**K. J. Somaiya College of Engineering, Mumbai-77**
**(Autonomous College Affiliated to University of Mumbai)**

**Batch: B1** **Roll No.: 1711072**

**Experiment / assignment / tutorial No. 3**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Title:** Implementation of Database in SQL -DDL

**Objective:** Define/modify database definitions with proper constraints

---

**Expected Outcome of Experiment:**
CO 2: Convert entity-relationship diagrams into relational tables, populate a relational database and formulate SQL queries on the data Use SQL for creation and query the database.

CO 3: Define and apply integrity constraints and improve database design using normalization techniques.

---

**Books/ Journals/ Websites referred:**

1. Sharaman Shah,"*Oracle for Professional*", SPD.
2. Dr. P.S. Deshpande, SQL and PL/SQL for Oracle 10g.Black book, Dreamtech Press
3. Korth, Slberchatz, Sudarshan: "Database Systems Concept", 5th Edition , McGraw Hill
4. Peter Rob and Carlos Coronel,"Database Systems Design, Implementation and Management", Thompson Learning, 5th Edition

**Pre Lab/ Prior Concepts**:
Resources used: Postgresql

**Theory:** The set of relations in a database must be specifies to the system by means of a data definition language (DDL). The SQL DDL allows specification of not only a set of relations but also specific information about the relation including,

1. The schema for each relation
2. The domain of values associated with each attribute
3. The integrity constraints
4. The set of indices to be maintained for each relation
5. The security and authorization information for each relation
6. The physical storage structure of each relation on disk

**Syntax Create Table:**

create table employee(ssn,fname varchar(10), mname varchar(10), lname varchar(10),

desg varchar(20), gender varchar(5), addr varchar(20), bdate datetime, sal float,primary

key(ssn));

create table manages(ssn int, dept_code int, start_dt datetime, foreign key(ssn)

create table manages(ssn int, dept_code int, start_dt datetime, foreign key(ssn)

references employee, foreign key(dept_code) refrences department, key(ssn,dept_code) ) on delete set null;primary

**Data Constraints**

Busines managers of the organization determine the a set of rules that must be applied before the data is stored in the database. The application of such rules on raw data ensures **data integrity**.

**Eg:-** An employee belonging to Sales department cannot have salary higher than Rs. 1000.

An employee has an unique identification number.

## Applying Data Constraints

Oracle permits data constraints to be attached to table columns using SQL syntax.

Constraints can be attached to table columns using

Alter table

**Unique Constraint**

**Unique Constraint- At column level Syntax**

**<ColumnName><Datatype>(<size>)**

**UNIQUE Unique Constraint- At table level**

**CREATE TABLE<TableName>(**

**<ColumnName><Datatype>(<size>)**

**<ColumnName><Datatype>(<size>)**

**<Columnname><Datatype>(<size>)**

**UNIQUE(<ColumnName1>,<ColumnName2>);**

**Implementation Details (Problem Statement, Query and Screenshots of Results):**

Problem Statement:

People seek information via words, recommendation letters, etc. The search engines retrieve this information using specific keywords mentioned by the user. Recommendation systems imitate this social process to enable quick filtering of the information on the web.

We aim to develop a database for this movie recommendation system which will contain the details of the following:

1. Movie, like the title, director, movie length;

2. Genre: the basis on which movies will be suggested.

3. Customer: customer id, basic details

4. Theatre: name, address, timings

5. Cast: name, age, no. of movies done

6. Reviews: rating by customer, movie analyst

This database will serve as a basis for our movie recommendation system on the basis of user's genre preference and review of the movies.

Sample Queries:

```
postgres=# create database recommender;
CREATE DATABASE
postgres=# \c recommender;
You are now connected to database "recommender" as user "postgres".

recommender=# create table Actor(cast_id int primary key,name
varchar(30) not null,age int not null,no_of_movies int);
CREATE TABLE
recommender=# ALTER TABLE Actor
recommender-# RENAME TO Cast_Movie;
ALTER TABLE
recommender=# create table customer(email_id varchar(40) primary
key, name varchar(30) not null, age int not null);
CREATE TABLE
recommender=# create table movie(movie_id int primary
key,movie_name varchar(60) not null, release_date date not
null,rating float not null);CREATE TABLE
recommender=# create table theatre(theatre_id int primary key,
t_name varchar(30) not null, location varchar(20) not null);
CREATE TABLE
```

```
recommender=# create table review(mov_id int,cust_review
varchar(200), analyst_review varchar(200),foreign key(mov_id)
references movie(movie_id));
CREATE TABLE
recommender=# ALTER TABLE movie
recommender-# ADD genre varchar(15);
ALTER TABLE
recommender=# ALTER TABLE movie ALTER COLUMN genre SET NOT NULL;
ALTER TABLE
```

Screenshots:

```
postgres=# \c recommender
You are now connected to database "recommender" as user "postgres".
recommender=# \dt
           List of relations
 Schema |    Name    | Type  |  Owner
--------+------------+-------+----------
 public | cast_movie | table | postgres
 public | customer   | table | postgres
 public | movie      | table | postgres
 public | review     | table | postgres
 public | theatre    | table | postgres
(5 rows)
```

```
recommender=# \d cast_movie
                    Table "public.cast_movie"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 cast_id      | integer               |           | not null |
 name         | character varying(30) |           | not null |
 age          | integer               |           | not null |
 no_of_movies | integer               |           |          |
Indexes:
    "actor_pkey" PRIMARY KEY, btree (cast_id)
```

```
recommender=# \d customer
                    Table "public.customer"
  Column  |         Type          | Collation | Nullable | Default
----------+-----------------------+-----------+----------+---------
 email_id | character varying(40) |           | not null |
 name     | character varying(30) |           | not null |
 age      | integer               |           | not null |
Indexes:
    "customer_pkey" PRIMARY KEY, btree (email_id)
```

```
recommender=# \d movie
                        Table "public.movie"
    Column    |         Type          | Collation | Nullable | Default
--------------+-----------------------+-----------+----------+---------
 movie_id     | integer               |           | not null |
 movie_name   | character varying(60) |           | not null |
 release_date | date                  |           | not null |
 rating       | double precision      |           | not null |
 genre        | character varying(15) |           | not null |
Indexes:
    "movie_pkey" PRIMARY KEY, btree (movie_id)
Referenced by:
    TABLE "review" CONSTRAINT "review_mov_id_fkey" FOREIGN KEY (mov_id) REFERENC
ES movie(movie_id)
```

```
recommender=# \d review
                       Table "public.review"
     Column     |          Type          | Collation | Nullable | Default
----------------+------------------------+-----------+----------+---------
 mov_id         | integer                |           |          |
 cust_review    | character varying(200) |           |          |
 analyst_review | character varying(200) |           |          |
Foreign-key constraints:
    "review_mov_id_fkey" FOREIGN KEY (mov_id) REFERENCES movie(movie_id)
```

```
recommender=# \d theatre
                        Table "public.theatre"
   Column   |          Type          | Collation | Nullable | Default
------------+------------------------+-----------+----------+---------
 theatre_id | integer                |           | not null |
 t_name     | character varying(30) |           | not null |
 location   | character varying(20) |           | not null |
Indexes:
    "theatre_pkey" PRIMARY KEY, btree (theatre_id)
```

**Conclusion: The database was successfully created and tables were added to the database with required attributes, primary keys and foreign keys using DDL commands in PostgreSQL.**