SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

DBMS PROJECT-1

HOSPITAL MANAGEMENT SYSTEM

SUBMITTED TO

Ms.D.Hemavathi

Date:

Project Team:

ANKIT PAUL (RA2111042010060)

PRANAY BARANWAL (RA2111042010040)

KARTHIK MSV (RA2111042010058)

SRM INSTITUTE OF SCIENCE & TECHNOLOGY
COLLEGE OF ENGINEERING & TECHNOLOGY
S.R.M. NAGAR, KATTANKULATHUR – 603 203

BONAFIDE CERTIFICATE

Certified that this project report **Hospital Management System** is the bonafide work of **Ankit Paul, Pranay Baranwal, Karthik RA2111042010060, RA2111042010040, RA2111042010058** of II Year/IV Sem B.Tech(CSBS) who carried out the mini project work under my supervision for the course 18CSC303J- Database Management systems in SRM Institute of Science and Technology during the academic year 2022-2023(Even Sem).

SIGNATURE

Dr. Hemavathi D
Associate Professor
Data Science and Business systems

SIGNATURE

Dr. Lakshmi M
Head Of Department
Data Science and Business Systems

Abstract

The Hospital Management System (HMS) is a comprehensive and integrated database management system developed to help hospitals manage patient information, medical records, and hospital resources. The system is based on a relational database management system (RDBMS), which effectively stores and retrieves data, allowing for quick access to accurate and up-to-date information. Modules for patient registration, appointment scheduling, medical record administration, prescription management, billing and payment, inventory management, and staff management are included in the system. The Hospital Management System is intended to improve patient care quality, streamline administrative procedures, and increase the efficiency of hospital operations. The system can be tailored to match the specific requirements of individual hospitals, and it provides a dependable and secure platform for managing essential healthcare data.

Introduction

Welcome to the introduction of our Hospital Management Project. The healthcare industry is a critical sector that touches the lives of individuals and communities across the globe. As such, it is essential to have efficient and effective hospital management systems in place to ensure that patients receive the best possible care while maintaining operational efficiency.

Our Hospital Management Project is designed to provide an integrated, user-friendly, and comprehensive solution to healthcare providers, administrators, and patients. The system incorporates various functionalities, including patient management, appointment scheduling, inventory management, billing and payment processing, and reporting, among others.

By using our Hospital Management Project, healthcare providers can streamline their operations, improve patient care, and increase revenue. Patients, on the other hand, can easily access medical services, view their medical records, and make appointments from the comfort of their homes.

Our project has been developed with the latest technology and is highly customizable, making it suitable for hospitals, clinics, and other healthcare facilities of all sizes.

Proposed Work Details:

The proposed work is made up of seven interrelated tables. Team members work on tables and maintain them up to date by developing queries.

Each table's structure and function are discussed below:

Admit Table:

It contains all the information regarding the dates when the patient was admitted in the hospital with their respective admit id's.

- AD_ID Varchar2(25)
- AD_DATE DATE

Table Structure:

```
SQL> desc admit;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 AD_ID                                               VARCHAR2(25)
 AD_DATE                                             DATE

SQL> select * from admit;
```

Appointment:

This table describes about the dates when the appointments were taken by the patient and their respective appointment id's.

- A_ID NOT NULL Number(38)
- A_DATE DATE

Table Structure:

```
SQL> desc appointment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 A_ID                                      NOT NULL NUMBER(38)
 A_DATE                                             DATE
```

Department:

This is a brief description about the different department the hospital operates in and their respective id's for each department.

- DEPT_ID NOT NULL Number(38)
- DEPT_NAME NOT NULL Varchar2(25)

Table Structure:

```
SQL> desc department;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------
---------
 DEPT_ID                                   NOT NULL NUMBER(38)
 DEPT_NAME                                 NOT NULL VARCHAR2(25)
```

Doctor's Contact Details:

We have made a record for the contacts details for each department which can be acknowledged through doctor's id.

- DOC_ID Number(38)
- DOC_PH Number(38)

Table Structure:

```
SQL> desc doc_ph;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 DOC_ID                                             NUMBER(38)
 DOC_PH                                             NUMBER(38)

SQL>
```

Doctor:

This table describes all the details needed about a doctor like personal and contact information.

- DOC_ID NOT NULL Number(38)
- DOC_NAME NOT NULL Varchar2(25)
- DOC_EMAIL Varchar2(25)
- DOC_GENDER NOT NULL Varchar2(10)
- DOC_ADPIN NOT NULL Number(38)
- DOC ADSTATE Varchar2(25)

Table Structure:

```
SQL> desc doctor;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------------
---------
 DOC_ID                                    NOT NULL NUMBER(38)
 DOC_NAME                                  NOT NULL VARCHAR2(25)
 DOC_EMAIL                                          VARCHAR2(25)
 DOC_GENDER                                NOT NULL VARCHAR2(10)
 DOC_ADPIN                                 NOT NULL NUMBER(38)
 DOC_ADSTATE                                        VARCHAR2(25)
```

Patient's Contact Details:

We have collectively gathered the contact information of the patients that are admitted into the hospital.

- P_ID Number(38)
- P_PH Number(38)

Table Structure:

```
SQL> desc pat_ph;
 Name                                      Null?    Type
 ----------------------------------------- -------- ------------
 P_ID                                               NUMBER(38)
 P_PH                                               NUMBER(38)
```

Patient Details:

This schema describes the details about the patient including personal information and contact details.

- P_ID NOT NULL Number(38)
- P_NAME NOT NULL Varchar2(25)
- P_DOB DATE
- P_GENDER Varchar2(10)
- P_EMAIL Varchar2(25)
- P_PIN Number(38)
- P_STATE Varchar2(25)

Table Structure:

```
SQL> desc patient;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 P_ID                                      NOT NULL NUMBER(38)
 P_NAME                                    NOT NULL VARCHAR2(25)
 P_DOB                                              DATE
 P_GENDER                                           VARCHAR2(10)
 P_EMAIL                                            VARCHAR2(25)
 P_PIN                                              NUMBER(38)
 P_STATE                                            VARCHAR2(25)
```

Payments:

All the transaction details including amount paid for treatment, payment id and details about the patient is given here.

- P_ID Number(38)
- PAY_ID Number(38)
- PAY AMT Number(38)

Table Structure:

```
SQL> desc payment;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------
 P_ID                                               NUMBER(38)
 PAY_ID                                             NUMBER(38)
 PAY_AMT                                            NUMBER(38)
```

Room Details:

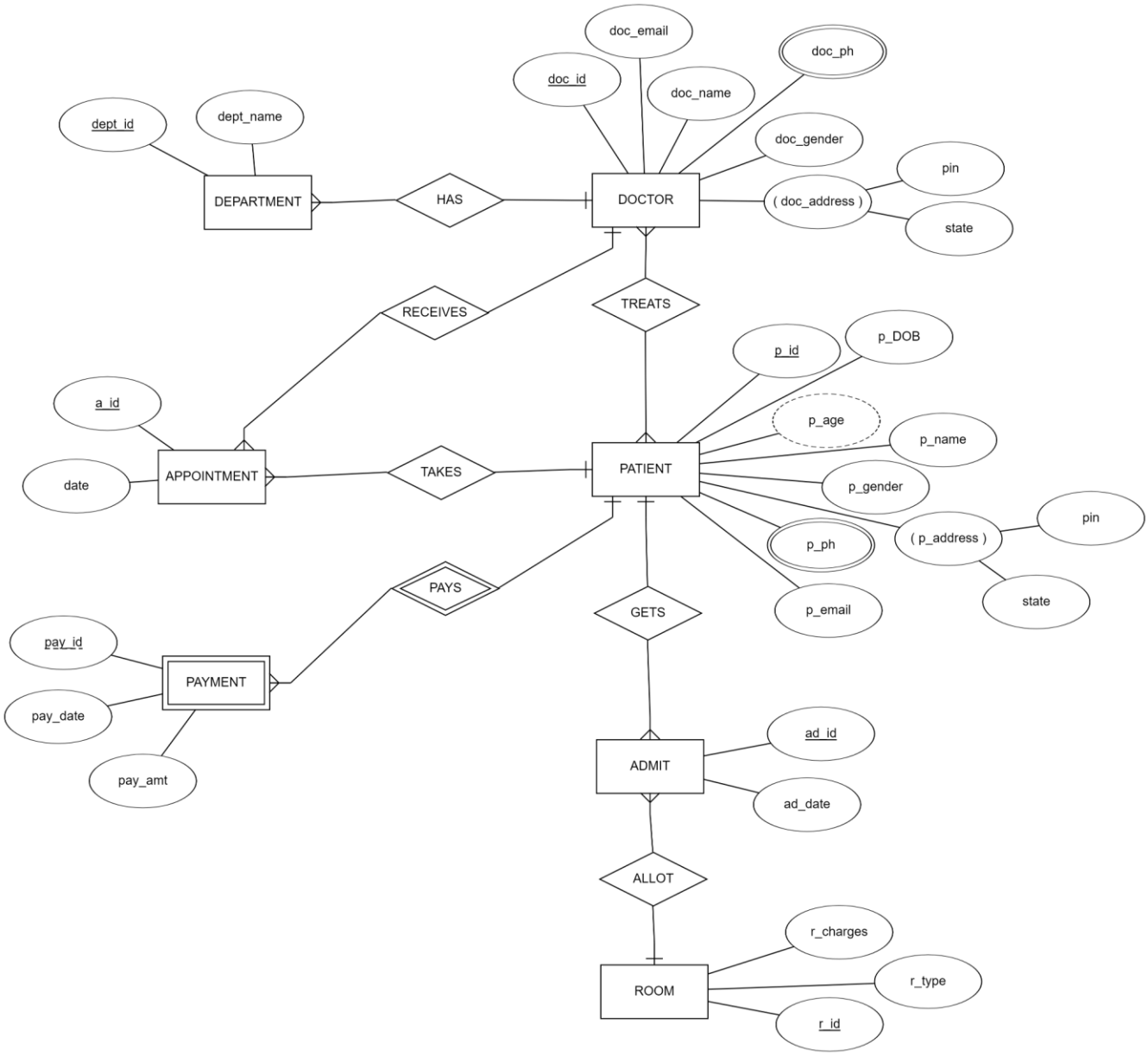
We have a description about the rooms that are allocated to different patients in the hospital.

- R_ID Varchar2(25)
- R_TYPE Varchar2(25)
- R CHARGES Number(38)

Table Structure:

```
SQL> desc room;
 Name                                      Null?    Type
 ----------------------------------------- -------- ----------------------------
 R_ID                                               VARCHAR2(25)
 R_TYPE                                             VARCHAR2(25)
 R_CHARGES                                          NUMBER(38)
```

ER Diagram


doc_email
doc_ph
doc_id
doc_name
dept_id
dept_name
doc_gender
pin
DEPARTMENT
HAS
DOCTOR
(doc_address)
state
RECEIVES
TREATS
p_id
p_DOB
a_id
p_age
p_name
date
APPOINTMENT
TAKES
PATIENT
p_gender
pin
p_ph
(p_address)
PAYS
GETS
state
p_email
pay_id
PAYMENT
pay_date
pay_amt
ad_id
ADMIT
ad_date
ALLOT
r_charges
ROOM
r_type
r_id

Relationship Tables:

- Has: This is a one-to-many relationship from the department to doctor to denote which doctor belongs to which department.
- Takes: This is a one-to-many relationship from patient to appointment which defines the number of appointments by a patient.
- Receive: This is a many-to-one relationship from appointment to doctor, This is simply one doctor can receive multiple appointments.
- Treats: This is a many to many relationships between the doctor and patient signifying many doctors treats multiple patients.
- Pays: This is a one-to-many relationship from patient to payment when one patient can make multiple payments.
- Gets: This is a one-to-many relationship from patient to admit where one patient can get admitted multiple times.

TABLE SCREENSHOTS

Data definition or data description language (DDL) is a SQL syntax for creating and changing database objects including tables, indices, and users.

- CREATE – to create a new table or database.
- ALTER – for alteration.
- TRUNCATE – to delete data from the table.
- DROP – to drop a table.
- RENAME – to rename a table.

```
SQL> create table appointment(a_id int PRIMARY KEY,a_date date);

Table created.
```

```
SQL> create table patient(p_id int PRIMARY KEY,p_name varchar2(25) NOT
NULL,p_dob date,p_gender varchar2(10),p_email varchar2(25),p_pin int ,
p_state varchar2(25));

Table created.
```

```
SQL> create table payment(p_id int , pay_id int , pay_amt int , foreign key(p_id) references patient(p_id));

Table created.
```

```
SQL> create table pat_ph(p_id int,p_ph int,FOREIGN KEY (p_ID)REFERENCES patient(p_ID));

Table created.
```

```
SQL> create table room(r_id varchar2(25),r_type varchar2(25) default 'normal',r_charges int);

Table created.
```

```
SQL> create table admit(ad_id varchar2(25),ad_date date);

Table created.
```

```
SQL> create table department(dept_id int PRIMARY KEY, dept_name varchar
2(25) NOT NULL);

Table created.
```

```
SQL> create table doctor(doc_id int PRIMARY KEY,doc_name varchar2(25) N
OT NULL,doc_email varchar2(25),doc_gender varchar2(10) NOT NULL,doc_adP
IN int NOT NULL,doc_adSTATE varchar2(25) default 'hyderabad');

Table created.
```

```
SQL> create table doc_ph(doc_id int,doc_ph int,FOREIGN KEY (DOC_ID)REFERENCES DOCTOR(DOC_ID));

Table created.
```

A data manipulation language (DML) is a type of computer programming language that is used to add (insert), delete, and alter (update) data in a database.

- INSERT – is used to insert data into a table.
- UPDATE – is used to update existing data within a table.
- DELETE – is used to delete records from a database table.

UPDATE:

```
SQL> update doctor set doc_adState='hyderabad' where doc_id=0069;

1 row updated.

SQL> update doctor set doc_adState='hyderabad' where doc_id=0020;

1 row updated.
```

DELETE:

```
SQL> delete from room where r_charges=2000;

1 row deleted.

SQL> delete from room where r_charges=2500;

1 row deleted.

SQL> delete from room where r_charges=4000;

1 row deleted.
```

INSERT:

```
SQL> insert into appointment values(&a_id,'&a_date');
Enter value for a_id: 9701
Enter value for a_date: 10-jan-2020
old   1: insert into appointment values(&a_id,'&a_date')
new   1: insert into appointment values(9701,'10-jan-2020')

1 row created.

SQL> /
Enter value for a_id: 9702
Enter value for a_date: 14-mar-2021
old   1: insert into appointment values(&a_id,'&a_date')
new   1: insert into appointment values(9702,'14-mar-2021')

1 row created.

SQL> /
Enter value for a_id: 9703
Enter value for a_date: 29-jun-2018
old   1: insert into appointment values(&a_id,'&a_date')
new   1: insert into appointment values(9703,'29-jun-2018')

1 row created.

SQL> /
Enter value for a_id: 9704
Enter value for a_date: 09-dec-2019
old   1: insert into appointment values(&a_id,'&a_date')
new   1: insert into appointment values(9704,'09-dec-2019')

1 row created.
```

```
SQL> insert into pat_ph values(98047,9988002277);

1 row created.

SQL> insert into pat_ph values(98005,8888002277);

1 row created.

SQL> insert into pat_ph values(98019,7788002277);

1 row created.
```

```
SQL> insert into department values(&dept_id,'&dept_name');
Enter value for dept_id: 101
Enter value for dept_name: Orthopedics
old   1: insert into department values(&dept_id,'&dept_name')
new   1: insert into department values(101,'Orthopedics')

1 row created.

SQL> /
Enter value for dept_id: 102
Enter value for dept_name: Radiology
old   1: insert into department values(&dept_id,'&dept_name')
new   1: insert into department values(102,'Radiology')

1 row created.

SQL> /
Enter value for dept_id: 103
Enter value for dept_name: Neurology
old   1: insert into department values(&dept_id,'&dept_name')
new   1: insert into department values(103,'Neurology')

1 row created.

SQL> /
Enter value for dept_id: 104
Enter value for dept_name: Cardiology
old   1: insert into department values(&dept_id,'&dept_name')
new   1: insert into department values(104,'Cardiology')

1 row created.

SQL> /
Enter value for dept_id: 105
Enter value for dept_name: Gynaecology
old   1: insert into department values(&dept_id,'&dept_name')
new   1: insert into department values(105,'Gynaecology')

1 row created.
```

```
SQL> insert into payment values(&p_id,&pay_id,&pay_amt);
Enter value for p_id: 98047
Enter value for pay_id: 110001
Enter value for pay_amt: 1050
old   1: insert into payment values(&p_id,&pay_id,&pay_amt)
new   1: insert into payment values(98047,110001,1050)

1 row created.

SQL> /
Enter value for p_id: 98005
Enter value for pay_id: 110003
Enter value for pay_amt: 22000
old   1: insert into payment values(&p_id,&pay_id,&pay_amt)
new   1: insert into payment values(98005,110003,22000)

1 row created.
```

```
SQL> insert into room (r_id,r_charges) values ('R01',1000);

1 row created.

SQL> insert into room (r_id,r_charges) values ('R02',1500);

1 row created.

SQL> insert into room (r_id,r_type,r_charges) values ('R02','private',4500);

1 row created.

SQL> select * from room;
```

VALUES AFTER INSERTION:

```
SQL> select * from pat_ph;

      P_ID       P_PH
---------- ----------
     98047 9988002277
     98005 8888002277
     98019 7788002277
```

```
SQL> select * from payment;

      P_ID     PAY_ID    PAY_AMT
---------- ---------- ----------
     98047     110001       1050
     98005     110003      22000
     98019      11023       7500
```

```
SQL> select * from patient;

      P_ID P_NAME                    P_DOB     P_GENDER
---------- ------------------------- --------- ----------
P_EMAIL                       P_PIN P_STATE
------------------------- ---------- -------------------------
     98047 tharun bhaskar            07-JUL-90 male
keedakola@gmail.com           500000 telangana

     98005 prashanth                 12-DEC-98 male
naasavnensastha@gmail.com     720100 tamil nadu

     98019 chitra                    29-JAN-00 female
foodtruck@gmail.com           420100 delhi
```

```
SQL> select * from department;

   DEPT_ID DEPT_NAME
---------- ------------------------
       101 Orthopedics
       102 Radiology
       103 Neurology
       104 Cardiology
       105 Gynaecology
```

```
SQL> select * from doc_ph;

    DOC_ID     DOC_PH
---------- ----------
        17 9828385601
        20 6739425866
        69 7693248566
         2 6969784525
         1 8143431869
        43 9768543577

6 rows selected.
```

```
SQL> select * from room;

R_ID                    R_TYPE                    R_CHARGES
----------------------- ------------------------- ----------
R01                     normal                          1000
R02                     normal                          1500
R03                     private                         4500
```

```
SQL> select * from doctor;

    DOC_ID DOC_NAME                   DOC_EMAIL                 DOC_GENDER
---------- ------------------------- ------------------------- ----------
 DOC_ADPIN DOC_ADSTATE
---------- -------------------------
        17 kaushik                   nenacotoravtha@gmail.com  male
    500012 hyderabad

        69 vivek                     thagudham@gmail.com       male
    500019 hyderabad

        20 karthik                   develop@gmail.com         male
    500072 hyderabad


    DOC_ID DOC_NAME                   DOC_EMAIL                 DOC_GENDER
---------- ------------------------- ------------------------- ----------
 DOC_ADPIN DOC_ADSTATE
---------- -------------------------
         2 uppi                      bahubalivfx@gmail.com     male
    500014

         1 rebecca                   rebecaaa@gmail.com        female
    720019 goa

        43 shirley                   angelakka@gmail.com       female
    720016 chennai


6 rows selected.
```

```
SQL> select * from appointment;

      A_ID A_DATE
---------- ---------
      9701 10-JAN-20
      9702 14-MAR-21
      9703 29-JUN-18
      9704 09-DEC-19
```

```
SQL> select * from admit;

AD_ID                  AD_DATE
---------------------- ---------
AD001                  01-JAN-22
AD002                  29-APR-23
AD003                  30-JUL-21
```

OPERATORS:

An operator is a reserved word or character that is used primarily in the WHERE clause of a SQL statement to execute operation(s) such as comparisons and arithmetic operations. These Operators are used in SQL statements to express conditions and as conjunctions for multiple conditions.

- Arithmetic operators
- Comparison operators
- Logical operators

```
SQL> select * from payment where pay_amt > 5000;

      P_ID     PAY_ID    PAY_AMT
---------- ---------- ----------
     98005     110003      22000
     98019      11023       7500

SQL> select * from payment where pay_id = 11023;

      P_ID     PAY_ID    PAY_AMT
---------- ---------- ----------
     98019      11023       7500

SQL> select * from room where r_charges >= 1500;

R_ID                    R_TYPE                    R_CHARGES
----------------------- ------------------------- ----------
R02                     normal                          1500
R03                     private                         4500
```

```
SQL> alter table payment add (tax number(10));

Table altered.
```

```
SQL> update payment set tax = 249 where p_id = 98047;

1 row updated.

SQL> update payment set tax = 1152 where p_id = 98005;

1 row updated.

SQL> update payment set tax = 479 where p_id = 98019;

1 row updated.

SQL> select * from payment;

      P_ID     PAY_ID    PAY_AMT        TAX
---------- ---------- ---------- ----------
     98047     110001       1050        249
     98005     110003      22000       1152
     98019      11023       7500        479

SQL> select pay_amt+tax as totfee from payment;

    TOTFEE
----------
      1299
     23152
      7979
```

```
SQL> select * from doctor where doc_adstate LIKE 'h%';

    DOC_ID DOC_NAME                  DOC_EMAIL                 DOC_GENDER
---------- ------------------------- ------------------------- ----------
 DOC_ADPIN DOC_ADSTATE
---------- -------------------------
        17 kaushik                   nenacotoravtha@gmail.com  male
    500012 hyderabad

        69 vivek                     thagudham@gmail.com       male
    500019 hyderabad

        20 karthik                   develop@gmail.com         male
    500072 hyderabad


    DOC_ID DOC_NAME                  DOC_EMAIL                 DOC_GENDER
---------- ------------------------- ------------------------- ----------
 DOC_ADPIN DOC_ADSTATE
---------- -------------------------
         2 uppi                      bahubalivfx@gmail.com     male
    500014 hyderabad
```

```
SQL> select * from patient where p_gender='male' AND p_state='telangana';

      P_ID P_NAME                    P_DOB     P_GENDER
---------- ------------------------- --------- ----------
P_EMAIL                       P_PIN P_STATE
------------------------- ---------- -------------------------
     98047 tharun bhaskar            07-JUL-90 male
keedakola@gmail.com           500000 telangana


SQL> select * from patient where p_state IN ('delhi','female');

      P_ID P_NAME                    P_DOB     P_GENDER
---------- ------------------------- --------- ----------
P_EMAIL                       P_PIN P_STATE
------------------------- ---------- -------------------------
     98019 chitra                    29-JAN-00 female
foodtruck@gmail.com           420100 delhi
```

CHARACTER FUNCTIONS:

Character functions receive character input and can output either characters or numbers. SQL supports a variety of character datatypes, including CHAR, VARCHAR, VARCHAR2, LONG, RAW, and LONG RAW.

SQL has a robust collection of character functions that enable you to obtain information about strings and alter their contents in a variety of ways.

Character functions are classified into two types:

1. Case-Making Functions (LOWER, UPPER, and INITCAP)

2. Functions for manipulating characters (CONCAT, LENGTH, SUBSTR, INSTR, LPAD, RPAD, TRIM, and REPLACE)

NUMERIC FUNCTIONS:

Numeric Functions are used to manipulate numbers and return numbers.

A few examples of numerical functions are: ABS(), MAX(), MIN(), and so on.

Numerous predefined aggregate functions in SQL may be used to create searches that yield information of this nature.

When aggregating data from a data table, the arrange BY clause explains how to arrange the rows while the HAVING clause filters out rows that do not belong in the provided categories.

Among the many tasks that aggregate functions carry out include counting all the rows in a table, averaging the data in a column, and adding together numerical data. Additionally, they may use a table search to locate the greatest "MAX" or lowest "MIN" values in a given field.

FUNCTIONS:

```
SQL> SELECT UPPER(p_name) FROM patient;

UPPER(P_NAME)
-------------------------
THARUN BHASKAR
PRASHANTH
CHITRA
```

```
SQL> SELECT LOWER(p_name) FROM patient;

LOWER(P_NAME)
-------------------------
tharun bhaskar
prashanth
chitra
```

```
SQL> SELECT MAX(pay_amt) as highest_amount_paid FROM payment;

HIGHEST_AMOUNT_PAID
-------------------
              22000

SQL> SELECT MIN(pay_amt) as highest_amount_paid FROM payment;

HIGHEST_AMOUNT_PAID
-------------------
               1050

SQL> SELECT p_id, MAX(pay_amt) as max_amount  FROM payment GROUP BY p_id;

      P_ID MAX_AMOUNT
---------- ----------
     98047       1050
     98005      22000
     98019       7500
```

JOINS:

The SQL Joins clause is used to combine records from two or more tables in a database. A JOIN is a means for combining fields from two tables by using values common to each. There are different types of joins available in SQL −

- INNER JOIN − returns rows when there is a match in both tables.
- LEFT JOIN − returns all rows from the left table, even if there are no matches in the right table.
- RIGHT JOIN − returns all rows from the right table, even if there are no matches in the left table.
- FULL JOIN − returns rows when there is a match in one of the tables.
- SELF JOIN − is used to join a table to itself as if the table were two tables, temporarily renaming at least one table in the SQL statement.

```
SQL> select doctor.doc_id,doctor.doc_name,department.dept_id,department.dept_name
 from doc_dep inner join doctor on doc_dep.doc_id=doctor.doc_id inner join
department on doc_dep.dept_id = department.dept_id;

    DOC_ID DOC_NAME                    DEPT_ID DEPT_NAME
---------- ------------------------ ---------- ------------------------
        17 kaushik                         102 Radiology
        69 vivek                           102 Radiology
        20 karthik                         103 Neurology
         2 uppi                            104 Cardiology
        43 shirley                         104 Cardiology
         1 rebecca                         105 Gynaecology

6 rows selected.
```

```
SQL> SELECT patient.p_name, doctor.doc_name FROM patient full JOIN doctor ON patient.p_pin = doctor.doc_adPIN;

P_NAME                   DOC_NAME
------------------------ ------------------------
                         kaushik
                         vivek
                         karthik
                         uppi
                         rebecca
                         shirley
chitra
prashanth
tharun bhaskar

9 rows selected.
```

```
SQL> create table doc_dep(doc_id int,dept_id int,primary key(doc_id,dept_id),
 foreign key(doc_id) references doctor(doc_id), foreign key (dept_id) referen
ces department(dept_id));

Table created.

SQL> insert into doc_dep(doc_id,dept_id) values (17,102)
  2
SQL> insert into doc_dep(doc_id,dept_id) values (17,102);

1 row created.

SQL> insert into doc_dep(doc_id,dept_id) values (69,102);

1 row created.

SQL> insert into doc_dep(doc_id,dept_id) values (20,103);

1 row created.

SQL> insert into doc_dep(doc_id,dept_id) values (2,104);

1 row created.

SQL> insert into doc_dep(doc_id,dept_id) values (1,105);

1 row created.

SQL> insert into doc_dep(doc_id,dept_id) values (43,104);

1 row created.
```

```
SQL> SELECT patient.p_name, doctor.doc_name FROM patient RIGHT JOIN doctor ON patient.p_pin = doctor.doc_adPIN;

P_NAME                   DOC_NAME
------------------------ ------------------------
                         kaushik
                         vivek
                         rebecca
                         uppi
                         shirley
                         karthik
```

```
      P_ID P_NAME                    P_DOB     P_GENDER
---------- ------------------------- --------- ----------
P_EMAIL                       P_PIN P_STATE                       DOC_ID
------------------------- ---------- ------------------------- ----------
DOC_NAME                  DOC_EMAIL                 DOC_GENDER  DOC_ADPIN
------------------------- ------------------------- ---------- ----------
DOC_ADSTATE
-------------------------
     98005 prashanth                12-DEC-98 male
naasavnensastha@gmail.com     720100 tamil nadu




      P_ID P_NAME                    P_DOB     P_GENDER
---------- ------------------------- --------- ----------
P_EMAIL                       P_PIN P_STATE                       DOC_ID
------------------------- ---------- ------------------------- ----------
DOC_NAME                  DOC_EMAIL                 DOC_GENDER  DOC_ADPIN
------------------------- ------------------------- ---------- ----------
DOC_ADSTATE
-------------------------
     98047 tharun bhaskar           07-JUL-90 male
keedakola@gmail.com           500000 telangana
```

```
SQL> select * from patient left join doctor on patient.p_pin=doctor.doc_adpin;

      P_ID P_NAME                    P_DOB     P_GENDER
---------- ------------------------- --------- ----------
P_EMAIL                       P_PIN P_STATE                       DOC_ID
------------------------- ---------- ------------------------- ----------
DOC_NAME                  DOC_EMAIL                 DOC_GENDER  DOC_ADPIN
------------------------- ------------------------- ---------- ----------
DOC_ADSTATE
-------------------------
     98019 chitra                   29-JAN-00 female
foodtruck@gmail.com           420100 delhi
```

SUBQUERY:

A subquery is used to return data that will be used in the main query as a condition to further restrict the data to be retrieved.

Subqueries can be used with the SELECT, INSERT, UPDATE, and DELETE statements along with the operators like =, <, >, >=, <=, etc.

```
SQL> select  doc_id,doc_name,doc_email from doctor where doc_adstate=(select
 doc_adstate from doctor where doc_name='vivek') AND doc_id > (select doc_id
 from doctor where doc_name='rebecca');

    DOC_ID DOC_NAME                   DOC_EMAIL
---------- ------------------------- -------------------------
         2 uppi                      bahubalivfx@gmail.com
        17 kaushik                   nenacotoravtha@gmail.com
        20 karthik                   develop@gmail.com
        69 vivek                     thagudham@gmail.com
```

```
SQL> select p_id,pay_id,pay_amt from payment where tax=(select MIN(tax) from
 payment);

      P_ID     PAY_ID    PAY_AMT
---------- ---------- ----------
     98047     110001       1050
```

```
SQL> select doc_name,doc_gender,doc_adstate from doctor where doc_adpin < ALL
(select doc_adpin from doctor where doc_adpin=720019) AND doc_adpin <> 720019;


DOC_NAME                  DOC_GENDER DOC_ADSTATE
------------------------- ---------- -------------------------
kaushik                   male       hyderabad
vivek                     male       hyderabad
karthik                   male       hyderabad
uppi                      male       hyderabad
shirley                   female     chennai
```

```
SQL> select p_id,AVG(tax) from payment group by p_id having AVG(tax)=(select
 MIN(AVG(tax)) from payment group by p_id);

      P_ID   AVG(TAX)
---------- ----------
     98047        249
```

```
SQL> select p_id,p_name,p_dob from patient where p_gender=(select p_gender from
patient where p_pin=720100);

      P_ID P_NAME                    P_DOB
---------- ------------------------- ---------
     98047 tharun bhaskar            07-JUL-90
     98005 prashanth                 12-DEC-98
```

```
SQL> select doc_name,doc_gender,doc_adstate from doctor where doc_adpin IN
(select MIN(doc_adpin) from doctor group by doc_adstate);

DOC_NAME                  DOC_GENDER DOC_ADSTATE
------------------------- ---------- -------------------------
kaushik                   male       hyderabad
rebecca                   female     goa
shirley                   female     chennai
```

```
SQL> select p_id,p_name,p_dob,p_email from patient where p_pin > ANY (select
 p_pin from patient where p_id=98019) AND p_gender <> 'female';

      P_ID P_NAME                    P_DOB     P_EMAIL
---------- ------------------------- --------- -------------------------
     98047 tharun bhaskar            07-JUL-90 keedakola@gmail.com
     98005 prashanth                 12-DEC-98 naasavnensastha@gmail.com
```

PLSQL:

PL/SQL is a combination of SQL along with the procedural features of programming languages. It was developed by Oracle Corporation in the early 90's to enhance the capabilities of SQL. PL/SQL is one of three key programming languages embedded in the Oracle Database, along with SQL itself and Java. This tutorial will give you great understanding on PL/SQL to proceed with Oracle database and other advanced RDBMS concepts.

```
SQL> DECLARE
  2      total_amount NUMBER := 0;
  3  BEGIN
  4      SELECT SUM(pay_amt) INTO total_amount
  5      FROM payment;
  6      DBMS_OUTPUT.PUT_LINE('Total = ' || total_amount);
  7  END;
  8  /
Total = 30550
```

TRIGGER:

A trigger is a stored procedure in database which automatically invokes whenever a special event in the database occurs. For example, a trigger can be invoked when a row is inserted into a specified table or when certain table columns are being updated.

```
SQL> CREATE OR REPLACE TRIGGER payment_update_trigger
  2  BEFORE UPDATE ON payment
  3  FOR EACH ROW
  4  DECLARE
  5    v_pay_tax CONSTANT NUMBER := 0.1;
  6  BEGIN
  7    :NEW.pay_amt := :NEW.pay_amt + (:NEW.pay_amt * v_pay_tax);
  8    DBMS_OUTPUT.PUT_LINE('Payment updated successfully. New amount: ' || :NEW.pay_amt);
  9  END;
 10  /

Trigger created.

SQL> UPDATE payment SET pay_amt = 1000 WHERE pay_id = 110003;

1 row updated.
```

CURSOR:

A cursor is a pointer to this context area. PL/SQL controls the context area through a cursor. A cursor holds the rows (one or more) returned by a SQL statement. The set of rows the cursor holds is referred to as the active set

```
SQL> DECLARE
  2    payid payment.pay_id%TYPE := 110001;
  3    payamt payment.pay_amt%TYPE;
  4    pay_tax CONSTANT NUMBER := 0.1;
  5  BEGIN
  6
  7    SELECT pay_amt INTO payamt
  8    FROM payment
  9    WHERE pay_id = payid;
 10
 11
 12    payamt := payamt + (payamt * pay_tax);
 13
 14
 15    UPDATE payment
 16    SET pay_amt = payamt
 17    WHERE pay_id = payid;
 18
 19    DBMS_OUTPUT.PUT_LINE('Payment updated successfully. New amount: ' || payamt);
 20  END;
 21  /
Payment updated successfully. New amount: 1155

PL/SQL procedure successfully completed.
```