pl ON s.LotID = pl.LotID
parkingticketingsystem-# GROUP BY pl.Name;
CREATE VIEW
parkingticketingsystem=#
```

2.7.2. To look at the created view

parkingticketingsystem=# SELECT * FROM LotRevenue;

```
parkingticketingsystem=# SELECT * FROM LotRevenue;
 lotname | totalrevenue
---------+--------------
 Lot A   |     20000.00
(1 row)
```

2.8. Trigger to mark space as available after payment completion

2.8.1. Function

```
CREATE OR REPLACE FUNCTION update_space_status()
RETURNS TRIGGER AS $$
BEGIN
   UPDATE Space
   SET Status = 'Available'
   WHERE SpaceID = (SELECT SpaceID FROM Ticket WHERE TicketID = NEW.TicketID);
   RETURN NEW;
END;
$$ LANGUAGE plpgsql;
```

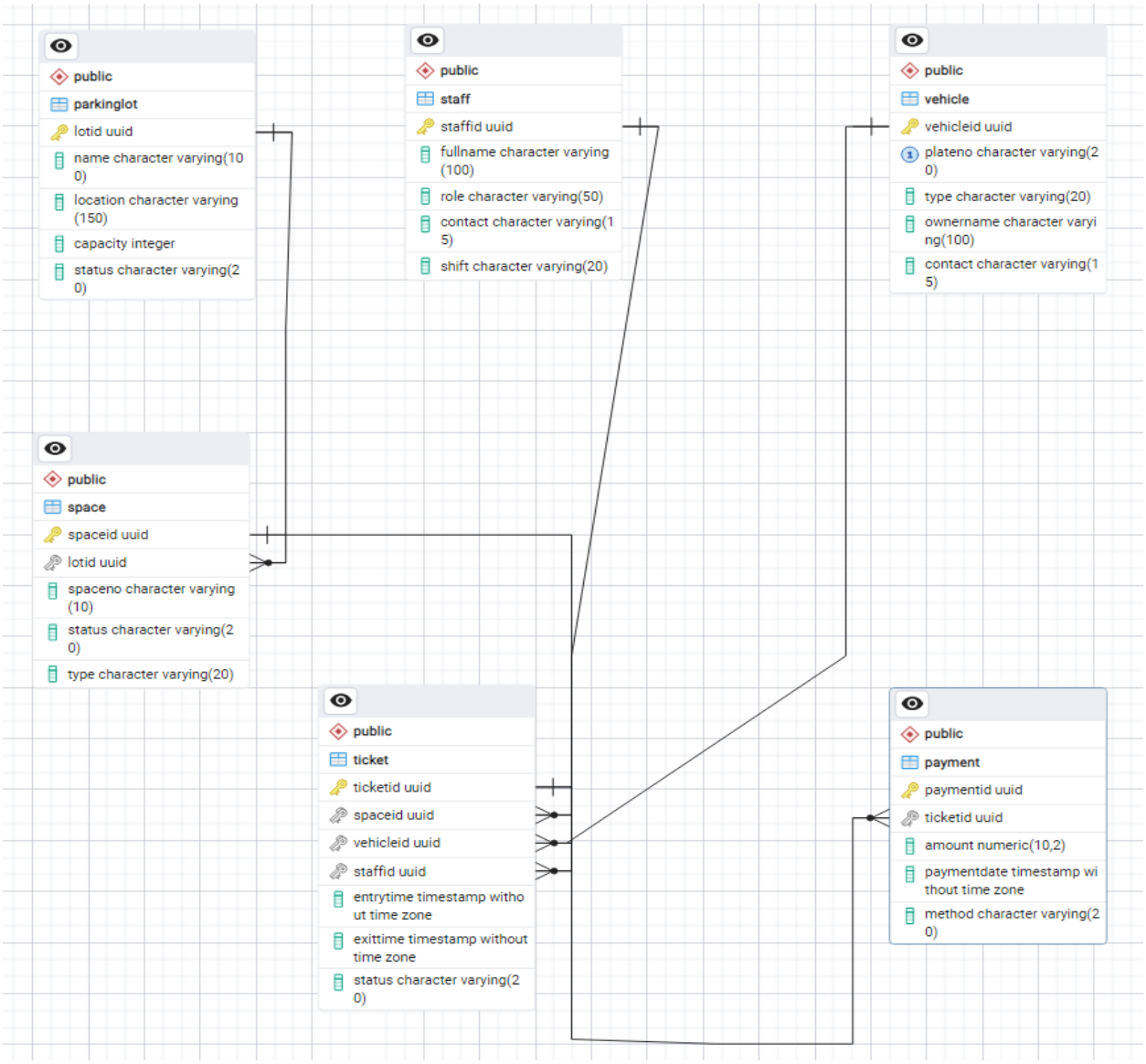
2.8.2. Trigger

```
CREATE TRIGGER trg_update_space_after_payment
AFTER INSERT ON Payment
FOR EACH ROW
EXECUTE FUNCTION update_space_status();
```

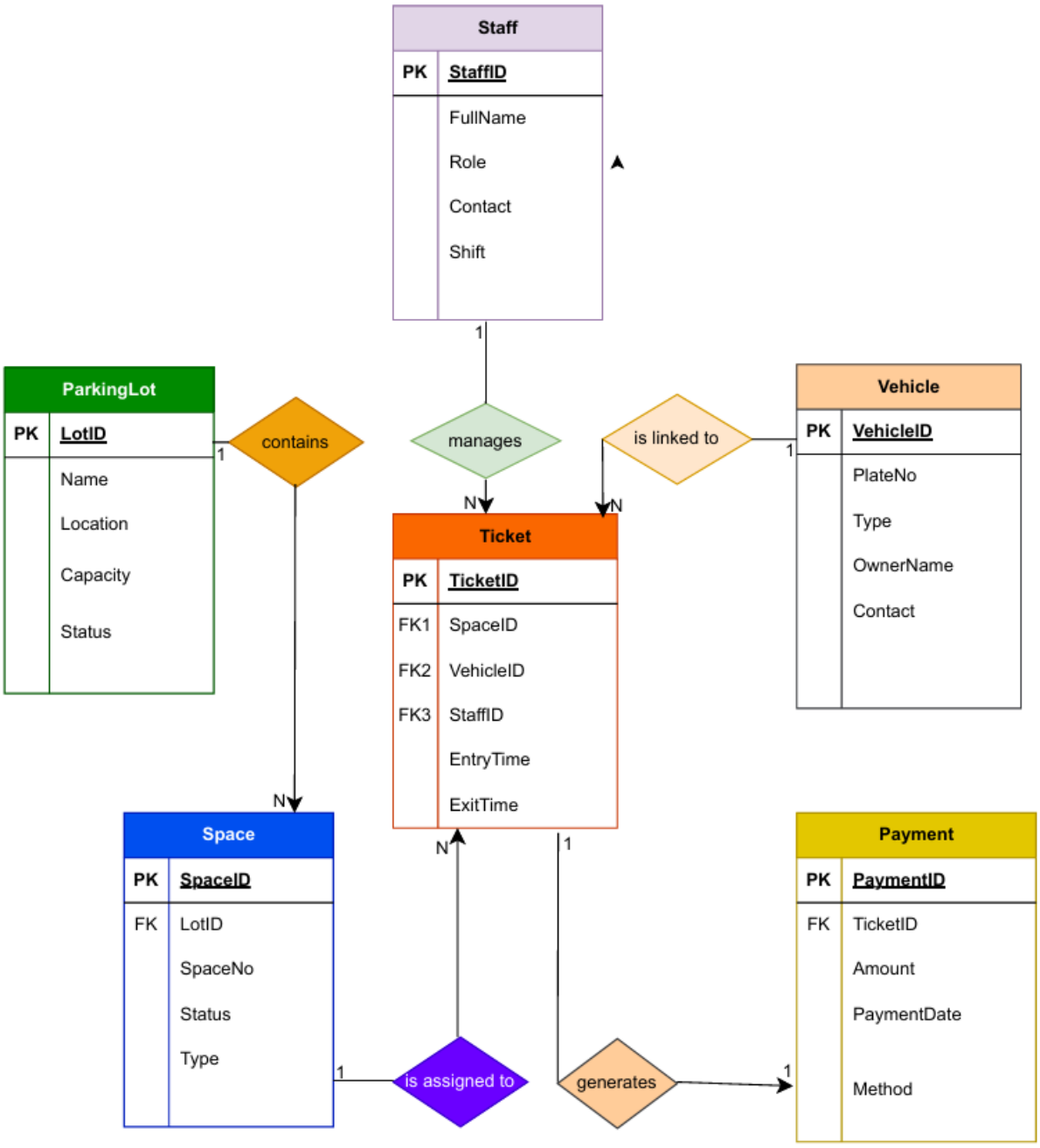
```
parkingticketingsystem=# CREATE OR REPLACE FUNCTION update_space_status()
parkingticketingsystem-# RETURNS TRIGGER AS $$
parkingticketingsystem$# BEGIN
parkingticketingsystem$#     UPDATE Space
parkingticketingsystem$#     SET Status = 'Available'
parkingticketingsystem$#     WHERE SpaceID = (SELECT SpaceID FROM Ticket WHERE TicketID = NEW.TicketID);
parkingticketingsystem$#     RETURN NEW;
parkingticketingsystem$# END;
parkingticketingsystem$# $$ LANGUAGE plpgsql;
CREATE FUNCTION
parkingticketingsystem=#
parkingticketingsystem=# -- Trigger
parkingticketingsystem=# CREATE TRIGGER trg_update_space_after_payment
parkingticketingsystem-# AFTER INSERT ON Payment
parkingticketingsystem-# FOR EACH ROW
parkingticketingsystem-# EXECUTE FUNCTION update_space_status();
ERROR:  trigger "trg_update_space_after_payment" for relation "payment" already exists
```

What you want to see	Required commands
View function code	SELECT pg_get_functiondef('update_space_status'::regproc);
List function (short)	\df+ update_space_status
Show trigger (quick)	\d payment
Show trigger (SQL way)	SELECT * FROM information_schema.triggers WHERE trigger_name = 'trg_update_space_after_payment';

2.9. **Entity relationship diagram from PostgreSQL pgAdmin**



2.10. Diagram of relationships of entities from draw.io



3. Conclusion

The smart parking management and ticketing system, developed under the Advanced Database Technology module using PostgreSQL, successfully demonstrates the application of advanced database concepts such as relational design, referential integrity, automation, and data consistency. The project efficiently models real-world parking operations by integrating key entities such as ParkingLot, Space, Vehicle, Ticket, Staff, and Payment while enforcing strong relationships through primary and foreign key constraints. Features like cascade delete, triggers for automated space updates, and views for revenue reporting enhance the system's intelligence and operational efficiency. Overall, this implementation highlights how database technologies can optimize urban parking management, improve revenue tracking, and support decision-making through reliable, automated data handling.