<u>ASSIGNMENT 2 – SCRAPPING TWITTER</u>

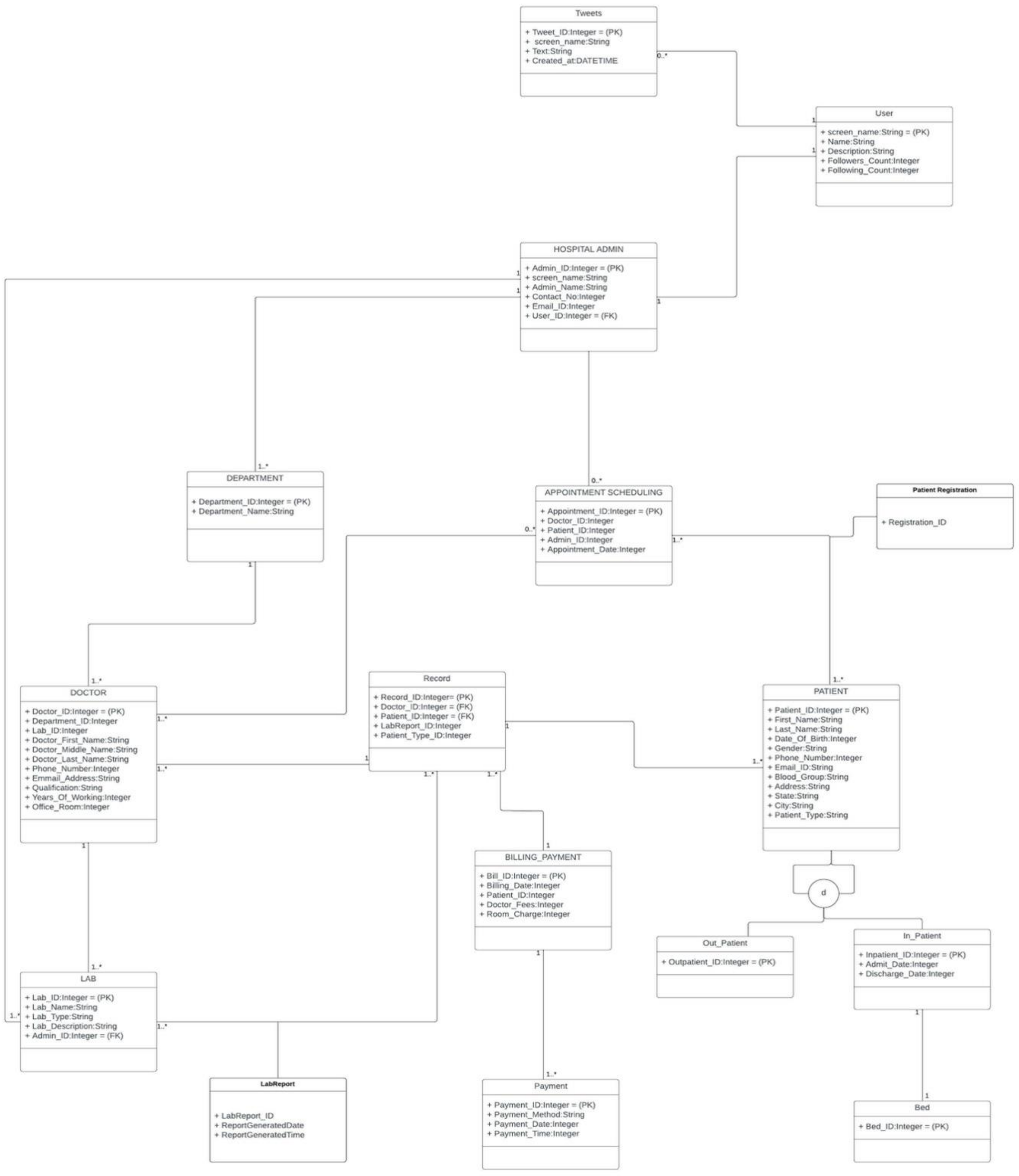
<u>Members:</u>

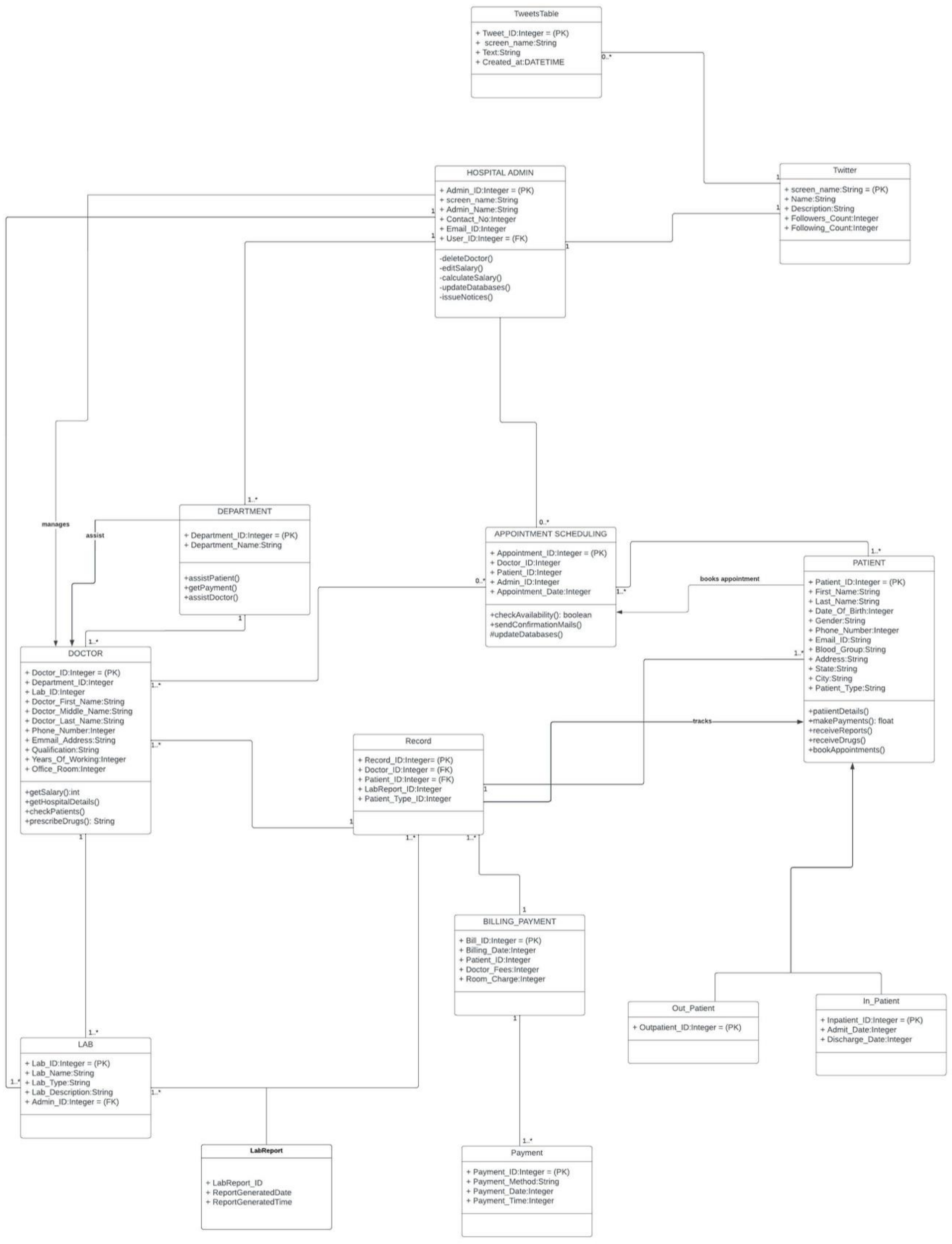
1. Yash Revadekar - 002738776
2. Ikra Bagwan - 002794307
3. Krutik Kanakia - 002787847

A Model on Hospital using Twitter :

The Hospital Management model includes Twitter database schema as well. In this setup, the administrator has a Twitter account for the hospital and can tweet hospital-related content. The user has a personal Twitter account and can tweet on hospital-related topics.

Find below the ER and UML diagram of the Hospital Twitter domain.


Tweets
+ Tweet_ID:Integer = (PK)
+ screen_name:String
+ Text:String
+ Created_at:DATETIME
User
+ screen_name:String = (PK)
+ Name:String
+ Description:String
+ Followers_Count:Integer
+ Following_Count:Integer
HOSPITAL ADMIN
+ Admin_ID:Integer = (PK)
+ screen_name:String
+ Admin_Name:String
+ Contact_No:Integer
+ Email_ID:Integer
+ User_ID:Integer = (FK)
DEPARTMENT
+ Department_ID:Integer = (PK)
+ Department_Name:String
APPOINTMENT SCHEDULING
+ Appointment_ID:Integer = (PK)
+ Doctor_ID:Integer
+ Patient_ID:Integer
+ Admin_ID:Integer
+ Appointment_Date:Integer
Patient Registration
+ Registration_ID
DOCTOR
+ Doctor_ID:Integer = (PK)
+ Department_ID:Integer
+ Lab_ID:Integer
+ Doctor_First_Name:String
+ Doctor_Middle_Name:String
+ Doctor_Last_Name:String
+ Phone_Number:Integer
+ Emmail_Address:String
+ Qualification:String
+ Years_Of_Working:Integer
+ Office_Room:Integer
Record
+ Record_ID:Integer= (PK)
+ Doctor_ID:Integer = (FK)
+ Patient_ID:Integer = (FK)
+ LabReport_ID:Integer
+ Patient_Type_ID:Integer
PATIENT
+ Patient_ID:Integer = (PK)
+ First_Name:String
+ Last_Name:String
+ Date_Of_Birth:Integer
+ Gender:String
+ Phone_Number:Integer
+ Email_ID:String
+ Blood_Group:String
+ Address:String
+ State:String
+ City:String
+ Patient_Type:String
BILLING_PAYMENT
+ Bill_ID:Integer = (PK)
+ Billing_Date:Integer
+ Patient_ID:Integer
+ Doctor_Fees:Integer
+ Room_Charge:Integer
d
Out_Patient
+ Outpatient_ID:Integer = (PK)
In_Patient
+ Inpatient_ID:Integer = (PK)
+ Admit_Date:Integer
+ Discharge_Date:Integer
LAB
+ Lab_ID:Integer = (PK)
+ Lab_Name:String
+ Lab_Type:String
+ Lab_Description:String
+ Admin_ID:Integer = (FK)
LabReport
+ LabReport_ID
+ ReportGeneratedDate
+ ReportGeneratedTime
Payment
+ Payment_ID:Integer = (PK)
+ Payment_Method:String
+ Payment_Date:Integer
+ Payment_Time:Integer
Bed
+ Bed_ID:Integer = (PK)



TweetsTable
+ Tweet_ID:Integer = (PK)
+ screen_name:String
+ Text:String
+ Created_at:DATETIME
0..*
Twitter
+ screen_name:String = (PK)
+ Name:String
+ Description:String
+ Followers_Count:Integer
+ Following_Count:Integer
1
1
HOSPITAL ADMIN
+ Admin_ID:Integer = (PK)
+ screen_name:String
+ Admin_Name:String
+ Contact_No:Integer
+ Email_ID:Integer
+ User_ID:Integer = (FK)
-deleteDoctor()
-editSalary()
-calculateSalary()
-updateDatabases()
-issueNotices()
1
1
1
manages
assist
DEPARTMENT
+ Department_ID:Integer = (PK)
+ Department_Name:String
+assistPatient()
+getPayment()
+assistDoctor()
1..*
1
APPOINTMENT SCHEDULING
+ Appointment_ID:Integer = (PK)
+ Doctor_ID:Integer
+ Patient_ID:Integer
+ Admin_ID:Integer
+ Appointment_Date:Integer
+checkAvailability(): boolean
+sendConfirmationMails()
#updateDatabases()
0..*
0..*
1..*
books appointment
PATIENT
+ Patient_ID:Integer = (PK)
+ First_Name:String
+ Last_Name:String
+ Date_Of_Birth:Integer
+ Gender:String
+ Phone_Number:Integer
+ Email_ID:String
+ Blood_Group:String
+ Address:String
+ State:String
+ City:String
+ Patient_Type:String
+patiientDetails()
+makePayments(): float
+receiveReports()
+receiveDrugs()
+bookAppointments()
1..*
1..*
tracks
DOCTOR
+ Doctor_ID:Integer = (PK)
+ Department_ID:Integer
+ Lab_ID:Integer
+ Doctor_First_Name:String
+ Doctor_Middle_Name:String
+ Doctor_Last_Name:String
+ Phone_Number:Integer
+ Emmail_Address:String
+ Qualification:String
+ Years_Of_Working:Integer
+ Office_Room:Integer
+getSalary():int
+getHospitalDetails()
+checkPatients()
+prescribeDrugs(): String
1..*
1..*
1..*
1
Record
+ Record_ID:Integer= (PK)
+ Doctor_ID:Integer = (FK)
+ Patient_ID:Integer = (FK)
+ LabReport_ID:Integer
+ Patient_Type_ID:Integer
1
1
1..*
1..*
BILLING_PAYMENT
+ Bill_ID:Integer = (PK)
+ Billing_Date:Integer
+ Patient_ID:Integer
+ Doctor_Fees:Integer
+ Room_Charge:Integer
1
1
Out_Patient
+ Outpatient_ID:Integer = (PK)
In_Patient
+ Inpatient_ID:Integer = (PK)
+ Admit_Date:Integer
+ Discharge_Date:Integer
LAB
+ Lab_ID:Integer = (PK)
+ Lab_Name:String
+ Lab_Type:String
+ Lab_Description:String
+ Admin_ID:Integer = (FK)
1..*
1..*
1..*
LabReport
+ LabReport_ID
+ ReportGeneratedDate
+ ReportGeneratedTime
Payment
+ Payment_ID:Integer = (PK)
+ Payment_Method:String
+ Payment_Date:Integer
+ Payment_Time:Integer
1..*

Explanation on some of the design decisions:

- The model consists of the User and Tweets entities.
- Each user may post an any amount of tweets. The hospital administrator who tweets health-related information; this information can be stored within the user table.
- A user can tweet and provide reviews about a hospital, as well as provide information regarding medical crises.
- A tweet is a table where user and hospital administration tweets are stored.

SQL Statements and Constraints for the conceptual model:

Tweets Table:

```
CREATE TABLE  ‘Tweets’
(
        ‘tweet_id’ INT NOT NULL,
        ‘screen_name’ INT,
        ‘tweet_text’  INT,
        ‘date_time’   INT,
        CONSTRAINT Tweets_PK  PRIMARY KEY ("tweet_id")
);
```

User Table:

```
CREATE TABLE ‘User_Table’
(
        ‘Screen_Name’ VARCHAR (30),
        ‘Name’        VARCHAR (30),
        ‘Description’         TEXT,
        ‘Follower’    INT,
        ‘Following’  INT,
        CONSTRAINT UserTable_PK PRIMARY KEY("Screen_Name")
);
```

Hospital Admin Table:

```
CREATE TABLE ‘HOSPITAL_ADMIN’
(
  ‘ADMIN_ID’ INT NOT NULL IDENTITY(100,1),
   ‘ADMIN_NAME’ VARCHAR(30) NOT NULL,
   ‘CONTACT_NO’ BIGINT NOT NULL,
   ‘EMAIL_ID’ VARCHAR(50) NOT NULL UNIQUE,
        CONSTRAINT HOSPITAL_ADMIN_PK PRIMARY KEY
(ADMIN_ID)
);
```

Patient Table:

```
CREATE TABLE ‘PATIENT’
(
   ‘PATIENT_ID’ INT IDENTITY(5000,1) CONSTRAINT PATIENT_PK
PRIMARY KEY, -- primary key column
   ‘FIRST_NAME’ VARCHAR(30) NOT NULL,
   ‘MIDDLE_NAME’ VARCHAR(20),
   ‘LAST_NAME’ VARCHAR(30) NOT NULL,
   ‘DOB’ DATE NOT NULL ,
   ‘WEIGHT’ INT CHECK(WEIGHT > 0), -- in pounds (lbs)
   ‘HEIGHT’ INT CHECK(HEIGHT > 0), -- in centimetres (cm)
   ‘GENDER’ VARCHAR(2) NOT NULL CHECK(gender IN ('M', 'F', 'NA')),
-- assigned at birth
   ‘STREET_NO’ INT NOT NULL,
   ‘STREET_NAME’ VARCHAR(100) NOT NULL,
   ‘CITY’ VARCHAR(30) NOT NULL,
   ‘STATE_NAME’ CHAR(2) NOT NULL, -- Two letter abbreviation for
stateName
   ‘ZIP’ INT NOT NULL,
   ‘PHONE_NO’ BIGINT  NOT NULL,
   ‘EMAIL_ID’ VARCHAR(50) NULL,
);
```

Department Table:

```
CREATE TABLE ‘DEPARTMENT’
(
   ‘DEPT_ID’ INT NOT NULL IDENTITY(3000,1),
   ‘DEPT_NAME’ VARCHAR(100) NOT NULL,
   ‘ADMIN_ID’ INT NOT NULL,
   CONSTRAINT DEPARTMENT_PK PRIMARY KEY(DEPT_ID),
   CONSTRAINT DEPARTMENT_FK FOREIGN KEY (ADMIN_ID)
REFERENCES HOSPITAL_ADMIN(ADMIN_ID)
);
```

Doctor Table:

```
CREATE TABLE ‘DOCTOR’
(
   ‘DOCTOR_ID’ INT NOT NULL IDENTITY(4000,1),
   ‘DEPT_ID’ INT NOT NULL,
   ‘FIRST_NAME’ VARCHAR(30) NOT NULL,
   ‘MIDDLE_NAME’ VARCHAR(30),
   ‘LAST_NAME’ VARCHAR(30) NOT NULL,
   ‘PHONE_NO’ VARCHAR(10) NOT NULL,
   ‘EMAIL_ID’ VARCHAR(50) UNIQUE NOT NULL,
   ‘QUALIFICATION’ VARCHAR(30) NOT NULL,
   ‘YEARS_OF_WORKING’ INT NOT NULL,
   ‘OFFICE_ROOM’ VARCHAR(4) NOT NULL,
   CONSTRAINT DOCTOR_PK PRIMARY KEY(DOCTOR_ID),
   CONSTRAINT DOCTOR_FK FOREIGN KEY (DEPT_ID) REFERENCES
DEPARTMENT(DEPT_ID)
);
```

Appointment Scheduling:

```
Create table ‘APPOINTMENT_SCHEDULING’
(
        ‘APPOINTMENT_ID’ int not null identity(1000,1),
        ‘DOCSCHEDULE_ID’ int not null ,
        ‘PATIENT_ID’ int not null ,
        ‘ADMIN_ID’ int not null ,
        ‘APPOINTMENT_DATE’ datetime not null,
        ‘START_TIME TIME’ NOT NULL,
        ‘END_TIME TIME’ NOT NULL,
        ‘APPOINTMENT_STATUS’ varchar(30),
        ‘APPOINTMENT_REASON’ VARCHAR(50),
        ‘PATIENT_TYPE’ VARCHAR(1)
        Constraint Appointment_Scheduling_PK PRIMARY KEY
(APPOINTMENT_ID),
        Constraint Appointment_Scheduling_FK1 FOREIGN KEY
(DOCSCHEDULE_ID) REFERENCES
Doctor_Schedule(DOCSCHEDULE_ID),
        Constraint Appointment_Scheduling_FK2 FOREIGN KEY
(PATIENT_ID) REFERENCES Patient (PATIENT_ID),
        Constraint Appointment_Scheduling_FK3 FOREIGN KEY
(ADMIN_ID) REFERENCES Hospital_Admin (ADMIN_ID)
);
```

Lab Table:

```
CREATE TABLE ‘LAB’
(
   ‘LAB_ID’ INT NOT NULL IDENTITY(8000,1),
   ‘LAB_NAME’ VARCHAR(30) NOT NULL,
        ‘LAB_TYPE’ VARCHAR(30) NOT NULL,
   ‘LAB_DESCRIPTION’ VARCHAR(50) NOT NULL,
        ‘ADMIN_ID’ INT NOT NULL,
   CONSTRAINT LAB_PK PRIMARY KEY (LAB_ID),
        CONSTRAINT LAB_FK FOREIGN KEY (ADMIN_ID) REFERENCES
HOSPITAL_ADMIN(ADMIN_ID)
);
```

Billing Table:

```
Create table ‘BILLING’
(
        ‘BILLING_ID’ int not null identity (9000,1),
        ‘BILLING_DATE’ date not null,
        ‘PATIENT_ID’ int not null ,
        ‘DOCTOR_FEES’ int not null,
        ‘ROOM_CHARGES’ int not null,
        Constraint Billing_PK PRIMARY KEY (BILLING_ID),
        Constraint Billing_FK FOREIGN KEY (PATIENT_ID) REFERENCES
Patient (PATIENT_ID)
);
```

Record Table:

```
CREATE TABLE ‘RECORD’
(
   ‘RECORD_ID’ INT PRIMARY KEY NOT NULL IDENTITY(10001,1),
   ‘DOCTOR_ID’ INT FOREIGN KEY (DOCTOR_ID) REFERENCES
DOCTOR(DOCTOR_ID),
   ‘PATIENT_ID’ INT FOREIGN KEY (PATIENT_ID) REFERENCES
PATIENT(PATIENT_ID),
   ‘ADMIT_DATE’ DATE,
   ‘DISCHARGEDATE’ DATE,
   ‘BILLING_ID’ INT NULL FOREIGN KEY (BILLING_ID) REFERENCES
BILLING(BILLING_ID),
   ‘PATIENT_TYPE’ VARCHAR(1) CONSTRAINT CHK_SUBJECT
CHECK (PATIENT_TYPE IN ('I', 'O')),
);
```

Lab Report Table:

```
CREATE TABLE ‘LAB_REPORT’
(
‘LABREPORT_ID’ INT NOT NULL IDENTITY(12001,1),
‘LAB_ID’ INT NULL,
‘RECORD_ID’ INT NOT NULL,
‘RPTGENERATED_DTTM’ DATETIME NOT NULL
CONSTRAINT LABREPORT_PK PRIMARY KEY (“LABREPORT_ID”),
CONSTRAINT LAB_REPORT_FK1 FOREIGN KEY (“LAB_ID”)
REFERENCES LAB(LAB_ID),
CONSTRAINT LAB_REPORT_FK2 FOREIGN KEY (RECORD_ID)
REFERENCES RECORD(RECORD_ID)
);
```

Payment Table:

```
CREATE TABLE ‘PAYMENT’
(
   ‘PAYMENT_ID’ INT NOT NULL IDENTITY (11001,1),
   ‘PAYMENT_METHOD’ VARCHAR(30) NOT NULL,
   ‘PAYMENT_DATE_TIME’ DATETIME NOT NULL,
   ‘BILLING_ID’ INT NOT NULL,
   CONSTRAINT PAYMENT_PK PRIMARY KEY (PAYMENT_ID),
   CONSTRAINT PAYMENT_FK FOREIGN KEY (BILLING_ID)
REFERENCES BILLING (BILLING_ID)
);
```

USE-CASE

1. **Use Case:** View the follower and tweet id

Description: Admin views the follower and tweet id

Actor: Admin

Precondition: There must be an twitter account

Steps:

Actor action: Admin views follower and tweet id from users

System Responses: Number of followers and tweet id would be displayed

Post Condition: System displays the whole follower and tweet id.

2. **Use Case:** View the total number of tweets by a particular user

Description: Admin views the total number of tweets by a user

Actor: Admin

Precondition: User must have a twitter account

Steps:

Actor action: Admin checks total number of tweets

System Responses: Displays the count of tweet

3. **Use Case:** View the patients count

Description: View patient count as per department

Actor: Admin

Precondition: There must be an department

Steps:

Actor action: Admin views patients from department

System Responses: Number of patients would be displayed

Post Condition: System displays patients count

4. **Use case:** Average Days Spent by patients in Hospital per Department

   Description: View Days of patients in Hospital

   Actor: Admin and Hospital

   Precondition: Patient must be present in the system

   Steps:

   Actor action: Admin views the days spent by patient

   System Responses: Days of the patient in hospital

   Post Condition: System displays average days of patients

5. **Use case:** Revenue per Department per month

   Description: Admin views total revenue of department

   Actor: Admin

   Precondition: Amount must be present

   Steps:

   Actor action: Admin views the revenue generated

   System Responses: Revenue generated

   Post Condition: System displays revenue of department

RELATIONAL-ALGEBRA EXPRESSIONS FOR THE USE CASES

1. **Use Case:** View the follower and tweet id :

π user . followers, tweets . id (user ⋈ user . screen_name = tweets . screen_name tweets)

2. **Use Case:** View the total number of tweets by a particular user :
   π COUNT (text)
    γ COUNT (text)
   σ screen_name = "LGCW2022" tweets

3. Use case: View the patients count
   π dt . dept_id, dt . dept_name, COUNT (patient_id) → total_patients
    γ dept_id, dept_name, COUNT (patient_id)
     (ρ a appointment_scheduling ⋈ a . patient_id = p . patient_id
      ρ p patient ⋈ ds . docschedule_id = a . docschedule_id
       ρ ds doctor_schedule ⋈ ds . doctor_id = dr . doctor_id
        ρ dr doctor ⋈ dr . dept_id = dt . dept_id
         ρ dt department)

SQL STATEMENTS

1. **Use Case:** View the follower and tweet id :

```
select user.followers,tweets.id
FROM user
INNER JOIN tweets ON user.screen_name=tweets.screen_name
```

2. **Use Case:** View the total number of tweets by a particular user :

```
Select COUNT(text) From tweets where screen_name ="LGCW2022
```

3. Use Case: View the patients count

```
        SELECT DT.DEPT_ID,DT.DEPT_NAME,
COUNT(P.PATIENT_ID) AS TOTAL_PATIENTS
      FROM APPOINTMENT_SCHEDULING A JOIN PATIENT P
       ON A.PATIENT_ID = P.PATIENT_ID JOIN DOCTOR_SCHEDULE
DS
       ON DS.DOCSCHEDULE_ID = A.DOCSCHEDULE_ID JOIN
DOCTOR DR
       ON DS.DOCTOR_ID = DR.DOCTOR_ID JOIN DEPARTMENT DT
       ON DR.DEPT_ID = DT.DEPT_ID
       GROUP BY DT.DEPT_ID, DT.DEPT_NAME;
```

4. Use Case: Average Days Spent by patients in Hospital per Department

```
Select
department_name,
Avg(days_in_hospital) as average_days_in_hospital

From

(Select
 d.department_id,
 de.department_name,
 p.patient_id
Discharge_date - Admit_date as days_in_hospital
From patients p
Left join record r on p.patient_id = r.patient_id
Left join doctor d on d.doctor_id = r.doctor_id
Left join department de on de.department_id = d.department_id
Left join in_patient ip on ip.in_patient_id = p.patient_id
Where discharge_date is not null
)
```

5. Use Case: Revenue per Department per month

```
  Select
department_name,
Date_trunc('month' , payment_date) as payment_month,
Sum(amount) as payment_amount
From patients p
Left join record r on p.patient_id = r.patient_id
Left join doctor d on d.doctor_id = r.doctor_id
Left join department de on de.department_id = d.department_id
Left join billing_payment bp on bp.patient_id = r.patient_id
Left join payment p on p.bill_id = bp.bill_id
Group by 1,2
```

Queries you must answer about your physical model

1. What time the user posted this tweet?

   SQL : Select created_at from tweets where screen_name="LGCW2022"

   Relational algebra : π created_at
   σ screen_name = "LGCW2022" tweets

2. What Tweet the User Posted?

   SQL : Select text from tweets where screen_name="EuropeanCancer"

   Relational algebra : π text
   σ screen_name = "EuropeanCancer" tweets

3. Count the number of tweets posted by user?

   SQL : Select COUNT(text) From tweets where screen_name ="LGCW2022"

   Relational algebra : π COUNT (text)
   γ COUNT (text)
   σ screen_name = "LGCW2022" tweets

4. Display entire table by not selecting a user?

   SQL : Select * from tweets Where NOT screen_name="SusannahStanwa1"

   Relational algebra : σ NOT (screen_name = "SusannahStanwa1") tweets

5. What are the number of followers of a user?

   SQL : Select followers from user where
   screen_name="SewantiLimaye"

   Relational algebra : π followers
   σ screen_name = "SewantiLimaye" user

6. What are the number of following user?

   SQL : Select following from user where
   screen_name="ThatPhysioAbu"

   Relational algebra : π following
   σ screen_name = "ThatPhysioAbu" user

7. How to Join followers from user and id from tweets?

   SQL : select user.followers,tweets.id FROM user INNER JOIN
   tweets ON user.screen_name=tweets.screen_name

   Relational algebra : π user . followers, tweets . id (user ⋈ user .
   screen_name = tweets . screen_name tweets)

8. How to Display Distinct screen_name?

   SQL: Select DISTINCT screen_name FROM user

   Relational algebra :
   δ
   π screen_name user